PHYSICS-INFORMED ONE-STEP GENERATION ON LONG-SHORT FLOW MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose Physics-Informed Long-Short Flow Maps with Direct Guidance (PI-LSFM), a one-step generative method for producing data-aligned and physically valid samples. Existing one-step physics-informed generative models must jointly control data fidelity and physical constraints without the iterative correction of multi-step samplers, making it difficult to coordinate indirect surrogate supervision with strong physics enforcement. PI-LSFM addresses this challenge by decomposing generation into a long flow map for global transport and a near-terminal short flow map, where closed-form data and physics potentials provide directly computable guidance targets instead of indirect regression surrogates. The short map guides the long map during training, yielding a single one-step generator that requires neither staged distillation nor inference-time correction. Across Darcy flow, Kolmogorov flow, dynamic stall, and topology optimization, PI-LSFM achieves a favorable trade-off among distributional fidelity, physical residuals, and inference efficiency while using only one function evaluation at inference.

1 INTRODUCTION

Many problems in scientific machine learning are inherently distributional: under uncertain boundary conditions, perturbed parameters, incomplete observations, or intrinsic stochasticity, one seeks not a single deterministic solution, but a distribution of physically feasible ones. This makes generative modeling a natural framework for physical systems, as it can capture high-dimensional solution distributions while incorporating PDEs, conservation laws, and other physical constraints. Recent works have therefore introduced physics-informed diffusion and related generative frameworks to produce samples that are both statistically plausible and physically valid Shu et al. (2023); Huang et al. (2024); Jacobsen et al. (2025). However, these methods typically rely on iterative multi-step generation and are often slow for practical use, which naturally motivates fast and ideally one-step physics-informed generation.

The main difficulty, however, is not merely that one-step generation is harder to train, but that existing physics-informed diffusion methods rely on indirect supervision for both data and physics. On the data side, flow matching learns the data distribution through the surrogate regression objective $\mathcal{L}_{\text{data}} = \mathbb{E}[|u_t^\theta(x_t) - u_t(x_t|x_1)|^2]$, where the model is supervised by the conditional velocity $u_t(x_t|x_1)$ because the true marginal velocity $u_t(x_t)$ is unavailable. On the physics side, directly applying a nonlinear residual to intermediate states causes a Jensen-type mismatch Bastek et al. (2024); existing methods therefore impose the residual on a reconstructed endpoint, $\mathcal{L}_{\text{phys}} = \mathbb{E}[|\mathcal{R}(\hat{x}_1(x_t))|^2]$, and use it to supervise intermediate states indirectly. This double indirectness makes distributional and physical objectives difficult to coordinate Baldan et al. (2026). The issue is amplified in one-step methods such as MeanFlow Geng et al. (2025) and ShortCut Frans et al. (2025). A long-range flow map $\psi_{t\to r}$ is difficult to learn and, unlike an instantaneous velocity, lacks a consistent data-conditioned counterpart $\psi_{t\to r}(\cdot|x_1)$ for surrogate supervision Li et al. (2025; 2026). Moreover, one-step sampling removes the iterative correction available in multi-step methods Bastek et al. (2024), requiring stronger physics enforcement during training. Thus, one-step physics-informed generation must couple a harder indirect distribution objective with a stronger physics objective, making training fragile. Even in endpoint-prediction methods such as sCM Lu & Song (2024), where the residual can be applied directly to $\psi_{t\to 1}(x_t)$, the data objective remains indirect. Chang et al. (2026) show that coupling strong physics constraints with indirect distribution objectives can lead to trivial solutions, severe mode collapse, and the need for two-stage distillation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

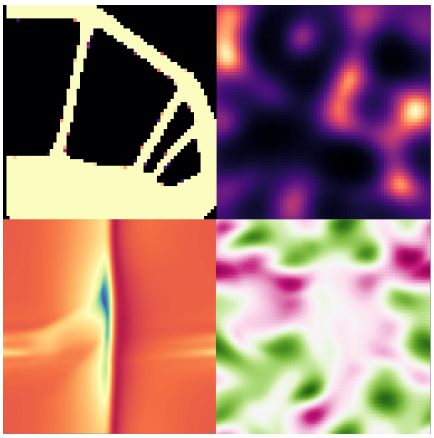


Figure 1: PI-LSFM enables physics-informed one-step generation from scratch via long-short flow-map direct guidance, achieving multi-step-level performance on unconditional generation, conditional generation, and inverse design without increasing batch size or training steps, and without extra inference-time parameters or iterative correction.

In this work, we study physics-informed one-step generation in flow-map-based frameworks and aim to enable stable training from scratch through direct data-distribution and physics-constraint objectives. This is difficult in existing methods: closed-form flow-matching solutions are typically too inefficient for practical training Bertrand et al. (2025), while directly imposing physical constraints at intermediate states leads to Jensen's gap. Our key observation is that both objectives can be imposed directly in the near-terminal data neighborhood, where they provide effective and correct guidance for learning the full flow map. This motivates a long-short flow-map approach, which decomposes one-step generation into a long flow map for global evolution and a near-terminal short flow map driven by computable closed-form solutions and physics potentials. As $\Delta t \to 0$, the short flow map shrinks away while providing direct data-distribution and physics-constraint guidance for learning the long flow map. This yields Long-Short Flow Maps with Direct Guidance (PI-LSFM), where data learning and physical validity are guided by near-terminal, directly computable targets rather than conditional-regression surrogates or reconstructed terminal states. A detailed discussion of related work is provided in Appendix A.

Our contributions: (1) We propose a long-short flow-map framework that addresses indirect-objective instability by modeling the short flow map as dynamics jointly driven by data and physics potentials, enabling direct guidance from distributional and physical constraints for one-step generation. (2) We study the numerical design and computation of the short flow map, leading to a stable and efficient training procedure with improved distributional fidelity and physical consistency. (3) We validate PI-LSFM on unconditional generation, conditional generation, and topology optimization, where it consistently achieves strong performance on practical physics-informed generation tasks.

2 Preliminaries and Motivations

2.1 Physics-Informed Generation and Indirect Objectives

Given a dataset $\mathcal{D} = \{x^i \in \mathcal{X}\}_{i=1}^n$ drawn from an unknown distribution p_{data} on $\mathcal{X} \subset \mathbb{R}^d$, a physics-informed generative model aims to learn a transport from a base distribution $x_0 \sim p_0$ to the target distribution $x_1 \sim p_1 = p_{\text{data}}$, while enforcing the generated samples to satisfy a physical constraint $\mathcal{R}(x_1) = 0$. A common continuous-time formulation, exemplified by Flow Matching, parameterizes a velocity field $u^\theta(x,t)$ and defines generation through the ODE

$$\frac{dx_t}{dt} = u^\theta(x_t, t), \qquad x_0 \sim p_0. \tag{1}$$

For training the model, modern generative methods, including diffusion and flow-based models, generally do not admit direct supervision for the final generative transport from the dataset, since no efficiently computable closed-form solution is available (see subsection F.2), and therefore rely on surrogate training objectives. For example, Flow Matching constructs a conditional path $p_t(x_t|x_1)$ with $x_1 \sim p_1$ and uses the corresponding conditional velocity $u_t(x_t|x_1) = \frac{x_1 - x_t}{1-t}$ as a surrogate target, with $u_t(x_t) = \mathbb{E}_{x_1 \sim p_{1|t}(x_t)}[u_t(x_t|x_1)]$, and optimizes

$$\mathcal{L}_{\text{data}}(\theta) = \mathbb{E}_{t, x_1 \sim p_1, x_t \sim p_{t|1}(x_t|x_1)}[\|u^\theta(x_t,t) - u_t(x_t|x_1)\|^2] \tag{2}$$

To enforce $\mathcal{R}(x_1) = 0$, prior work shows that directly imposing the constraint on the intermediate state x_t, or on its instantaneous prediction $x_1(x_t) = u_t^\theta(x_t)(1-t) + x_t$, leads to Jensen's gap $\mathcal{R}(\mathbb{E}[x_1|x_t]) \neq \mathbb{E}[\mathcal{R}(x_1(x_t))|x_t]$ Bastek et al. (2024). Existing methods therefore enforce the residual on a surrogate endpoint $\hat{x}_1^\theta(x_t)$ reconstructed from x_t, typically through a multi-step auxiliary DDIM- or Flow-Matching-style inference process, and optimize the indirect physics loss $\mathcal{L}_{phys}(\theta) = \mathbb{E}_{t, x_1 \sim p_1, x_t \sim p_{t|1}(x_t|x_1)}[\|\mathcal{R}(\hat{x}_1^\theta(x_t))\|^2]$ with respect to θ, encouraging the final generated sample associated with x_t to satisfy the physical constraint.

In practice, these two indirect objectives can pull optimization toward either better data fidelity or better constraint satisfaction, often at the expense of the other Bastek et al. (2024); Jacobsen et al. (2025); Utkarsh et al. (2025). Recent work Baldan et al. (2026) mitigates this issue in multi-step generative models through conflict-free optimization. In addition, multi-step sampling with $u^\theta(x,t)$ requires numerically discretizing Equation 1 from $x_0 \sim p_0$ over many steps, which remains expensive in practice and motivates one-step generation.

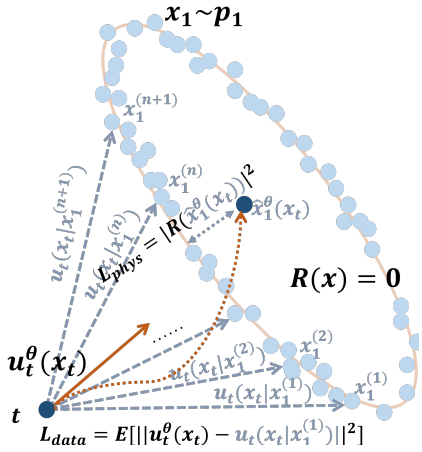


Figure 2: **Indirect Losses.** L_{data} does not directly supervise u_t^θ with its true target $u_t(\cdot)$, but instead regresses to the conditional velocity $u_t(x_t|x_1)$. Likewise, L_{physics} is not applied directly to u_t^θ, but only to the terminal prediction $\hat{x}_1^\theta(x_t)$ obtained after several integration steps. The indirectly supervised quantities are highlighted in muted blue.

2.2 One-Step Generation on Flow Maps

Specifically, flow-map methods, including ShortCut Frans et al. (2025), MeanFlow Geng et al. (2025), and consistency-based models Geng et al. (2024), learn a family of mappings $\psi_{t\to r} : \mathcal{X} \to \mathcal{X}$ along trajectories of Equation 1, such that $\psi_{0\to t}(x_0) = x_t$ and $\psi_{t\to r} = \psi_{0\to r} \circ \psi_{0\to t}^{-1}$. Flow maps satisfy

$$\frac{\partial}{\partial t}\psi_{0\to t} = u_t \circ \psi_{0\to t}, \qquad \psi_{0\to 0} = \mathrm{Id}_{\mathcal{X}}. \tag{3}$$

Accordingly, the marginal probability path $x_t \sim p_t$ can be written as the pushforward $p_t = (\psi_{0\to t})_\sharp p_0$. Once $\psi_{t\to r}$ is learned, flow-map methods enable one-step generation via $x_1 = \psi_{0\to 1}(x_0)$. A defining property of flow maps is their *trajectory consistency*:

Proposition 2.1 (Trajectory Consistency)**.** *For any $0 \le t \le s \le r \le 1$, the flow map satisfies*

$$\psi_{t\to r} = \psi_{s\to r} \circ \psi_{t\to s}. \tag{4}$$

This property can be encouraged by the trajectory-consistency loss $\mathcal{L}^{\mathrm{TC}}(\theta) = \mathbb{E}_{t,s,r,x_t\sim p_t}[\|\psi_{t\to r}^\theta(x_t) - \psi_{s\to r}^\theta(\psi_{t\to s}^\theta(x_t))\|_2^2]$. However, $\mathcal{L}^{\mathrm{TC}}$ itself contains no dataset-level supervision. In particular, introducing data supervision for *long-range* maps $\psi_{t\to r}$ is difficult: unlike the instantaneous velocity $u(x,t)$, such maps generally do not admit an easily computable conditional counterpart $\psi_{t\to r}(\cdot|x_1)$ from data samples over large time gaps; see Theorem F.1. Thus, indirectly learning $\psi_{t\to r}$ from data is substantially harder than supervising local dynamics.

Existing flow-map methods inject dataset supervision through progressive extension or derivative-based objectives; see subsection F.1. For instance, MeanFlow use the averaged velocity $u_{t\to r}(x) = \frac{\psi_{t\to r}(x)-x}{r-t}$ and trains it using a derivative-based surrogate regression target $\|u_{t\to r}^\theta(x) - sg(u_t(x|x_1) + (r-t)(\partial_t u_{t\to r}^\theta(x) + u_t(x|x_1)\partial_x u_{t\to r}^\theta(x)))\|^2$ with the conditional velocity $u_t(x \mid x_1)$. On the physical-supervision side, variable-endpoint methods such as ShortCut and MeanFlow remain problematic: $\psi_{t\to r}(x_t)$ is still an intermediate state, so directly imposing a nonlinear residual suffers from the Jensen-type mismatch. Endpoint-fixed consistency models can apply the residual to $\psi_{t\to 1}(x_t)$, but without multi-step correction they require stronger physics enforcement during training. This strong physics loss interacts poorly with the indirect, high-order flow-map objective, making optimization fragile, as illustrated in Figure 9, motivating the staged distillation of Chang et al. (2026). Table 11 complements this visualization with a comparison of physical residuals across batch sizes in the same toy experiment.

3 Physics-Informed Long-Short Flow-Map Method

To address the above challenges, we propose a **Physics-Informed Long-Short Flow Map with Direct Guidance (PI-LSFM)** framework, which treats dataset-level supervision and physical constraints as direct near-terminal guidance derived from closed-form solutions. Applying such guidance to the full generative transport is difficult, since closed-form data guidance is expensive away from the terminal time and direct physics guidance can suffer from Jensen-type mismatch; see subsection 2.1. Near $t = 1$, however, both issues are alleviated: the closed-form solution becomes efficient to compute (subsection F.2), and physics constraints can be applied at the endpoint, avoiding intermediate-state

Jensen-type mismatch since $\mathcal{R}(\mathbb{E}[x_1 \mid x_t]) = \mathbb{E}[\mathcal{R}(x_1(x_t)) \mid x_t] = \mathcal{R}(x_1)$ at $t = 1$. This motivates our long-short decomposition with Proposition Proposition 2.1.

$$\psi_{0\to 1} = \psi_{1-\Delta t\to 1} \circ \psi_{0\to 1-\Delta t}, \tag{5}$$

where a long flow map $\psi_{0\to 1-\Delta t}$ transports noise close to the data manifold, and a short terminal flow map $\psi_{1-\Delta t\to 1}$ provides direct near-terminal guidance.

To let the short-range terminal flow map $\psi_{1-\Delta t\to 1}$ provide near-terminal direct guidance, we model its dynamics as jointly driven by a data-distribution potential and a physics-constraint potential. We compute this dynamics numerically as $\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)]$, where $\mathcal{I}[f(x,t); s, r, x_0]$ denotes numerical integration of the integrand $f(x,t)$ over $[s,r]$ with initial state x_0. As $\Delta t \to 0$, Equation 5 leads to the following loss:

$$\begin{aligned}\mathcal{L}(\theta) =& \mathbb{E}_{x_0\sim p_0}\|\psi^{\theta}_{0\to 1}(x_0) - \lim_{\Delta t\to 0} sg(\psi^{\theta}_{0\to 1-\Delta t}(x_0) \\ &+\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x); 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)])\|^2 \\ =& \mathbb{E}_{x_0\sim p_0}\|\psi^{\theta}_{0\to 1}(x_0) - sg(\psi^{\theta}_{0\to 1}(x_0) \\ &+ \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x); 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)])\|^2\end{aligned} \tag{6}$$

Here sg denotes the stop-gradient operator. The short-range dynamics defines **a direct guidance target** for $\psi^{\theta}_{0\to 1}(x_0)$ as $\psi^{\theta}_{0\to 1}(x_0) + \mathcal{T}(x_0)$, where $\mathcal{T}(x_0) = \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)]$. This avoids indirect objectives based on regression surrogates or intermediate-state reconstruction. When the limit exists, $\psi^{\theta}_{0\to 1-\Delta t}(x_0)$ inside $\mathcal{T}(x_0)$ converges to $\psi^{\theta}_{0\to 1}(x_0)$, so $\mathcal{L}(\theta)$ depends only on the single flow map $\psi^{\theta}_{0\to 1}(x_0)$ and is therefore self-consistent. Taking $\Delta t \to 0$ is also necessary to avoid Jensen's gap Bastek et al. (2024) and ensure well-defined kernel computation, as summarized in subsection 3.3.

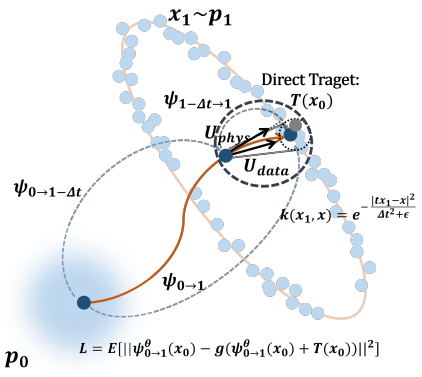


Figure 3: **Long-Short Flow Map.** The flow map $\psi_{0\to 1}$ is decomposed into a long flow map $\psi_{0\to 1-\Delta t}$ and a short flow map $\psi_{1-\Delta t\to 1}$. The short flow map is modeled through dynamics driven by the potentials U_{phys} and U_{data}, providing guidance for the long flow map. As $\Delta t \to 0$, this becomes direct guidance for learning $\psi_{0\to 1}$, integrating both data and physical information.

We now specify the short-range dynamics $\psi_{1-\Delta t\to t}(x)$. We model it as a dynamics jointly driven by a data-distribution potential U_{data} and a physics-constraint potential U_{phys}:

$$\frac{dx_\tau}{d\tau} = -\nabla U_{\text{data}}(x_\tau, \tau) - \lambda \nabla U_{\text{phys}}(x_\tau, \tau), \qquad \tau \in [t, 1]. \tag{7}$$

Here λ is constant coefficient and $x_\tau = \psi_{1-\Delta t\to\tau}(x)$. The data-distribution potential drives the short flow map toward the target data distribution $x_1 \sim p_1$, corresponding to the closed-form optimal Flow Matching solution in Equation 2, whose explicit form is given in subsection 3.1. The physics-constraint potential drives the dynamics toward physical feasibility, i.e., $\mathcal{R}(x) = 0$, and we instantiate it using a remaining-time-normalized constraint energy,

$$U_{\text{phys}}(x, t) = \frac{1}{1-t}\|\mathcal{R}(x)\|^2. \tag{8}$$

The factor $\frac{1}{1-t}$ increasingly emphasizes physical feasibility near the terminal time, guiding the short flow toward the constraint manifold $\mathcal{R}(x) = 0$. Then, we numerically solve Equation 7 and separately examine the data and physics terms in subsection 3.1 and subsection 3.2.

3.1 Numerical Calculation For Data Term

We first present the expression of the data potential U_{data}, which is consistent with the training objective in Equation 2 and admits the following potential-driven formulation.

Theorem 3.1 (Potential-Driven System for Flow Matching)**.** *Assume $p_0 = \mathcal{N}(0, I)$, independent of the data endpoint x_1, and that p_1 is a smooth distribution with $p_1(x) > 0$. The Flow Matching dynamics with velocity u_t, learned in Equation 2, can be interpreted as a dynamical system that*

starts from an initial state $x_0 \sim p_0$ and evolves under a closed-form potential.

$$\frac{d}{dt}x_t = -\nabla U_{\text{data}}(x, t), \tag{9}$$

where $U_{\text{data}}(x,t)$ is defined as

$$U_{\text{data}}(x,t) = -\frac{1-t}{t}\log \mathbb{E}_{x_1\sim p_1}[k_t(x_1,x)] - \frac{1}{2t}\|x\|^2 + C_t, \qquad k_t(y,x) = e^{-\frac{\|ty-x\|^2}{2(1-t)^2}} \tag{10}$$

where C_t is a function that depends only on t. See subsection B.1 for proof.

Using the closed-form solution of $U_{\text{data}}(x,t)$ in Theorem 3.1, we numerically solve Equation 9 to compute the data-distribution component $\mathcal{T}_{\text{data}}(x_0)$ of the target $\mathcal{T}(x_0)$ in Equation 6. In particular, we consider three standard time-discretization schemes, namely Forward Euler (FE), Backward Euler (BE), and the Trapezoidal Rule (TR), and then take the limit $\Delta t \to 0$, which yields the update formulas and targets shown in Table 1. To avoid singular behavior during computation of the limit, we mollify the kernel k_t in Equation 10 as $k_t^\epsilon(y,x) = e^{-\frac{\|ty-x\|^2}{2(1-t)^2+\epsilon}}$. Notably, during numerical integration, the time step Δt cancels the factor $\frac{1}{1-t}$ in ∇U_{data}, making the limit well defined and computable. Detailed derivations are given in subsection B.2.

Table 1: Three standard discretizations of the data potential-driven dynamics in Equation 9.

Scheme	Update formula	Target
FE	$x_{n+1} = x_n - \Delta t\, \nabla U_{\text{data}}(x_n, t_n)$	$\mathcal{T}_{data}(x_0) = \frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^\theta_{0\to1}(x_0))k_1^\epsilon(x_1, \psi^\theta_{0\to1}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k_1^\epsilon(x_1, \psi^\theta_{0\to1}(x_0))]}$
BE	$x_{n+1} = x_n$ $- \Delta t \nabla U_{\text{data}}(x_{n+1}, t_{n+1})$	$\mathcal{T}_{data}(x_0) = \frac{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}]_\sharp p_0}[(\psi^\theta_{0\to1}(x_0) - \tilde{x}_1)k_1^\epsilon(\psi^\theta_{0\to1}(x_0), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}]_\sharp p_0}[k_1^\epsilon(\psi^\theta_{0\to1}(x_0), \tilde{x}_1)]}$
TR	$x_{n+1} = x_n - \frac{\Delta t}{2}(\nabla U_{\text{data}}(x_n, t_n)$ $+ \nabla U_{\text{data}}(x_{n+1}, t_{n+1}))$	$\mathcal{T}_{data}(x_0) = \frac{1}{2}\frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^\theta_{0\to1}(x_0))k_1^\epsilon(x_1, \psi^\theta_{0\to1}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k_1^\epsilon(x_1, \psi^\theta_{0\to1}(x_0))]}$ $+ \frac{1}{2}\frac{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}]_\sharp p_0}[(\psi^\theta_{0\to1}(x_0) - \tilde{x}_1)k_1^\epsilon(\psi^\theta_{0\to1}(x_0), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}]_\sharp p_0}[k_1^\epsilon(\psi^\theta_{0\to1}(x_0), \tilde{x}_1)]}$

Here, $k_1(y,x) = \lim_{t\to1} k_t^\epsilon(y,x) = e^{-\frac{\|y-x\|^2}{\epsilon}}$. In subsection 4.1, we compare different numerical schemes and their impact on training performance. The results indicate that the computation of the data term is highly sensitive to the order of discretization, with the second-order Trapezoidal Rule showing a significant advantage over the lower-order alternatives.

3.2 Numerical Calculation For Physics Term

Using U_{phys} from Equation 8, we derive the physics-driven dynamics $\frac{d}{dt}x_t = -\lambda\nabla U_{\text{phys}}(x,t)$ under three schemes: Forward Euler (FE), Backward Euler (BE), and Backward Euler with $\lambda \to \infty$ (BE-Lim). As in subsection 3.1, we integrate over the terminal interval and take $\Delta t \to 0$ to obtain the physics component $\mathcal{T}_{\text{phys}}(x_0)$ of $\mathcal{T}(x_0)$ in Equation 6. The resulting updates and targets are summarized in Table 2, with derivations in subsection B.3.

Table 2: Three standard discretizations of the Physics potential-driven dynamics.

Scheme	Update formula	Target
FE	$x_{n+1} = x_n - \Delta t\lambda\nabla U_{\text{phys}}(x_n, t_n)$	$\mathcal{T}_{phys}(x_0) = -\lambda J_R^\top(\psi^\theta_{0\to1}(x_0))R(\psi^\theta_{0\to1}(x_0))$
BE	$x_{n+1} = x_n - \Delta t\lambda\nabla U_{\text{phys}}(x_{n+1}, t_{n+1})$	$\mathcal{T}_{phys}(x_0) = -(J_R^\top(\psi^\theta_{0\to1}(x_0))J_R(\psi^\theta_{0\to1}(x_0)) + \gamma E)^{-1}$ $J_R^\top(\psi^\theta_{0\to1}(x_0))R(\psi^\theta_{0\to1}(x_0))$
BE-Lim	$x_{n+1} = x_n - \Delta t\lambda\nabla U_{\text{phys}}(x_{n+1}, t_{n+1})$, $\lambda \to \infty$	$\mathcal{T}_{phys}(x_0) = -J_R^\top(\psi^\theta_{0\to1}(x_0))(J_R(\psi^\theta_{0\to1}(x_0))$ $J_R^\top(\psi^\theta_{0\to1}(x_0)))^{-1}R(\psi^\theta_{0\to1}(x_0))$

J_R denotes the Jacobian of R, E denotes the identity matrix, and $\gamma = \frac{1}{2\lambda}$.

Unlike the data term, we consider only first-order schemes, which already perform well in subsection 4.1. We additionally include the $\lambda \to \infty$ Backward Euler limit due to its connection to classical position-based dynamics (subsubsection B.3.3). Formally, Forward Euler corresponds to direct gradient evaluation, whereas Backward Euler gives a Gauss-Newton-type gradient computation.

3.3 Algorithm and Analysis

Combining $\mathcal{T}_{\text{data}}(x_0)$ from subsection 3.1 and $\mathcal{T}_{\text{phys}}(x_0)$ from subsection 3.2, we obtain the final target for the short-flow dynamics,

$$\mathcal{T}(x_0) = \mathcal{T}_{\text{data}}(x_0) + \bar{\lambda}\,\mathcal{T}_{\text{phys}}(x_0), \quad (11)$$

where $\bar{\lambda}$ is a tunable hyperparameter introduced to unify different numerical schemes. This helps give the concrete loss form of Equation 6. Here, $\mathcal{T}_{\text{data}}$ can be chosen from the three forms in Table 1, while $\mathcal{T}_{\text{phys}}$ can be chosen from the three forms in Table 2, yielding different combinations. Our training algorithm based on this direct-target loss is shown in Algorithm 1. After training, $\psi^{\theta}_{0\to1}$ is encouraged to generate $\psi^{\theta}_{0\to1}(x_0)$ that match the target distribution and satisfy the physical constraints.

Algorithm 1 PI-LSFM Training

Require: dataset $\mathcal{D}$, batch size B, initial model parameters θ, learning rate η, hyperparameter $\bar{\lambda}$

repeat
  Sample $\mathcal{X}_1 = \{x_1^j\}_{j=1}^B$ from the dataset $\mathcal{D}$
  Sample noise $\mathcal{X}_0 = \{x_0^i\}_{i=1}^B$ from $\mathcal{N}(0, I)$
  $x_1^{i,\theta} \leftarrow \psi^{\theta}_{0\to1}(x_0^i)$ and $\mathcal{X}_1^{\theta} = \{x_1^{i,\theta}\}_{i=1}^B$
  Compute $\{\mathcal{T}_{\text{data}}(x_0^i)\}_{i=1}^B$ from $\mathcal{X}_1^{\theta}$ and $\mathcal{X}_1$ using tab. 1 and the expectation computation in eq.12
  Compute $\{\mathcal{T}_{\text{phys}}(x_0^i)\}_{i=1}^B$ from $\mathcal{X}_1^{\theta}$ using tab. 2
  $\mathcal{T}(x_0^i) \leftarrow \mathcal{T}_{data}(x_0^i) + \bar{\lambda}\mathcal{T}_{phys}(x_0^i)$
  $\mathcal{L}(\theta) \leftarrow \frac{1}{B}\sum_{i=1}^B \|x_1^{i,\theta} - sg(x_1^{i,\theta} + \mathcal{T}(x_0^i))\|^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta)$
until convergence

Expectation computation The formula of $\mathcal{T}_{\text{data}}(x_0)$ in Table 1 involves expectations. In practice, we approximate them using samples within the same batch in Algorithm 1. Specifically, for each $\mathcal{T}(x_0^i)$, the expectations $\mathbb{E}_{x_1\sim p_1}[F(x_1, x_0^i)]$ and $\mathbb{E}_{\bar{x}_1\sim[\psi^{\theta}_{0\to1}]_\sharp p_0}[F(\bar{x}_1, x_0^i)]$ are estimated using $\mathcal{X}_1$ and $\mathcal{X}_1^{\theta}$, respectively, where F is an arbitrary function:

$$\mathbb{E}_{x_1\sim p_1}[F(x_1, x_0^i)] = \frac{1}{B}\sum_{j=1}^{B} F(x_1^j, x_0^i), \quad \mathbb{E}_{\bar{x}_1\sim[\psi^{\theta}_{0\to1}]_\sharp p_0}[F(\bar{x}_1, x_0^i)] = \frac{1}{B}\sum_{j=1}^{B} F(x_1^{j,\theta}, x_0^i) \quad (12)$$

This computation is fully parallelizable; the PyTorch implementation is given in subsection C.3.

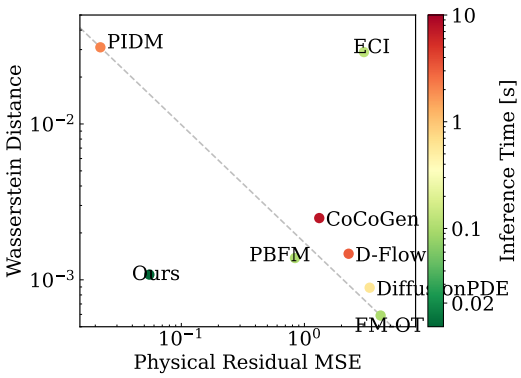
Conditional generation For conditional generation, the learned map becomes $\psi^{\theta}_{0\to1}(\cdot|c)$, and the corresponding expectations in Table 1 are taken under $p_1(\cdot|c)$ and $[\psi^{\theta}_{0\to1}(\cdot|c)]_\sharp p_0$ (see subsection C.1). Accordingly, Algorithm 1 only requires a different grouping strategy: we sample B_c conditions, sample B_s data samples for each condition, and sample $B = B_cB_s$ noise samples. The update in Algorithm 1 is then computed within each condition group and averaged over all B_c conditions. The detailed algorithm is given in Algorithm 2. This supports both label-conditioned generation (subsection 4.2) and generation conditioned on complex fields (subsection 4.3).

Necessity of the $\Delta t \to 0$ limit Taking the limit $\Delta t \to 0$ in Equation 6 is necessary for three reasons. First, it turns $\psi^{\theta}_{0\to1-\Delta t}$ into $\psi^{\theta}_{0\to1}$, so that the loss depends on a single unknown flow map and becomes self-consistent. Second, it makes the kernel $k_t^\epsilon(x, y)$ sufficiently sharp near $t = 1$, allowing accurate estimation of the $k_t^\epsilon(x, y)$-weighted expectation in Table 1 with a moderate batch size. Third, it applies $\mathcal{R}(\cdot)$ directly to $\psi^{\theta}_{0\to1}(x_0)$ rather than to the intermediate state $\psi^{\theta}_{0\to1-\Delta t}(x_0)$ for Table 2, avoiding the Jensen-type mismatch from intermediate residual enforcement.

Stabilization Computing the kernel $k_t^\epsilon(x, y)$ in high-dimensional $\mathcal{X}$ may suffer from the curse of dimensionality and destabilize optimization. Following Deng et al. (2026), we therefore perform the computation in feature spaces $\mathcal{Z} = \phi(X)$, where ϕ denotes an encoder. The resulting loss is

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0\sim p_0}[\|\phi(\psi^{\theta}_{0\to1}(x_0)) - sg(\phi(\psi^{\theta}_{0\to1}(x_0) + \bar{\lambda}\mathcal{T}_{phys}(x_0)) + \mathcal{T}^{\mathcal{Z}}_{data}(x_0))\|^2] \quad (13)$$

where $\mathcal{T}^{\phi}_{\text{data}}(\cdot)$ denotes the corresponding data term computed in the feature space $\mathcal{Z}$. In subsection 4.1, we compare different encoder choices, including several physics foundation models. The encoder is used only to stabilize the kernel-weighted estimation of the data guidance target in high-dimensional field spaces. Without the encoder, Darcy flow still achieves reasonable physical residuals and only slightly poorer distribution matching (RE=0.044, WD$\cdot10^2$ = 0.446, and JS$\cdot10^1$ = 0.889), but becomes more prone to training collapse, as illustrated in Figure 11. Figure 4b shows that different encoders yield competitive feature-space training performance, with further comparisons in subsection D.3. Thus, the feature encoder is a stabilization device rather than an inherent component of PI-LSFM. We use it only for the weakly conditioned Darcy and Kolmogorov flow benchmarks (Sections 4.1 and 4.2), while dynamic stall (Section 4.2) and topology optimization (Section 4.3) are trained directly without any encoder. Although MAE roughly doubles the per-step training time (0.271s vs. 0.132s without an encoder), PI-LSFM uses about one third of the training steps of Baldan et al. (2026), largely offsetting this overhead. Unlike latent-based generation, PI-LSFM still generates $\psi^{\theta}_{0\to1}(x_0)$ in the original space and computes only the loss in feature space; see subsection C.2 and Algorithm 3.



(a) Pareto front of physical residual MSE vs. Wasserstein distance for Darcy flow for the proposed methods.

(b) Feature extractors (100K training steps)

Extractor	Best $\bar{\lambda}$	RE	WD $\cdot 10^2$	JS $\cdot 10^1$	Sec/Step
MAE	3.0	0.117	0.213	0.216	0.271
Poseidon	1.0	0.410	0.238	0.271	0.834
DPOT	10.0	0.596	0.071	0.126	0.65
MPP	10.0	0.109	1.027	0.694	0.652

(c) Physics term methods (400K training steps)

Method	Best $\bar{\lambda}$	RE	WD $\cdot 10^2$	JS $\cdot 10^1$	Sec/Step
FE	3.0	0.055	0.108	0.171	0.271
BE	piecewise	0.436	0.122	0.159	0.388
BE-Lim.	piecewise	2.540	0.184	0.192	0.376

† "piecewise" denotes the $\bar{\lambda}$ scheduler described in subsection D.1.

(d) Data term methods (100K training steps)

Method	Best $\bar{\lambda}$	RE	WD $\cdot 10^2$	JS $\cdot 10^1$	Sec/Step
FE	3.0	8.125	4.244	5.846	0.268
BE	10.0	1.853	4.857	7.462	0.267
TR	3.0	0.117	0.213	0.216	0.271

(e) Comparison of different methods in terms of RE, WD, JS, and NFE. **Best** and second-best results are highlighted. Our model uses the same architecture and number of parameters as PIDM, training with 400K steps

Metric	Ours	sCM-PINN	PBFM	FM-OT	CoCoGen†	PIDM†	DiffusionPDE	D-Flow‡	ECI§
RE	0.055	0.107	0.838	4.159	1.320	**0.022**	3.388	2.286	3.045
WD $\cdot 10^2$	0.108	6.453	0.138	**0.059**	0.249	3.103	0.089	0.147	2.892
JS $\cdot 10^1$	0.171	7.401	0.256	**0.131**	0.360	3.179	0.139	0.237	2.818
NFE	**1**	**1**	20	20	100	100	20	20	20
IT [s]	**0.012**	0.025	0.101	0.100	7.395	2.050	0.590	3.126	0.122

† Our method and the reproduced sCM-PINN share the same U-Net architecture. By removing time-conditioning and modulation (since t is not required for our formulation), our method achieves faster inference.

Figure 4: Comparison of design choices and overall performance in our framework. (a) Pareto front of physical residual MSE vs. Wasserstein distance for Darcy flow. (b)–(d) compare different feature extractors, numerical discretizations for the physics term and the data term respectively. Training time per step (s) is also reported in these comparisons. More details are provided in Appendix D. (e) Overall comparison with competing methods on RE, WD, JS, NFE, and inference time.

4 Experiments

We choose the four benchmarks to progressively evaluate unconditional generation, conditional generalization, coupled multi-field physical constraints, and physics-constrained inverse design. We use the U-Net architecture of Bastek et al. (2024) without architecture-specific optimization, and match parameter counts whenever possible. For fair comparison, we report NFE and wall-clock inference time when available, using standard sampling settings or reported numbers for multi-step baselines. Since multi-step samplers benefit from iterative refinement, we target a favorable accuracy-efficiency trade-off rather than uniformly dominating every metric. All experiments use an NVIDIA A100 GPU. Our main benchmark results are obtained from single training runs with the fixed seed 42 and no seed selection. Appendix G.2, Table 17 reports a three-seed, 100K-step Darcy stability check.

4.1 Darcy Flow and Design Studies

Darcy flow is a two-dimensional steady-state porous-media benchmark on 64×64 pressure p and permeability K fields, with 10k training and 1k validation samples. Following Baldan et al. (2026), we enforce the Darcy residual $\mathcal{R}_{\text{pde}} = \nabla \cdot (K \nabla p) + f = 0$ together with the boundary-condition loss, and evaluate physical validity and distributional fidelity using residual error (RE), Wasserstein distance (WD), and Jensen–Shannon divergence (JS) of pressure field. This is an unconditional generation task, and lower values indicate better performance.

On Darcy flow, we compare with multi-step baselines PBFM Baldan et al. (2026), FM-OT Lipman et al. (2023), CoCoGen Jacobsen et al. (2025), PIDM Bastek et al. (2024), DiffusionPDE Huang et al. (2024), D-Flow Ben-Hamu et al. (2024), and ECI Cheng et al. (2024), using reported numbers from Baldan et al. (2026), as well as our one-step flow-map baselines built on sCM-PINN Chang et al. (2026). As shown in Figure 4e and Figure 4a, our method significantly outperforms other one-step methods, matches strong multi-step baselines, and lies on the best Pareto front, while using fewer

Table 3: Generative performance metrics for Kolmogorov flow and dynamic stall problems. RE: physical Residual MSE, WD: Wasserstein Distance, JS: Jensen-Shannon divergence, MMSE: MSE of the mean fields, SMSE: MSE of the standard deviation fields, NFE: Number of function evaluations, IT: inference wall-clock time. **Best** and second-best results are highlighted.

Dataset	Metric	Ours	PBFM	OT-FM	DiffusionPDE	D-Flow†	PCFM†
Kolmogorov Flow	RE $\cdot 10^1$	**0.404**	1.362	2.314	1.930	-	-
	WD $\cdot 10^1$	**0.140**	1.222	2.124	3.698	-	-
	JS $\cdot 10^2$	**1.621**	7.440	12.49	19.39	-	-
	MMSE $\cdot 10^4$	25.83	4.455	**4.188**	5.669	-	-
	SMSE $\cdot 10^4$	**1.765**	2.484	2.574	21.82	-	-
	NFE	**1**	20	20	20	20	20
	IT $[ms]$	**27.25**	98.97	98.75	267.8	6431	-
Dynamic Stall	RE $\cdot 10^6$	**0.00015**	0.339	11.02	12.20	11.32	0.143
	WD $\cdot 10^4$	**1.633**	1.814	2.707	2.509	3.484	4.013
	JS $\cdot 10^2$	**0.403**	0.680	0.983	1.029	1.014	1.206
	MMSE $\cdot 10^5$	5.218	**1.490**	2.791	2.626	2.507	5.669
	SMSE $\cdot 10^5$	4.894	**0.874**	1.458	1.236	1.372	7.674
	NFE	**1**	20	20	20	20	20
	IT $[ms]$	**21.53**	60.47	59.75	171.7	138.9	3906

† D-Flow and PCFM are excluded from Kolmogorov-flow evaluation due to unstable generation and prohibitive cost.

training steps with the same batch size and roughly matched parameter count to the PBFM baseline. At inference, its one-step sampling yields substantial acceleration, including about a $5\times$ speedup over architecture-optimized PBFM under a standard backbone. Additional experimental details, including the training procedure, evaluation metrics, and qualitative visualizations, are given in subsection E.2.

To stabilize training, we adopt the feature-space stabilization strategy described in subsection 3.3. We study different encoder choices, including a multi-resolution MAE trained on this dataset He et al. (2022), the pretrained multi-resolution U-Net-based physics foundation model Poseidon Herde et al. (2024), the transformer-based physics foundation model DPOT Hao et al. (2024), and MPP McCabe et al. (2024). The results show that MAE, Poseidon, and DPOT achieve comparable performance, with MAE providing the strongest overall results and the fastest training speed. We therefore use MAE as the encoder for Darcy and Kolmogorov flow (Sections 4.1 and 4.2). In Figure 4d and Figure 4c, we further compare different numerical schemes for the data term and the physics term, respectively, while fixing the other term to its best-performing choice. The results show that the second-order trapezoidal rule works best for the data term, while Forward Euler performs best for the physics term. A more complete design study and parameter study, including the $\bar{\lambda}$ schedule, is provided in Appendix D.

On Darcy, we compare PI-Consistency (sCM), PI-MeanFlow, and PI-Shortcut with the same terminal physics loss under matched settings. PI-LSFM improves RE, WD, and JS at NFE $= 1$ (Table 4). Together with the toy results (Figure 9, Table 11), these comparisons support the value of direct data and physics guidance for one-step training. Further evidence is provided by the conflict-free controls in Appendix G.1, which show that conflict-free updates alone do not close the performance gap.

Table 4: Additional one-step Darcy comparisons. All methods use NFE $= 1$; IT is inference time.

Method	RE ↓	WD$\cdot 10^2$ ↓	JS$\cdot 10^1$ ↓	IT [s] ↓
PI-Consistency	0.240	3.889	5.023	0.025
PI-MeanFlow	4603.573	4.243	5.457	0.024
PI-Shortcut	0.435	2.336	2.518	0.024
PI-LSFM	0.055	0.108	0.171	0.012

Table 5: Training time per generator-training step on Darcy.

Method	Seconds/step
PBFM	0.566
PI-Consistency	0.223
PI-MeanFlow	0.264
PI-Shortcut	0.328
PI-LSFM	0.308

Table 5 compares training time on an A100. PI-LSFM costs 0.308 s per generator-training step, including the full MAE pretraining cost amortized over training (0.037 s for MAE plus 0.271 s for the generator; 400K steps each), versus 0.566 s for PBFM. PI-LSFM achieves the reported Darcy results after 400K generator updates, about one third of the 1,248K training iterations reported for PBFM Baldan et al. (2026). The encoder adds no inference-time cost.

4.2 Kolmogorov Flow and Dynamic Stall

We evaluate conditional generation on two-dimensional incompressible Kolmogorov flow following Baldan et al. (2026). The model is conditioned on the Reynolds number and evaluated on 16 unseen Reynolds numbers, testing generalization across turbulent flow regimes. Physical consistency is enforced through the incompressibility constraint $\nabla \cdot U = 0$. As shown in Table 3, PI-LSFM achieves strong physical and distributional performance while requiring only a single function evalua-

tion, compared with 20-step baselines. We report physical residual error (RE), distributional metrics (WD and JS), conditional-statistics errors (MMSE and SMSE), and inference efficiency (NFE and IT). Full dataset, metric, and implementation details are provided in Appendix E.3.

We further evaluate PI-LSFM on conditional generation for dynamic stall, a compressible-flow benchmark around a sinusoidally pitching NACA0012 airfoil following Baldan et al. (2026). The model predicts six coupled aerodynamic and thermodynamic fields conditioned on four operating and motion parameters, and is evaluated on unseen configurations. Physical consistency is enforced through the ideal-gas relation and a temperature-dependent skin-friction constraint based on Sutherland's law. As shown in Table 3, PI-LSFM achieves competitive performance relative to multi-step baselines while requiring only one function evaluation. The evaluation follows the same distributional, statistical, and efficiency metrics as Kolmogorov flow. Full problem formulation, residual definitions, training details, and qualitative results are provided in Appendix E.4.

4.3 Topology Optimization

Topology optimization extends our evaluation from physics-constrained field generation to conditional inverse design. Following Bastek et al. (2024), the model conditions on loading and boundary conditions, auxiliary physical fields, and a target volume constraint, and jointly generates the material density and displacement fields. Physical consistency is evaluated through the FEM equilibrium residual, while structural quality and constraint satisfaction are measured by compliance error and volume-fraction error. We evaluate both in-distribution and out-of-distribution boundary conditions.

Table 6: Inference timing on the same hardware.

Method	NFE	IT [s]
Diffusion	100	1.720
PG-Diffusion	100	10.488
CoCoGen	100	7.911
TopoDiff-G	100	11.182
DOM	100	3.314
PIDM	100	1.701
PI-LSFM	1	0.013

As shown in Table 7, PI-LSFM achieves competitive inverse-design performance compared with multi-step diffusion-based baselines while requiring only a single function evaluation. Table 6 further shows that PI-LSFM generates a design in 0.013 s, compared with 1.701–11.182 s for the listed 100-step baselines. Full problem formulation, metric definitions, training details, and qualitative results are provided in Appendix E.5.

Table 7: Topology optimization results under both in-distribution and out-of-distribution boundary-condition settings. PI-LSFM attains competitive performance against 100-step diffusion-based baselines while using only a single function evaluation.

Model	Size	*in-distribution*			*out-of-distribution*		
		$\mathcal{R}_{\mathrm{MAE}}\downarrow$	MDN % CE $\downarrow$	% VFE $\downarrow$	$\mathcal{R}_{\mathrm{MAE}}\downarrow$	MDN % CE $\downarrow$	% VFE $\downarrow$
Diffusion	136M	1.86e−3	-0.20	2.93	1.97e−3	0.30	2.80
PG-Diffusion (Shu et al. (2023))	136M	1.82e−3	0.09	3.59	1.92e−3	0.81	3.23
CoCoGen (Jacobsen et al. (2025))	136M	1.51e−3	0.14	4.00	1.56e−3	0.58	3.64
TopoDiff-G (Mazé & Ahmed (2023))	239M	–	0.83	1.49	–	1.82	1.80
DOM Giannone et al. (2023)	121M	–	0.74	1.52	–	3.47	1.59
PIDM Bastek et al. (2024)	136M	1.24e−3	0.06	2.25	1.29e−3	0.56	1.91
PI-LSFM (ours)	136M	1.66e−3	-0.00	1.53	1.95e−3	1.86	1.66

"–" indicates that $\mathcal{R}_{\mathrm{MAE}}$ is not applicable because the method does not provide predicted solution fields.

5 Conclusion

We introduced Long-Short Flow Maps with Direct Guidance (PI-LSFM), a one-step physics-informed generative framework that decomposes the full flow map into a long-range map and a near-terminal short map. This decomposition converts indirect flow-map and physics-informed supervision into a directly computable near-terminal guidance target, enabling joint data and physics guidance. We studied numerical instantiations of the short-flow dynamics and their effects on stability and generation quality. Experiments on unconditional generation, conditional generation, and physics-constrained inverse design show that PI-LSFM achieves a favorable trade-off among stability, distributional fidelity, physical consistency, and inference efficiency, matching strong multi-step baselines while preserving one-step efficiency without distillation. Our framework has several limitations. Similar to prior physics-informed generative methods Baldan et al. (2026); Bastek et al. (2024), PI-LSFM enforces physical constraints only during training, leaving inference-time constraint enforcement to future work. It would be interesting to combine PI-LSFM with inference-time constraint methods Cheng et al. (2024); Chang et al. (2026). We study only a limited set of numerical schemes for the data and physics terms; broader numerical families may further improve stability and performance. Scaling to larger models, datasets, and more complex physics tasks is another important direction.

AI USE STATEMENT

We used generative AI tools primarily to collect, organize, and transcribe experimental information for the appendix, and to polish the writing of the main text. We also used these tools for literature retrieval and discovery to identify relevant related work. For mathematical work, AI assisted with developing the counterexample construction for Theorem 3.1 and writing the proof of Theorem 3.1. The authors take responsibility for the final content of this paper, including the AI-assisted text, mathematical claims, and proofs.

REFERENCES

ANSYS. ANSYS Fluent 2024 R2. `https://www.ansys.com/products/fluids/ansys-fluent`, 2024.

Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Physics vs distributions: Pareto optimal flow matching with physics constraints. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jan-Hendrik Bastek, WaiChing Sun, and Dennis M Kochmann. Physics-informed diffusion models. *arXiv preprint arXiv:2403.14404*, 2024.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Quentin Bertrand, Anne Gagneux, Mathurin Massias, and Rémi Emonet. On the closed-form of flow matching: Generalization does not arise from target stochasticity. *arXiv preprint arXiv:2506.03719*, 2025.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research (TMLR)*, 2025.

Che-Chia Chang, Chen-Yang Dai, Te-Sheng Lin, Ming-Chih Lai, and Chieh-Hsin Lai. Stabilizing physics-informed consistency models via structure-preserving training. *arXiv preprint arXiv:2602.09303*, 2026.

Chaoran Cheng, Boran Han, Danielle C Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W Mahoney, and Yuyang Wang. Gradient-free generation for hard-constrained systems. *arXiv preprint arXiv:2412.01786*, 2024.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Giorgio Giannone, Akash Srivastava, Ole Winther, and Faez Ahmed. Aligning optimization trajectories with diffusion models for constrained design generation. *Advances in neural information processing systems*, 36:51830–51861, 2023.

Ralf Gommers, Pauli Virtanen, Matt Haberland, Evgeni Burovski, Tyler Reddy, Warren Weckesser, Travis E Oliphant, Andrew Nelson, David Cournapeau, Pamphile Roy, et al. scipy/scipy: Scipy 1.16. 0. *Zenodo*, 2025.

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, et al. Splitmeanflow: Interval splitting consistency in few-step generative modeling. *arXiv preprint arXiv:2507.16884*, 2025.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation. *Advances in Neural Information Processing Systems*, 37: 130291–130323, 2024.

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

Zhiqi Li and Bo Zhu. A long-short flow-map perspective for drifting models. *arXiv preprint arXiv:2602.20463*, 2026.

Zhiqi Li, Yuchen Sun, Greg Turk, and Bo Zhu. Functional mean flow in hilbert space. *arXiv preprint arXiv:2511.12898*, 2025.

Zhiqi Li, Yuchen Sun, Duowen Chen, Jinjin He, and Bo Zhu. Trajectory consistency for one-step generation on euler mean flows. *arXiv preprint arXiv:2602.02571*, 2026.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 2002.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Yuxuan Liu, Jingmin Sun, Xinjie He, Griffin Pinney, Zecheng Zhang, and Hayden Schaeffer. Prose-fd: A multimodal pde foundation model for learning multiple operators for forecasting fluid dynamics. *arXiv preprint arXiv:2409.09811*, 2024.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

François Mazé and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 9108–9116, 2023.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

Mahindra Singh Rautela, Alexander Most, Siddharth Mansingh, Bradley C Love, Alexander Scheinker, Diane Oyen, Nathan Debardeleben, Earl Lawrence, and Ayan Biswas. Morph: Pde foundation models with arbitrary data modality. *arXiv preprint arXiv:2509.21670*, 2025.

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.

Abu Bucker Siddik, Diane Oyen, Alexander Most, Michal Kucer, and Ayan Biswas. Spus: A lightweight and parameter-efficient foundation model for pdes. *arXiv preprint arXiv:2510.01370*, 2025.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gomez-Bombarelli, and Christopher Vincent Rackauckas. Physics-constrained flow matching: Sampling generative models with hard constraints. *arXiv preprint arXiv:2506.04171*, 2025.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Sifan Wang, Zehao Dou, Siming Shan, Tong-Rui Liu, and Lu Lu. Fundiff: Diffusion models over function spaces for physics-informed generative modeling. *arXiv preprint arXiv:2506.07902*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. *arXiv preprint arXiv:2510.20771*, 2025.

Yi Zhang and Difan Zou. Physics-informed distillation of diffusion models for pde-constrained generation. *arXiv preprint arXiv:2505.22391*, 2025.

APPENDIX

Appendix **13**

A Related Work **13**

B Missing Proofs **15**

B.1 Proof of Theorem 3.1 15

B.2 Derivation of Table 1 16

B.3 Derivation of Table 2 18

C Additional Theoretical Results and Algorithms **19**

C.1 Conditional Generation 19

C.2 Stabilization via Feature Spaces 21

C.3 Pytorch Code 23

D Design Study and Additional Results **24**

D.1 Physics Term Numerical Integration Study 24

D.2 Data Term Numerical Integration Study 25

D.3 Encoder Study 26

E Experiment Details and Additional Results **27**

E.1 Toy Example 27

E.2 Darcy Flow 28

E.3 Kolmogorov Flow 32

E.4 Dynamic Stall 35

E.5 Topology Optimization 37

F Additional Theoretical Background **40**

F.1 Flow Map Learning 40

F.2 Closed-Form Solution 41

G Additional Darcy Comparisons and Experimental Settings **42**

G.1 Conflict-Free One-Step Comparisons 42

G.2 Three-Seed Darcy Check 42

G.3 Batch-Size Sensitivity 42

G.4 Default Settings and Selection Protocol 43

A RELATED WORK

Physics-informed generation. Physics-informed generative methods can be broadly divided into training-time and inference-time approaches. Training-time methods incorporate physical constraints directly into the learning objective. For example, physics-informed diffusion models Bastek et al.

(2024); Shu et al. (2023) add PDE residuals or boundary losses during training, while CoCoGen Jacobsen et al. (2025) and DiffusionPDE Huang et al. (2024) study physically consistent generation and PDE-conditioned modeling. FunDiff Wang et al. (2025) instead formulates physics-informed generative modeling over function spaces. Inference-time methods guide or correct samples during generation. D-Flow Ben-Hamu et al. (2024) differentiates through the generative flow for controlled generation, while hard-constrained and gradient-free methods Utkarsh et al. (2025); Cheng et al. (2024) enforce constraints during sampling without fully retraining the generator. Physics-informed distillation (PIDDM) Zhang & Zou (2025) follows a teacher-based route, distilling a pretrained diffusion model into a one-step generator for PDE-constrained generation. PI-LSFM instead trains a one-step generator from scratch using joint data and physics guidance, without a pretrained generative teacher. Most closely related to our work, Physics vs Distributions Baldan et al. (2026) highlights the trade-off between distribution matching and physical validity, and proposes a Pareto-optimal flow-matching framework with physics constraints. Physics-informed consistency models Chang et al. (2026), building on continuous-time consistency models Lu & Song (2024), aim to stabilize one-step physics-informed generation through structure-preserving training. In contrast, our long-short flow-map direct guidance provides a direct one-step training objective that better balances distributional fidelity and physical validity, without requiring iterative inference-time correction.

One-step generation. Early one-step generative models were mainly developed through distillation or adversarial training. For example, distribution matching distillation Yin et al. (2024) compresses multi-step diffusion sampling into a single-step generator, while GAN-based methods Heusel et al. (2017) directly learn one-step mappings with adversarial objectives. More recently, increasing attention has been given to training one-step generators from scratch without relying on a pretrained teacher. A prominent direction formulates one-step generation as learning a flow map from noise to data Boffi et al. (2025). Representative methods include consistency models Song et al. (2023); Lu & Song (2024); Geng et al. (2024), shortcut models Frans et al. (2025), mean-flow methods Geng et al. (2025), and extensions such as functional mean flow Li et al. (2025). More recently, drifting models Deng et al. (2026) revisit one-step generation from a distribution-matching perspective, improving the stability and scalability of training-from-scratch one-step generators and concurrent work Li & Zhu (2026) provides a long-short flow-map interpretation of drifting models. Our work instead studies physics-informed one-step generation, where the central challenge is to balance data-distribution fidelity and physical constraint satisfaction within a single flow map. We analyze this interaction through a potential-based formulation, which avoids the terminal-time singularity of velocity-based formulations at $t = 1$, and propose a long-short flow-map framework with direct guidance for joint control of distributional and physical objectives.

PI-LSFM starts from a long-short flow-map decomposition of physics-informed one-step generation, providing a unified framework for analyzing and constructing direct guidance from both the data distribution and physical constraints. The near-terminal short flow is jointly driven by data and physics potentials, while different numerical discretizations induce explicit, implicit, and projection-based update rules within the same construction. This framework also clarifies its connection to drifting models Deng et al. (2026): under a particular raw-space specialization, the data-side correction recovers a drifting-style update. PI-LSFM extends beyond this special case by jointly deriving physics guidance, systematically studying alternative data and physics discretizations, and supporting feature-space guidance for high-dimensional physical fields. The matched one-step comparisons and numerical studies evaluate these components and their interaction in physics-informed generation.

Physics foundation models. Self-supervised learning has recently enabled foundation models for scientific systems. Inspired by vision methods such as masked autoencoders (MAE) He et al. (2022), PDE foundation models aim to learn transferable spatiotemporal representations across equations, parameters, and conditions. Representative examples include Poseidon Herde et al. (2024), which uses multiscale operator learning for fluid-dynamics pretraining; DPOT Hao et al. (2024), which learns autoregressive denoising operators with a transformer backbone; and MPP McCabe et al. (2024), which pretrains surrogate models across multiple physical systems. PROSE-FD Liu et al. (2024) incorporates multimodal PDE descriptions and numerical fields, while MORPH Rautela et al. (2025) and SPUS Siddik et al. (2025) further address heterogeneous modalities, resolutions, and parameter efficiency. Unlike these works, we use physics foundation models only as feature encoders to stabilize our kernel-based training objective, while generation remains in the original physical field space.

B MISSING PROOFS

B.1 PROOF OF THEOREM 3.1

Theorem 3.1 (Potential-Driven System for Flow Matching) Assume $p_0 = \mathcal{N}(0, I)$, independent of the data endpoint x_1, and that p_1 is a smooth distribution with $p_1(x) > 0$. The Flow Matching dynamics with velocity u_t, learned in Equation 2, can be interpreted as a dynamical system that starts from an initial state $x_0 \sim p_0$ and evolves under a closed-form potential.

$$\frac{d}{dt}x_t = -\nabla U_{\text{data}}(x, t), \tag{14}$$

where $U_{\text{data}}(x, t)$ is defined as

$$U_{\text{data}}(x, t) = -\frac{1-t}{t}\log \mathbb{E}_{x_1\sim p_1}[k_t(x_1, x)] - \frac{1}{2t}\|x\|^2 + C_t, \qquad k_t(y, x) = e^{-\frac{\|ty-x\|^2}{2(1-t)^2}} \tag{15}$$

where C_t is a function that depends only on t.

Proof. Since Flow Matching uses the conditional path $X_t = tX_1 + (1-t)X_0$, Tweedie's formula gives $\mathbb{E}_{x_1\sim p_{1|t}}[x_1] = \mathbb{E}[x_1|x_t = x] = \frac{1}{t}\left(x + (1-t)^2\nabla \log(cp_t(x))\right)$,where c is an arbitrary constant. Moreover, by the property of the regression loss, the optimal velocity corresponding to Equation 2 can be written as $u_t^*(x) = \mathbb{E}_{p_{1|t}(x)}[\frac{x_1-x}{1-t}] = \frac{\mathbb{E}_{p_{1|t}(x)}[x_1]-x}{1-t}$. Therefore, we have

$$\begin{aligned} u_t^*(x) &= \frac{\mathbb{E}_{p_{1|t}(x)}[x_1] - x}{1-t} \\ &= \frac{\frac{1}{t}\left(x + (1-t)^2\nabla \log(cp_t(x))\right) - x}{1-t} \\ &= \frac{x}{t} + \frac{1-t}{t}\nabla \log cp_t(x) \\ &= -\nabla(-\frac{1}{2t}\|x\|^2 - \frac{1-t}{t}\log cp_t(x)). \end{aligned} \tag{16}$$

Hence, the Flow Matching dynamics can be driven by the potential $U_{\text{data}}(x, t) = -\frac{1}{2t}\|x\|^2 - \frac{1-t}{t}\log cp_t(x)$. Moreover, $p_t(x)$ admits a closed-form expression that can be computed directly from the data distribution:

$$\begin{aligned} p_t(x) &= \int\int p_{t|0,1}(x|x_1, x_0)p_1(x_1)p_0(x_0)dx_0dx_1 \\ &= \int\int \delta(x_t - (tx_1 + (1-t)x_0)p_1(x_1)p_0(x_0)dx_0dx_1 \\ &= \int p_1(x_1)p_0(\frac{x_t - tx_1}{1-t})dx_1 \\ &= \int p_1(x_1)\frac{1}{(2\pi)^{\frac{d}{2}}(1-t)^d}e^{-\frac{\|x-tx_1\|^2}{2(1-t)^2}}dx_1 \\ &= \frac{1}{c}\mathbb{E}_{x_1\sim p_1}[k_t(x_1, x)] \end{aligned} \tag{17}$$

Here, we choose c as$(2\pi)^{\frac{d}{2}}(1-t)^d$. Then U_{data} have the formula:

$$\begin{aligned} U_{\text{data}}(x, t) &= -\frac{1}{2t}\|x\|^2 - \frac{1-t}{t}\log cp_t(x) \\ &= -\frac{1}{2t}\|x\|^2 - \frac{1-t}{t}\log \mathbb{E}_{x_1\sim p_1}[k_t(x_1, x)] \end{aligned} \tag{18}$$

□

B.2 Derivation of Table 1

B.2.1 Forward Euler

We first derive the formula for Forward Euler. Applying Forward Euler to numerically integrate Equation 10 over $[1-\Delta t, 1]$ gives:

$$\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] = -\Delta t \nabla U_{data}(\psi^{\theta}_{0\to 1-\Delta t}(x_0), 1-\Delta t) \tag{19}$$

Substituting the expression of U_{data} from Theorem 3.1 yields:

$$\begin{aligned}
&\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] \\
&= -\Delta t \nabla[-\frac{1}{2(1-\Delta t)}\|x\|^2 - \frac{\Delta t}{1-\Delta t}\log \mathbb{E}_{x_1\sim p_1}[k_{1-\Delta t}(x_1, x)]]|_{x=\psi^{\theta}_{0\to 1-\Delta t}(x_0)} \\
&\overset{①}{=} \Delta t[\frac{x}{1-\Delta t} + \frac{\Delta t}{1-\Delta t}\frac{\mathbb{E}_{x_1\sim p_1}[\frac{(1-\Delta t)x_1 - x}{\Delta t^2}k_{1-\Delta t}(x_1, x)]}{\mathbb{E}_{x_1\sim p_1}[k_{1-\Delta t}(x_1, x)]}]|_{x=\psi^{\theta}_{0\to 1-\Delta t}(x_0)} \\
&= \Delta t[\frac{\mathbb{E}_{x_1\sim p_1}[\frac{x_1 - x}{\Delta t}k_{1-\Delta t}(x_1, x)]}{\mathbb{E}_{x_1\sim p_1}[k_{1-\Delta t}(x_1, x)]}]|_{x=\psi^{\theta}_{0\to 1-\Delta t}(x_0)} \\
&= \frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^{\theta}_{0\to 1-\Delta t}(x_0))k_{1-\Delta t}(x_1, \psi^{\theta}_{0\to 1-\Delta t}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k_{1-\Delta t}(x_1, \psi^{\theta}_{0\to 1-\Delta t}(x_0))]}
\end{aligned} \tag{20}$$

where ① holds because of $\nabla_x k_t(y, x) = \frac{ty - x}{(1-t)^2}k_t(y, x)$. After mollifying the above k_t into k_t^ϵ and taking the limit, we obtain:

$$\begin{aligned}
\mathcal{T}_{data}(x_0) &= \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] \\
&= \frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^{\theta}_{0\to 1}(x_0))k^{\epsilon}_1(x_1, \psi^{\theta}_{0\to 1}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k^{\epsilon}_1(x_1, \psi^{\theta}_{0\to 1}(x_0))]}
\end{aligned} \tag{21}$$

B.2.2 Backward Euler

We next consider the computation for Backward Euler. For Backward Euler, we need to use the potential at the endpoint ($t = 1$). By Theorem 3.1, we have $U_{\text{data}}(x, 1) = -\frac{1}{2}|x|^2$, which gives the direct endpoint velocity $u_1(x) = x$. Although this velocity is directly available, we seek a form that is symmetric to the Forward Euler case and preserves the same kernel-based computation. This symmetry allows both the forward and backward endpoint velocities to be expressed using the same type of kernel-weighted estimator. It therefore avoids the inconsistency in the trapezoidal rule (subsubsection B.2.3) that would arise from using an analytic velocity at one endpoint and a kernel-estimated velocity at the other endpoint, while ensuring that numerical errors are reflected in the same manner at both endpoints. To this end, we use the following lemma.

Lemma 1. *Let $\tilde{p}_t = (\psi^{\theta}_{0\to t})_{\#}p_0$. For $t \in (0, 1)$, the following equation holds:*

$$\frac{\mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[\frac{x - \tilde{x}_t}{1-t}k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[k_t(x, \tilde{x}_t)]} + \frac{1-t}{t}\nabla_x \log \mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[k_t(x, \tilde{x}_t)] = x. \tag{22}$$

Moreover, assume that $\psi^{\theta}_{0\to t}$ is a well-conditioned C^1-diffeomorphism. Then, on any compact region where $p_0((\psi^{\theta}_{0\to t})^{-1}(z)) \geq m_0 > 0$, $\nabla_x \log \mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[k_t(x, \tilde{x}_t)]$ is bounded and satisfies $\|\nabla_x \log \mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[k_t(x, \tilde{x}_t)]\| \leq \frac{e^{-1/2}LC + J}{(2\pi)^{d/2}m_0 c}$, and consequently

$$-\nabla U_{data}(x, 1) = \frac{\mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[\frac{x - \tilde{x}_t}{1-t}k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t\sim\tilde{p}_t}[k_t(x, \tilde{x}_t)]} + O(1-t) \tag{23}$$

Here, the constants $L, c, C, J > 0$ are defined by the bounds $\|\nabla_z(\psi^{\theta}_{0\to t})^{-1}(z)\| \leq L$, $0 < c \leq |\det \nabla_z(\psi^{\theta}_{0\to t})^{-1}(z)| \leq C$, and $\|\nabla_z|\det \nabla_z(\psi^{\theta}_{0\to t})^{-1}(z)|\| \leq J$.

Proof. Since $\nabla_x k_t(x, \tilde{x}_t) = -\frac{t(tx-\tilde{x}_t)}{(1-t)^2} k_t(x, \tilde{x}_t)$, we have

$$\nabla_x \log \mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)] = -\frac{t}{(1-t)^2} \frac{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[(tx - \tilde{x}_t) k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]}. \tag{24}$$

Therefore,

$$\begin{aligned} &\frac{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[\frac{x-\tilde{x}_t}{1-t} k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]} + \frac{1-t}{t} \nabla_x \log \mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)] \\ &= \frac{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[\frac{x-\tilde{x}_t}{1-t} k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]} - \frac{1}{1-t} \frac{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[(tx - \tilde{x}_t) k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]} \\ &= x \end{aligned} \tag{25}$$

This proves the identity.

Next, since $\tilde{p}_t = (\psi^\theta_{0\to t})_\# p_0$, the change-of-variables formula gives $\tilde{p}_t(z) = p_0((\psi^\theta_{0\to t})^{-1}(z)) |\det \nabla_z (\psi^\theta_{0\to t})^{-1}(z)|$. Hence,

$$\tilde{p}_t(z) \geq m_0 c \tag{26}$$

Moreover, since $\nabla_z \tilde{p}_t(z) = \nabla p_0((\psi^\theta_{0\to t})^{-1}(z)) \nabla_z (\psi^\theta_{0\to t})^{-1}(z) |\det \nabla_z (\psi^\theta_{0\to t})^{-1}(z)| + p_0((\psi^\theta_{0\to t})^{-1}(z)) \nabla_z |\det \nabla_z (\psi^\theta_{0\to t})^{-1}(z)|$, taking norms and using $\|\nabla p_0(z)\| = \|z\| p_0(z) = \frac{\|z\|}{(2\pi)^{d/2}} e^{-\frac{\|z\|^2}{2}} \leq \frac{e^{-1/2}}{(2\pi)^{d/2}}$, $\|\nabla_z (\psi^\theta_{0\to t})^{-1}\| \leq L$, $|\det \nabla_z (\psi^\theta_{0\to t})^{-1}| \leq C$, $p_0(z) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\|z\|^2}{2}} \leq \frac{1}{(2\pi)^{d/2}}$, and $\|\nabla_z |\det \nabla_z (\psi^\theta_{0\to t})^{-1}|\| \leq J$, we obtain

$$\|\nabla_z \tilde{p}_t(z)\| \leq \frac{e^{-1/2} LC + J}{(2\pi)^{d/2}}. \tag{27}$$

Since $\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)] = \int \exp(-\frac{\|tx-z\|^2}{2(1-t)^2}) \tilde{p}_t(z)\, dz$, this quantity is a Gaussian smoothing of $\tilde{p}_t$ evaluated at tx. Gaussian smoothing preserves the lower bound and the gradient bound. Therefore,

$$\|\nabla_x \log \mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]\| \leq \frac{e^{-1/2} LC + J}{(2\pi)^{d/2} m_0 c} \tag{28}$$

It follows that

$$\| - \frac{1-t}{t} \nabla_x \log \mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]\| \leq \frac{1-t}{t} \frac{e^{-1/2} LC + J}{(2\pi)^{d/2} m_0 c} = O(1-t). \tag{29}$$

Therefore,

$$-\nabla U_{data}(x, 1) = x = \frac{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[\frac{x-\tilde{x}_t}{1-t} k_t(x, \tilde{x}_t)]}{\mathbb{E}_{\tilde{x}_t \sim \tilde{p}_t}[k_t(x, \tilde{x}_t)]} + O(1-t). \tag{30}$$

This completes the proof. □

Using the above lemma, we have

$$\begin{aligned} &\mathcal{I}[\frac{d}{dt} \psi_{1-\Delta t \to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\ &= \Delta t \frac{\mathbb{E}_{\tilde{x}_{1-\Delta t} \sim \tilde{p}_{1-\Delta t}}[\frac{x-\tilde{x}_{1-\Delta t}}{\Delta t} k_{1-\Delta t}(x, \tilde{x}_{1-\Delta t})]}{\mathbb{E}_{\tilde{x}_{1-\Delta t} \sim \tilde{p}_{1-\Delta t}}[k_{1-\Delta t}(x, \tilde{x}_{1-\Delta t})]}|_{\psi^\theta_{0\to 1}(x_0)} + O(\Delta t^2) \\ &= \frac{\mathbb{E}_{\tilde{x}_{1-\Delta t} \sim \tilde{p}_{1-\Delta t}}[(\psi^\theta_{0\to 1}(x_0) - \tilde{x}_{1-\Delta t}) k_{1-\Delta t}(\psi^\theta_{0\to 1}(x_0), \tilde{x}_{1-\Delta t})]}{\mathbb{E}_{\tilde{x}_{1-\Delta t} \sim \tilde{p}_{1-\Delta t}}[k_{1-\Delta t}(\psi^\theta_{0\to 1}(x_0), \tilde{x}_{1-\Delta t})]} + O(\Delta t^2) \end{aligned} \tag{31}$$

After mollifying the above k_t into k_t^ϵ and taking the limit, we obtain:

$$\begin{aligned} \mathcal{T}_{data}(x_0) &= \lim_{\Delta t \to 0} \mathcal{I}[\frac{d}{dt} \psi_{1-\Delta t \to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\ &= \frac{\mathbb{E}_{\tilde{x}_1 \sim \tilde{p}_1}[(\psi^\theta_{0\to 1}(x_0) - \tilde{x}_1) k_1^\epsilon(\psi^\theta_{0\to 1}(x_0), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1 \sim \tilde{p}_1}[k_1^\epsilon(\psi^\theta_{0\to 1}(x_0), \tilde{x}_1)]} \end{aligned} \tag{32}$$

B.2.3 TRAPEZOIDAL RULE

Based on the calculation of the Forward Euler and Backward Euler cases, for the Trapezoidal Rule we have:

$$\begin{aligned}
&\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] \\
&= -\Delta t \frac{1}{2}[\nabla U_{data}(\psi^{\theta}_{0\to 1-\Delta t}(x_0), 1-\Delta t) + \nabla U_{data}(\psi^{\theta}_{0\to 1}(x_0), 1)] \\
&= \frac{1}{2}\frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^{\theta}_{0\to 1-\Delta t}(x_0))k_{1-\Delta t}(x_1, \psi^{\theta}_{0\to 1-\Delta t}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k_{1-\Delta t}(x_1, \psi^{\theta}_{0\to 1-\Delta t}(x_0))]} + \\
&\frac{1}{2}\frac{\mathbb{E}_{\tilde{x}_{1-\Delta t}\sim \tilde{p}_{1-\Delta t}}[(\psi^{\theta}_{0\to 1}(x_0) - \tilde{x}_{1-\Delta t})k_{1-\Delta t}(\psi^{\theta}_{0\to 1}(x_0), \tilde{x}_{1-\Delta t})]}{\mathbb{E}_{\tilde{x}_{1-\Delta t}\sim \tilde{p}_{1-\Delta t}}[k_{1-\Delta t}(\psi^{\theta}_{0\to 1}(x_0), \tilde{x}_{1-\Delta t})]} + O(\Delta t^2)
\end{aligned} \tag{33}$$

After mollifying the above k_t into k_t^ϵ and taking the limit, we obtain:

$$\begin{aligned}
\mathcal{T}_{data}(x_0) &= \lim_{\Delta t\to 0} \mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] \\
&= \frac{1}{2}\frac{\mathbb{E}_{x_1\sim p_1}[(x_1 - \psi^{\theta}_{0\to 1}(x_0))k^{\epsilon}_1(x_1, \psi^{\theta}_{0\to 1}(x_0))]}{\mathbb{E}_{x_1\sim p_1}[k^{\epsilon}_1(x_1, \psi^{\theta}_{0\to 1}(x_0))]} + \frac{1}{2}\frac{\mathbb{E}_{\tilde{x}_1\sim \tilde{p}_1}[(\psi^{\theta}_{0\to 1}(x_0) - \tilde{x}_1)k^{\epsilon}_1(\psi^{\theta}_{0\to 1}(x_0), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1\sim \tilde{p}_1}[k^{\epsilon}_1(\psi^{\theta}_{0\to 1}(x_0), \tilde{x}_1)]}
\end{aligned} \tag{34}$$

B.3 DERIVATION OF TABLE 2

B.3.1 FORWARD EULER METHOD

Consider the physics-potential-driven dynamical system $\frac{dx_\tau}{d\tau} = -\lambda\nabla U_{\text{phys}}(x_\tau, \tau)$, where $U_{\text{phys}}(x,\tau) = \frac{1}{1-\tau}\|R(x)\|^2$ and $\tau \in [t, 1]$. Applying Forward Euler to numerically integrate this system over $[1-\Delta t, 1]$ gives:

$$\begin{aligned}
\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] &= -\Delta t\lambda\nabla U_{phys}(\psi^{\theta}_{0\to 1-\Delta t}(x_0), 1-\Delta t) \\
&= -\Delta t\lambda\frac{1}{\Delta t}\nabla\|\mathcal{R}(x)\|^2|_{x=\psi^{\theta}_{0\to 1-\Delta t}(x_0)} \\
&= -\lambda J^{\top}_{\mathcal{R}}(\psi^{\theta}_{0\to 1-\Delta t}(x_0))\mathcal{R}(\psi^{\theta}_{0\to 1-\Delta t}(x_0))
\end{aligned} \tag{35}$$

Taking the limit $\Delta t \to 0$, we obtain:

$$\begin{aligned}
\mathcal{T}_{phys}(x_0) &= \lim_{\Delta t\to 0} \mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^{\theta}_{0\to 1-\Delta t}(x_0)] \\
&= -\lambda J^{\top}_{\mathcal{R}}(\psi^{\theta}_{0\to 1}(x_0))\mathcal{R}(\psi^{\theta}_{0\to 1}(x_0))
\end{aligned} \tag{36}$$

B.3.2 BACKWARD EULER METHOD

Applying Backward Euler to numerically integrate $\frac{dx_\tau}{d\tau} = -\lambda\nabla U_{\text{phys}}(x_\tau, \tau)$ over $[1-\Delta t, 1]$ requires evaluating the gradient at the next state $x_1 = x_{1-\Delta t} - \Delta t\lambda\nabla U_{\text{phys}}(x_1, 1-\Delta t)$. This is equivalent to the first-order optimality condition of the proximal problem

$$\begin{aligned}
x_1 &= \arg\min_x \frac{1}{2}\|x - x_{1-\Delta t}\|^2 + \Delta t\lambda U_{\text{phys}}(x, 1-\Delta t) \\
&= \arg\min_x \frac{1}{2}\|x - x_{1-\Delta t}\|^2 + \lambda\|R(x)\|^2.
\end{aligned} \tag{37}$$

Indeed, this follows by setting the derivative of the above objective to zero, which recovers $x_1 = x_{1-\Delta t} - \Delta t\lambda\nabla U_{\text{phys}}(x_1, 1-\Delta t)$. Letting $\Delta x = x - x_{1-\Delta t}$, and linearizing the residual around x_t as $R(x_{1-\Delta t} + \Delta x) \approx R(x_{1-\Delta t}) + J_R(x_{1-\Delta t})\Delta x$, the above problem becomes

$$\min_{\Delta x} \frac{1}{2}\|\Delta x\|^2 + \lambda\|R(x_{1-\Delta t}) + J_R(x_{1-\Delta t})\Delta x\|^2 \tag{38}$$

Define $f(\Delta x) = \frac{1}{2}\|\Delta x\|^2 + \lambda\|R(x_{1-\Delta t}) + J_R(x_{1-\Delta t})\Delta x\|^2$. Its gradient is $\nabla_{\Delta x} f(\Delta x) = \Delta x + 2\lambda J_R^\top(x_{1-\Delta t})(R(x_{1-\Delta t}) + J_R(x_{1-\Delta t})\Delta x)$. To solve the optimization problem, setting $\nabla_{\Delta x} f(\Delta x) = 0$ yields $\Delta x = -(J_R^\top(x_{1-\Delta t})J_R(x_{1-\Delta t}) + \frac{1}{2\lambda}E)^{-1}J_R^\top(x_{1-\Delta t})R(x_{1-\Delta t})$. Therefore,

$$\begin{aligned}
&\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\
&= -(J_R^\top(x)J_R(x) + \frac{1}{2\lambda}E)^{-1}J_R^\top(x)R(x)|_{x=\psi^\theta_{0\to 1-\Delta t}(x_0)} \\
&= -(J_R^\top(\psi^\theta_{0\to 1-\Delta t}(x_0))J_R(\psi^\theta_{0\to 1-\Delta t}(x_0)) + \frac{1}{2\lambda}E)^{-1}J_R^\top(\psi^\theta_{0\to 1-\Delta t}(x_0))R(\psi^\theta_{0\to 1-\Delta t}(x_0))
\end{aligned} \tag{39}$$

Taking the limit $\Delta t \to 0$, we obtain:

$$\begin{aligned}
\mathcal{T}_{phys}(x_0) &= \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\
&= -(J_R^\top(\psi^\theta_{0\to 1}(x_0))J_R(\psi^\theta_{0\to 1}(x_0)) + \frac{1}{2\lambda}E)^{-1}J_R^\top(\psi^\theta_{0\to 1}(x_0))R(\psi^\theta_{0\to 1}(x_0))
\end{aligned} \tag{40}$$

B.3.3 Backward Euler Method with $\lambda \to \infty$

In the limit $\lambda \to \infty$, the soft-penalty formulation Equation 37 degenerates into the hard-constraint projection

$$x_1 = \arg\min_x \frac{1}{2}\|x - x_{1-\Delta t}\|^2 \quad \text{s.t.} \quad R(x) = 0. \tag{41}$$

This is a classical projection problem encountered in Position-Based Dynamics (PBD) Müller et al. (2007). Following a similar treatment, we linearize the constraint around $x_{1-\Delta t}$, yielding

$$\min_{\Delta x} \frac{1}{2}\|\Delta x\|^2 \qquad \text{s.t.} \qquad R(x_{1-\Delta t}) + J_R(x_{1-\Delta t})\Delta x = 0. \tag{42}$$

This is a constrained least-squares problem with a linear constraint. Its solution is given by

$$\Delta x = -J_R^\top(x_{1-\Delta t})\big(J_R(x_{1-\Delta t})J_R^\top(x_{1-\Delta t})\big)^{-1}R(x_{1-\Delta t}). \tag{43}$$

Therefore,

$$\begin{aligned}
&\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\
&= -J_R^\top(\psi^\theta_{0\to 1-\Delta t}(x_0))\big(J_R(\psi^\theta_{0\to 1-\Delta t}(x_0))J_R^\top(\psi^\theta_{0\to 1-\Delta t}(x_0))\big)^{-1}R(\psi^\theta_{0\to 1-\Delta t}(x_0))
\end{aligned} \tag{44}$$

Taking the limit $\Delta t \to 0$, we obtain:

$$\begin{aligned}
\mathcal{T}_{phys}(x_0) &= \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0)] \\
&= -J_R^\top(\psi^\theta_{0\to 1}(x_0))\big(J_R(\psi^\theta_{0\to 1}(x_0))J_R^\top(\psi^\theta_{0\to 1}(x_0))\big)^{-1}R(\psi^\theta_{0\to 1}(x_0))
\end{aligned} \tag{45}$$

C Additional Theoretical Results and Algorithms

C.1 Conditional Generation

In the conditional generation setting, we train a conditional flow map $\psi_{t\to r}(\cdot|c)$, where c denotes the condition. Therefore, under our long-short flow-map formulation, the loss in Equation 6 becomes

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0\sim p_0}\|\psi^\theta_{0\to 1}(x_0|c) - sg(\psi^\theta_{0\to 1}(x_0|c) + \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x|c); 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0|c)])\|^2 \tag{46}$$

Here, the direct target becomes $\psi^\theta_{0\to 1}(x_0|c) + \mathcal{T}(x_0|c)$, where $\mathcal{T}(x_0|c) = \lim_{\Delta t\to 0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x|c);\, 1-\Delta t, 1, \psi^\theta_{0\to 1-\Delta t}(x_0|c)]$. We can see that the formulas for conditional generation are obtained by replacing $\psi_{t\to r}$ with $\psi_{t\to r}(\cdot|c)$. Similarly, Table 1 and Table 2 can be directly modified as follows:

Table 8: Three standard discretizations of the data potential-driven dynamics for conditional generation.

Scheme	Update formula	Target
FE	$x_{n+1} = x_n - \Delta t\, \nabla U_{\text{data}}(x_n, t_n)$	$\mathcal{T}_{data}(x_0) = \frac{\mathbb{E}_{x_1\sim p_1(\cdot\|c)}[(x_1 - \psi^\theta_{0\to1}(x_0\|c))k^\epsilon_1(x_1, \psi^\theta_{0\to1}(x_0\|c))]}{\mathbb{E}_{x_1\sim p_1(\cdot\|c)}[k^\epsilon_1(x_1, \psi^\theta_{0\to1}(x_0\|c))]}$
BE	$x_{n+1} = x_n - \Delta t \nabla U_{\text{data}}(x_{n+1}, t_{n+1})$	$\mathcal{T}_{data}(x_0\|c) = \frac{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}(\cdot\|c)]_\sharp p_0}[(\psi^\theta_{0\to1}(x_0\|c) - \tilde{x}_1)k^\epsilon_1(\psi^\theta_{0\to1}(x_0\|c), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}(\cdot\|c)]_\sharp p_0}[k^\epsilon_1(\psi^\theta_{0\to1}(x_0\|c), \tilde{x}_1)]}$
TR	$x_{n+1} = x_n - \frac{\Delta t}{2}(\nabla U_{\text{data}}(x_n, t_n) + \nabla U_{\text{data}}(x_{n+1}, t_{n+1}))$	$\mathcal{T}_{data}(x_0\|c) = \frac{1}{2}\frac{\mathbb{E}_{x_1\sim p_1(\cdot\|c)}[(x_1 - \psi^\theta_{0\to1}(x_0\|c))k^\epsilon_1(x_1, \psi^\theta_{0\to1}(x_0\|c))]}{\mathbb{E}_{x_1\sim p_1(\cdot\|c)}[k^\epsilon_1(x_1, \psi^\theta_{0\to1}(x_0\|c))]} + \frac{1}{2}\frac{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}(\cdot\|c)]_\sharp p_0}[(\psi^\theta_{0\to1}(x_0\|c) - \tilde{x}_1)k^\epsilon_1(\psi^\theta_{0\to1}(x_0\|c), \tilde{x}_1)]}{\mathbb{E}_{\tilde{x}_1\sim[\psi^\theta_{0\to1}(\cdot\|c)]_\sharp p_0}[k^\epsilon_1(\psi^\theta_{0\to1}(x_0\|c), \tilde{x}_1)]}$

Table 9: Three standard discretizations of the Physics potential-driven dynamics for conditional generation.

Scheme	Update formula	Target
FE	$x_{n+1} = x_n - \Delta t\lambda \nabla U_{\text{phys}}(x_n, t_n)$	$\mathcal{T}_{phys}(x_0\|c) = -\lambda J_R^\top(\psi^\theta_{0\to1}(x_0\|c))R(\psi^\theta_{0\to1}(x_0\|c))$
BE	$x_{n+1} = x_n - \Delta t\lambda \nabla U_{\text{phys}}(x_{n+1}, t_{n+1})$	$\mathcal{T}_{phys}(x_0\|c) = -(J_R^\top(\psi^\theta_{0\to1}(x_0\|c))J_R(\psi^\theta_{0\to1}(x_0\|c)) + \gamma E)^{-1} J_R^\top(\psi^\theta_{0\to1}(x_0\|c))R(\psi^\theta_{0\to1}(x_0\|c))$
BE-Lim	$x_{n+1} = x_n - \Delta t\lambda \nabla U_{\text{phys}}(x_{n+1}, t_{n+1})$, $\lambda \to \infty$	$\mathcal{T}_{phys}(x_0\|c) = -J_R^\top(\psi^\theta_{0\to1}(x_0\|c))(J_R(\psi^\theta_{0\to1}(x_0\|c)) J_R^\top(\psi^\theta_{0\to1}(x_0\|c)))^{-1} R(\psi^\theta_{0\to1}(x_0\|c))$

Based on the above modification for conditional generation, Algorithm 1 can be adapted into a grouping-based implementation, as shown in Algorithm 2. Specifically, we sample B_c conditions, sample B_s data samples for each condition, and sample $B = B_c B_s$ noise samples. Each step in Algorithm 2 is then computed within each condition group and averaged over all B_c conditions. After training, the learned flow map $\psi^\theta_{0\to1}(\cdot|c)$ can be used to generate samples that satisfy both the physical constraints and the data distribution with condition c input, namely,

$$x_1^\theta = \psi^\theta_{0\to1}(x_0|c) \Rightarrow x_1^\theta \sim p_1(\cdot|c), \mathcal{R}(x_1^\theta) = 0 \tag{47}$$

Algorithm 2 PI-LSFM Conditional Training

Require: dataset $\mathcal{D}$, condition batch size B_c, samples per condition B_s, initial model parameters θ, learning rate η, Gaussian noise sampler $\mathcal{N}(0, I)$, hyperparameter $\bar{\lambda}$

repeat
  Sample conditions $\{c_k\}_{k=1}^{B_c}$
  for $k = 1, \dots, B_c$ **do**
    Sample $\mathcal{X}_1^{(k)} = \{x_1^{k,j}\}_{j=1}^{B_s}$ from $\mathcal{D}(\cdot|c_k)$
    Sample noise $\mathcal{X}_0^{(k)} = \{x_0^{k,j}\}_{j=1}^{B_s}$ from $\mathcal{N}(0, I)$
    Compute $x_1^{k,j,\theta} \leftarrow \psi_{0\to1}^{\theta}(x_0^{k,j}|c_k)$ for $j = 1, \dots, B_s$
    Compute $\mathcal{T}_{\text{data}}(x_0^{k,j}|c_k)$ using tab. 8
    Compute $\mathcal{T}_{\text{phys}}(x_0^{k,j}|c_k)$ using tab. 9
    $\mathcal{T}(x_0^{k,j}|c_k) \leftarrow \mathcal{T}_{\text{data}}(x_0^{k,j}|c_k) + \bar{\lambda}\mathcal{T}_{\text{phys}}(x_0^{k,j}|c_k)$
  end for
  $\mathcal{L}(\theta) \leftarrow \frac{1}{B_c B_s}\sum_{k=1}^{B_c}\sum_{j=1}^{B_s}\|x_1^{k,j,\theta} - sg(x_1^{k,j,\theta} + \mathcal{T}(x_0^{k,j}|c_k))\|^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta)$
until convergence

Algorithm 3 PI-LSFM Feature-Space Training

Require: dataset $\mathcal{D}$, batch size B, feature encoder ϕ, initial model parameters θ, learning rate η, Gaussian noise sampler $\mathcal{N}(0, I)$, hyperparameter $\bar{\lambda}$

repeat
  Sample $\mathcal{X}_1 = \{x_1^j\}_{j=1}^{B}$ from $\mathcal{D}$
  Sample noise $\mathcal{X}_0 = \{x_0^i\}_{i=1}^{B}$ from $\mathcal{N}(0, I)$
  $x_1^{i,\theta} \leftarrow \psi_{0\to1}^{\theta}(x_0^i)$ and $\mathcal{X}_1^{\theta} = \{x_1^{i,\theta}\}_{i=1}^{B}$
  Compute feature sets $\phi(\mathcal{X}_1)$ and $\phi(\mathcal{X}_1^{\theta})$
  Compute $\{\mathcal{T}_{\text{data}}^{\mathcal{Z}}(x_0^i)\}_{i=1}^{B}$ in feature space using tab. 10
  Compute $\{\mathcal{T}_{\text{phys}}(x_0^i)\}_{i=1}^{B}$ in the original space using tab. 2
  $\mathcal{L}(\theta) \leftarrow \frac{1}{B}\sum_{i=1}^{B}\|\phi(x_1^{i,\theta}) - sg(\phi(x_1^{i,\theta} + \mathcal{T}_{phys}(x_0^i)) + \mathcal{T}_{data}^{\mathcal{Z}}(x_0^i))\|^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta)$
until convergence

Figure 5: Conditional and feature-space implementations of PI-LSFM training.

C.2 Stabilization via Feature Spaces

As discussed above, directly computing the kernel $k_t^{\epsilon}(y, x)$ in the high-dimensional data space may suffer from the curse of dimensionality, making the kernel weights less informative and the estimated direction unstable. To alleviate this issue, we compute the kernel in a lower-dimensional feature space.

Let $\phi : \mathbb{R}^n \to \mathbb{R}^m$ be an encoder with $m < n$, and define the feature variable $\mathcal{Z} = \phi(X)$. Let $Z_1 = \phi(X_1)$, where $X_1 \sim p_1$, and denote the density of Z_1 by p_1^Z. Equivalently, for any test function f,

$$\mathbb{E}_{Z_1\sim p_1^Z}[f(Z_1)] = \mathbb{E}_{X_1\sim p_1}[f(\phi(X_1))]. \tag{48}$$

If Flow Matching is trained in the feature space Z, then, similarly to Theorem 3.1, the corresponding dynamics can be interpreted as a potential-driven system $\frac{d}{dt}z_t = -\nabla_z U_{\text{data}}^Z(z_t, t)$, with potential $U_{\text{data}}^Z(z, t)$

$$U_{\text{data}}^Z(z,t) = -\frac{1-t}{t}\log\mathbb{E}_{Z_1\sim p_1^Z}[k_t^Z(Z_1, z)] - \frac{1}{2t}\|z\|^2 + C_t^Z, \qquad k_t^Z(y,z) = e^{(-\frac{\|ty-z\|^2}{2(1-t)^2})}. \tag{49}$$

Equivalently,

$$U_{\text{data}}^Z(z,t) = -\frac{1-t}{t}\log\mathbb{E}_{X_1\sim p_1}[k_t^Z(\phi(X_1), z)] - \frac{1}{2t}\|z\|^2 + C_t^Z. \tag{50}$$

The kernel in $U_{\text{data}}^Z(z, t)$ is computed in the low-dimensional feature space, where the kernel weights are more stable. We next relate this feature-space potential to the original data-space potential, so that $U_{\text{data}}^Z(z, t)$ can be used to characterize the dynamics on $\mathcal{X}$.

Theorem C.1 (Local-linear feature-data potential gradient relation)**.** *Let $z = \phi(x)$, and let $p_{x|z}(x|z)$ denote the conditional density induced by $z = \phi(x)$. Assume that, near $t = 1$, we approximately have $\phi(X_t) = Z_t$, where X_t and Z_t denote the conditional paths in the spaces $\mathcal{X}$ and $\mathcal{Z}$, respectively. Then, in the near-endpoint regime $t \to 1$, we have the following approximation:*

$$\nabla_z U_{\text{data}}^Z(z,t) = \mathbb{E}_{x\sim p_{x|z}}[J_\phi(x)\nabla_x U_{\text{data}}(x,t)]. \tag{51}$$

Proof. By the definition of the potential-driven dynamics, we have $u_t^x(x) = -\nabla_x U_{\text{data}}(x,t)$ and $u_t^z(z) = -\nabla_z U^Z_{\text{data}}(z,t)$, where $u_t^x(x)$ and $u_t^z(z)$ denote the Flow Matching velocities on $\mathcal{X}$ and $\mathcal{Z}$, respectively. We now relate $u_t^x(x)$ and $u_t^z(z)$. Near $t=1$, we approximately have $\phi(X_t) = Z_t$, and then we have

$$\begin{aligned}
u_t^{z,*}(z) &= \mathbb{E}[\frac{Z_1 - z}{1-t}|Z_t = z] \\
&= \mathbb{E}[\mathbb{E}[\frac{Z_1 - z}{1-t}|Z_t = z, X_t]|Z_t = z] \\
&= \mathbb{E}_{x\sim p_{x|z}}[\mathbb{E}[\frac{Z_1 - z}{1-t}|Z_t = z, X_t = x]] \\
&\overset{Z_t=\phi(X_t)}{=} \mathbb{E}_{x\sim p_{x|z}}[\mathbb{E}[\frac{Z_1 - \phi(x)}{1-t}|X_t = x]] \\
&= \mathbb{E}_{x\sim p_{x|z}}[\frac{\mathbb{E}[Z_1|X_t = x] - \phi(x)}{1-t}] \\
&\overset{①}{=} \mathbb{E}_{x\sim p_{x|z}}[\frac{\mathbb{E}[\phi(X_1)|X_t = x] - \phi(x)}{1-t}] \\
&= \mathbb{E}_{x\sim p_{x|z}}[\mathbb{E}[\frac{\phi(X_1) - \phi(x)}{1-t}|X_t = x]] \\
&\overset{②}{\approx} \mathbb{E}_{x\sim p_{x|z}}[J_\phi(x)\mathbb{E}[\frac{X_1 - x}{1-t}|X_t = x]] \\
&= \mathbb{E}_{x\sim p_{x|z}}[J_\phi(x) u_t^{x,*}(x)].
\end{aligned} \tag{52}$$

where ② follows from a Taylor expansion and ① holds because

$$\begin{aligned}
\mathbb{E}[Z_1|X_t = x] &= \mathbb{E}\big[\mathbb{E}[Z_1|X_t = x, X_1]|X_t = x\big] \\
&= \mathbb{E}\big[\phi(X_1)|X_t = x\big].
\end{aligned} \tag{53}$$

Substituting $u_t^{x,*}(x) = -\nabla_x U_{\text{data}}(x,t)$ and $u_t^{z,*}(z) = -\nabla_z U^Z_{\text{data}}(z,t)$, we obtain

$$-\nabla_z U^Z_{\text{data}}(z,t) = \mathbb{E}_{x\sim p_{x|z}}[J_\phi(x)(-\nabla_x U_{\text{data}}(x,t))]. \tag{54}$$

This completes the proof. □

We then consider learning trajectory consistency in the feature space using Equation 5, namely $\|\phi(\psi_{0\to1}(x_0)) - \phi(\psi_{1-\Delta t\to1} \circ \psi_{0\to1-\Delta t}(x_0))\|^2$. We assume that the encoder ϕ is well-behaved and approximately invertible. In this case, Equation 6 becomes

$$\begin{aligned}
&\mathcal{L}(\theta) \\
&= \mathbb{E}_{x_0\sim p_0}\|\phi(\psi^\theta_{0\to1}(x_0)) - sg(\phi(\psi^\theta_{0\to1}(x_0) + \lim_{\Delta t\to0}\mathcal{I}[\frac{d}{dt}\psi_{1-\Delta t\to t}(x); 1-\Delta t, 1, \psi^\theta_{0\to1-\Delta t}(x_0)]))\|^2 \\
&= \mathbb{E}_{x_0\sim p_0}\|\phi(\psi^\theta_{0\to1}(x_0)) - sg(\phi(\psi^\theta_{0\to1}(x_0) + \mathcal{T}_{phys}(x_0) - \lim_{\Delta t\to0}\mathcal{I}[\nabla U_{data}(x); 1-\Delta t, 1, \psi^\theta_{0\to1-\Delta t}(x_0)]))\|^2 \\
&\overset{①}{\approx} \mathbb{E}_{x_0\sim p_0}\|\phi(\psi^\theta_{0\to1}(x_0)) - sg(\phi(\psi^\theta_{0\to1}(x_0) + \mathcal{T}_{phys}(x_0)) - \lim_{\Delta t\to0}\mathcal{I}[J_\phi(x)\nabla U_{data}(x); 1-\Delta t, 1, \psi^\theta_{0\to1-\Delta t}(x_0)])\|^2 \\
&= \mathbb{E}_{x_0\sim p_0}\|\phi(\psi^\theta_{0\to1}(x_0)) - sg(\phi(\psi^\theta_{0\to1}(x_0) + \mathcal{T}_{phys}(x_0)) - \lim_{\Delta t\to0}\mathcal{I}[\nabla_z U^{\mathcal{Z}}_{data}; 1-\Delta t, 1, \phi(\psi^\theta_{0\to1-\Delta t}(x_0))])\|^2 \\
&= \mathbb{E}_{x_0\sim p_0}\|\phi(\psi^\theta_{0\to1}(x_0)) - sg(\phi(\psi^\theta_{0\to1}(x_0) + \mathcal{T}_{phys}(x_0)) + \mathcal{T}^{\mathcal{Z}}_{data}(x_0))\|^2
\end{aligned} \tag{55}$$

where ① follows from a Taylor expansion. This is exactly the formula stated in the main text as Equation 13, where $\mathcal{T}^{\mathcal{Z}}_{data}(x_0) = \lim_{\Delta t\to0}\mathcal{I}[\nabla_z U^{\mathcal{Z}}_{data}; 1-\Delta t, 1, \phi(\psi^\theta_{0\to1-\Delta t}(x_0))]$ is numerically computed in the feature space. Note that the physics target $\mathcal{T}_{phys}(x_0)$ does not require any modification; only the data target $\mathcal{T}^{\mathcal{Z}}_{data}(x_0)$ needs to be computed in the feature space. The corresponding targets obtained by different numerical schemes in the feature space are given in Table 10. Our derivation relies on the assumption that the encoder ϕ is well defined and well behaved. Empirically, we find that the encoders used in our experiments work effectively under this formulation.

It is worth noting that the use of the encoder and feature space only affects the training process. During inference, generation is still performed in the original space $\mathcal{X}$. Therefore, after learning the flow map $\psi^{\theta}_{0\to 1}$, the inference procedure is the same as in subsection 3.3.

$$x_1^{\theta} = \psi^{\theta}_{0\to 1}(x_0) \Rightarrow x_1^{\theta} \sim p_1, \mathcal{R}(x_1^{\theta}) = 0 \tag{56}$$

Table 10: Three standard discretizations of the data potential-driven dynamics in feature space.

Scheme	**Update formula**	**Target**
FE	$z_{n+1} = z_n - \Delta t\, \nabla U^{\mathcal{Z}}_{\text{data}}(z_n, t_n)$	$\mathcal{T}^{\mathcal{Z}}_{data}(x_0) = \frac{\mathbb{E}_{z_1\sim p_1^{\mathcal{Z}}}[(z_1 - \phi(\psi^{\theta}_{0\to 1}(x_0)))k_1^{\epsilon}(z_1, \phi(\psi^{\theta}_{0\to 1}(x_0)))]}{\mathbb{E}_{z_1\sim p_1^{\mathcal{Z}}}[k_1^{\epsilon}(z_1, \phi(\psi^{\theta}_{0\to 1}(x_0)))]}$
BE	$z_{n+1} = z_n - \Delta t \nabla U^{\mathcal{Z}}_{\text{data}}(z_{n+1}, t_{n+1})$	$\mathcal{T}^{\mathcal{Z}}_{data}(x_0) = \frac{\mathbb{E}_{\tilde{z}_1\sim[\phi\circ\psi^{\theta}_{0\to 1}]_\sharp p_0}[(\phi(\psi^{\theta}_{0\to 1}(x_0)) - \tilde{z}_1)k_1^{\epsilon}(\phi(\psi^{\theta}_{0\to 1}(x_0)), \tilde{z}_1)]}{\mathbb{E}_{\tilde{z}_1\sim[\phi\circ\psi^{\theta}_{0\to 1}]_\sharp p_0}[k_1^{\epsilon}(\phi(\psi^{\theta}_{0\to 1}(x_0)), \tilde{z}_1)]}$
TR	$z_{n+1} = z_n - \frac{\Delta t}{2}(\nabla U^{\mathcal{Z}}_{\text{data}}(z_n, t_n) + \nabla U^{\mathcal{Z}}_{\text{data}}(z_{n+1}, t_{n+1}))$	$\mathcal{T}^{\mathcal{Z}}_{data}(x_0) = \frac{1}{2}\frac{\mathbb{E}_{z_1\sim p_1^{\mathcal{Z}}}[(z_1 - \phi(\psi^{\theta}_{0\to 1}(x_0)))k_1^{\epsilon}(z_1, \phi(\psi^{\theta}_{0\to 1}(x_0)))]}{\mathbb{E}_{z_1\sim p_1^{\mathcal{Z}}}[k_1^{\epsilon}(z_1, \phi(\psi^{\theta}_{0\to 1}(x_0)))]} + \frac{1}{2}\frac{\mathbb{E}_{\tilde{z}_1\sim[\phi\circ\psi^{\theta}_{0\to 1}]_\sharp p_0}[(\phi(\psi^{\theta}_{0\to 1}(x_0)) - \tilde{z}_1)k_1^{\epsilon}(\phi(\psi^{\theta}_{0\to 1}(x_0)), \tilde{z}_1)]}{\mathbb{E}_{\tilde{z}_1\sim[\phi\circ\psi^{\theta}_{0\to 1}]_\sharp p_0}[k_1^{\epsilon}(\phi(\psi^{\theta}_{0\to 1}(x_0)), \tilde{z}_1)]}$

C.3 PyTorch Code

Here, we take the trapezoidal rule of Table 1 as an example and provide the parallel PyTorch implementation of the expectation computation in Equation 12. In each update, we sample a data batch $\mathcal{X}_1$ and compute a generated batch $\mathcal{X}_1^{\theta}$ by mapping noise samples through the learned flow map. The generated batch $\mathcal{X}_1^{\theta}$ is also used as samples from the distribution $[\psi^{\theta}_{0\to 1}]_\sharp p_0$, denoted by $\tilde{\mathcal{X}}_1 = \mathcal{X}_1^{\theta}$.

For each generated sample $x_1^{\theta,i} \in \mathcal{X}_1^{\theta}$, we compute kernel-weighted averages over all samples in the data batch $\mathcal{X}_1$ and the generated batch $\tilde{\mathcal{X}}_1$. These kernel-weighted averages provide empirical estimates of the expectations used to construct the target $\mathcal{T}_{data}(x_0^i)$ for $x_1^{\theta,i}$. The expectation computation for all samples in $\mathcal{X}_1^{\theta}$ is fully parallelized, and the corresponding PyTorch implementation is shown below.

```
import torch

def compute_kernel_weighted_targets(
    x1_theta: torch.Tensor, #[B...]
    x1: torch.Tensor, #[B...]
    tilde_x1: torch.Tensor, #[B...]
    epsilon: float,
    eps: float = 1e-12,
):
    dist2_data = torch.cdist(x1_theta, x1, p=2) ** 2
    weights_data = torch.exp(-dist2_data / epsilon).clamp_min(eps)
    V_forward = (weights_data @ x1) / weights_data.sum(dim=1, keepdim=
        True)

    dist2_tilde = torch.cdist(x1_theta, tilde_x1, p=2) ** 2
    weights_tilde = torch.exp(-dist2_tilde / epsilon).clamp_min(eps)
    V_backward = -(weights_tilde @ tilde_x1) / weights_tilde.sum(dim=1,
        keepdim=True)

    return 0.5*(V_forward+V_backward)
```

D Design Study and Additional Results

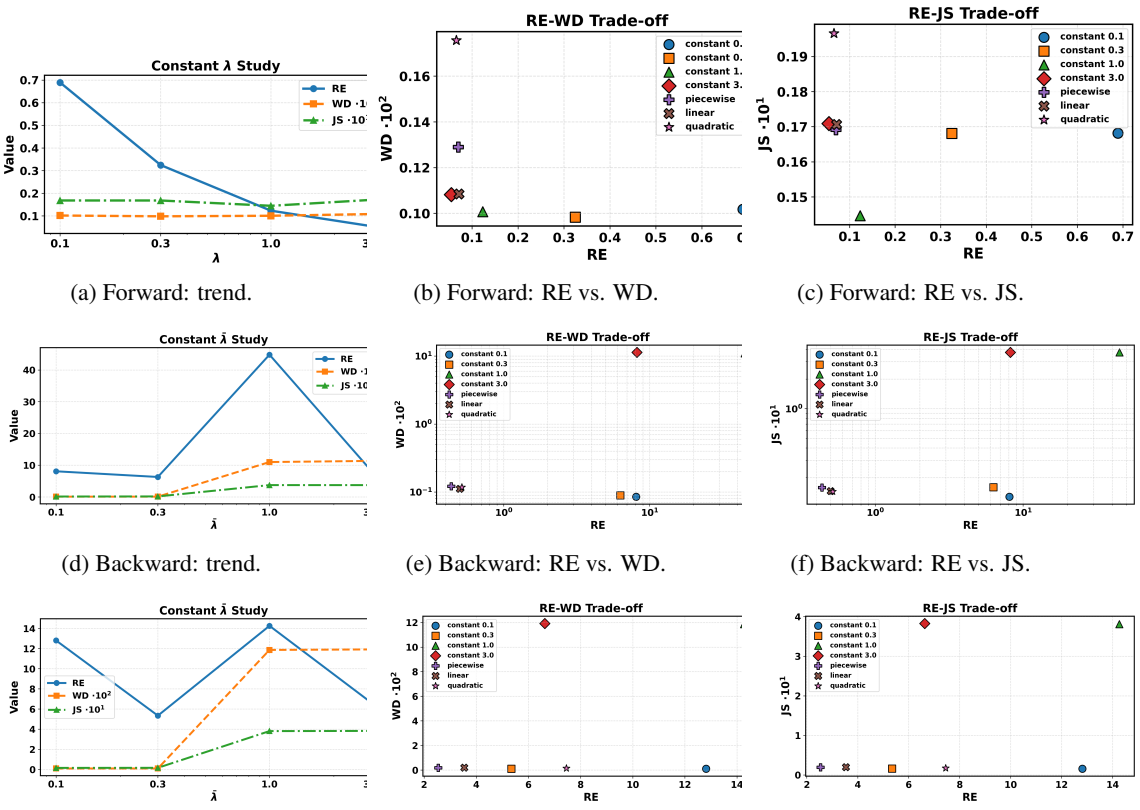
D.1 Physics Term Numerical Integration Study

We next study the impact of different numerical discretizations in Table 2 for the physics term. Specifically, we compare Forward Euler, Backward Euler, and Backward Euler(limit), while fixing the data term to its best-performing configuration, namely the trapezoidal rule.

In the experiments, we evaluate four constant values of the coefficient $\bar{\lambda}$ from 0.1 to 3.0, which controls the relative strength between the data term and the physics term. For each numerical scheme, we plot the trends of RE, WD, and JS as $\bar{\lambda}$ varies, as shown in Figure 6a,Figure 6d and Figure 6g. Unlike Forward Euler, Backward Euler does not exhibit a simple smooth trend. In particular, as shown in Figure 6d, WD and JS increase sharply when $\bar{\lambda}$ changes from 0.3 to 0.1. To address this sensitivity, we additionally consider three scheduling strategies for $\bar{\lambda}$: piecewise, linear, and quadratic schedules. Let s denote the training step, and let s_{warm} and s_{ramp} denote the end of the warm-up stage and ramping stage, respectively. The piecewise schedule is defined as $\bar{\lambda}(s) = \begin{cases} 0.1, & s < s_{\text{warm}}, \\ 1.0, & s_{\text{warm}} \le s < s_{\text{ramp}}, \\ 3.0, & s \ge s_{\text{ramp}}. \end{cases}$ with $s_{\text{ramp}} = 200k$ and $s_{\text{warm}} = 50k$. For the linear and quadratic schedules, we set $\bar{\lambda}_{\text{start}} = 0.1$ and $\bar{\lambda}_{\text{end}} = 3.0$, define $\tau(s) = \frac{s - s_{\text{warm}}}{s_{\text{ramp}} - s_{\text{warm}}}$ and then the linear schedule is $\bar{\lambda}_{\text{lin}}(s) = \bar{\lambda}_{\text{start}} + (\bar{\lambda}_{\text{end}} - \bar{\lambda}_{\text{start}})\tau(s)$ and the quadratic schedule is $\bar{\lambda}_{\text{quad}}(s) = \bar{\lambda}_{\text{start}} + (\bar{\lambda}_{\text{end}} - \bar{\lambda}_{\text{start}})\tau(s)^2$. For $s < s_{\text{warm}}$, both schedules use $\bar{\lambda}_{\text{start}}$, and for $s \ge s_{\text{ramp}}$, both schedules use $\bar{\lambda}_{\text{end}}$.

To better compare the trade-off between the data term and the physics term under different constant and scheduled choices of $\bar{\lambda}$, we also plot RE-WD and RE-JS scatter plots. These plots allow us to select the best configuration for each numerical scheme. We observe that Forward Euler consistently achieves better trade-offs across RE, WD, and JS compared with the two implicit variants. We then select the best-performing $\bar{\lambda}$ for each method and report the corresponding RE, WD, and JS in Figure 4c. The selected configurations are $\bar{\lambda} = 3.0$ for Forward Euler, the piecewise schedule for Backward Euler, and the piecewise schedule for Backward Euler (limit).

All experiments use the same model architecture and training configuration as in subsection E.2. We train each variant for 400K steps. Overall, Forward Euler achieves the best performance for the physics term. We hypothesize that this is because the physics term has an explicit analytical form, so the steepest-descent direction provided by Forward Euler gives the most direct and stable correction.



(a) Forward: trend. (b) Forward: RE vs. WD. (c) Forward: RE vs. JS.

(d) Backward: trend. (e) Backward: RE vs. WD. (f) Backward: RE vs. JS.

(g) Backward (Limit): trend. (h) Backward (Limit): RE vs. WD. (i) Backward (Limit): RE vs. JS.

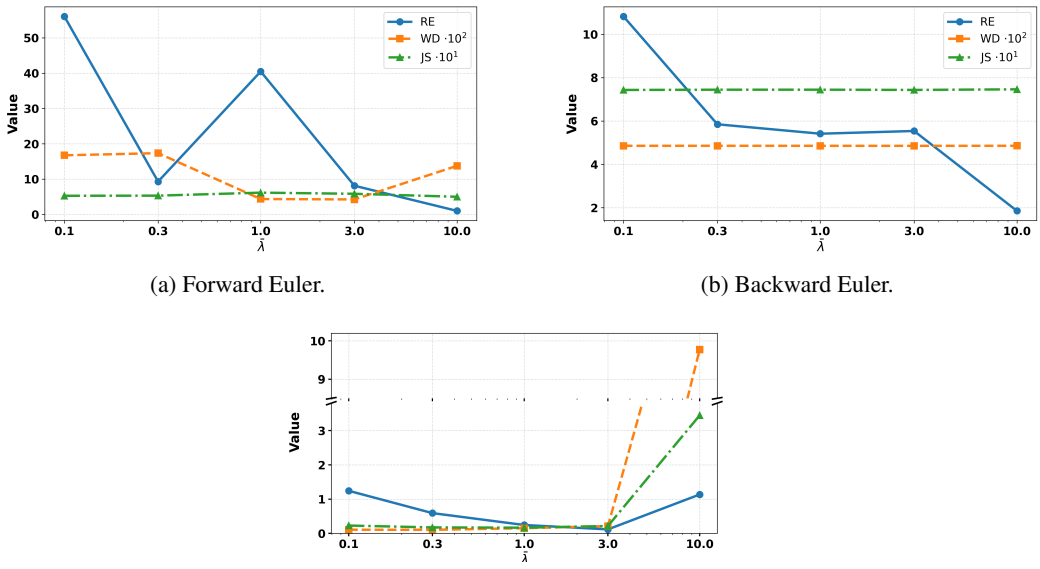
Figure 6: Comparison of constant step sizes and scheduling strategies for different physics-term discretizations. Each row corresponds to one numerical scheme, while columns show metric trends, RE–WD trade-offs, and RE–JS trade-offs. Due to the large dynamic range of the metrics for the Backward Euler methods, logarithmic scales are used in the scatter plots for better visualization.

D.2 Data Term Numerical Integration Study

We next study the effect of different numerical schemes in Table 1 for the data term. Similar to the physics-term study, we compare Forward Euler, Backward Euler, and the trapezoidal rule, while fixing the physics term to its best-performing configuration, namely the Forward Euler method.

In the experiments, we evaluate five values of the coefficient $\bar{\lambda}$ from 0.3 to 10.0, which controls the relative strength between the data term and the physics term. For each numerical scheme, we plot the trends of RE, WD, and JS as $\bar{\lambda}$ varies, as shown in Figure 7. The trapezoidal rule consistently yields smaller values across all three metrics compared with the two Euler schemes. We then select the best-performing $\bar{\lambda}$ for each method and report the corresponding RE, WD, and JS in Figure 4d. The selected values are $\bar{\lambda} = 3.0$ for Forward Euler, $\bar{\lambda} = 10.0$ for Backward Euler, and $\bar{\lambda} = 3.0$ for the trapezoidal rule.

All experiments use the same model architecture and training configuration as in subsection E.2. To reduce computational cost, we train each variant for 100K steps. In contrast to the physics term, we find that the second-order trapezoidal rule consistently achieves the best performance for the data term. This can be attributed to the kernel-based expectation in the data target, where a higher-order discretization provides a more accurate approximation of the near-terminal dynamics.



(a) Forward Euler.

(b) Backward Euler.

(c) Trapezoidal rule.

Figure 7: Effect of the coefficient $\bar{\lambda}$ under different numerical discretizations.

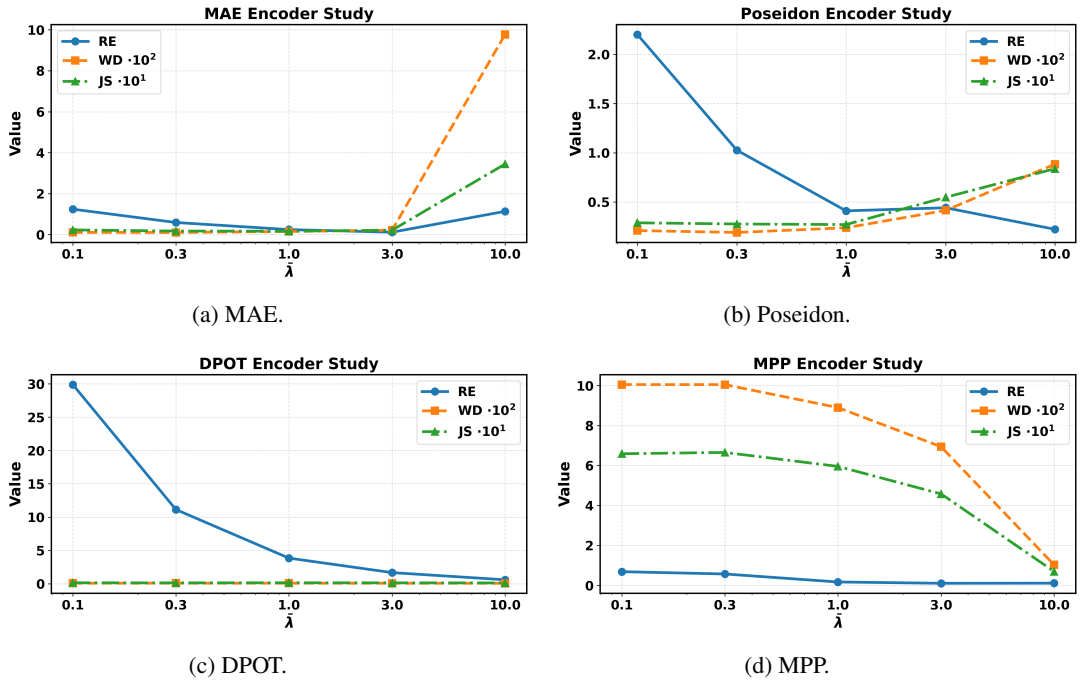
D.3 Encoder Study

We investigate the effect of different feature encoders used in the feature-space stabilization strategy described in subsection 3.3. The corresponding training procedure is summarized in Algorithm 3. We consider several representative encoder choices, including a multi-resolution MAE He et al. (2022) trained on the target dataset, the pretrained U-Net-based physics foundation model Poseidon Herde et al. (2024), and the transformer-based physics foundation models DPOT Hao et al. (2024) and MPP McCabe et al. (2024). For a fair comparison, we keep the encoder parameter counts approximately matched at around 30M parameters. For Poseidon, we use the basic variant, while for DPOT and MPP, we use their small variants to ensure comparable model sizes.

The MAE encoder is implemented as a ResNet–U-Net architecture trained directly on the Darcy dataset (for Kolmogorov flow, we use the same architecture and training setup, only adapting the input and output channels). Each input consists of two channels, which are first patchified using 2×2 patches, resulting in an 8-channel representation on a 32×32 grid. The encoder follows a ResNet34-style He et al. (2016) design with Group Normalization, producing multi-resolution feature maps with channel widths $64, 128, 256$, and 512. The decoder adopts a U-Net structure with bilinear upsampling and skip connections to reconstruct the input. During training, we randomly mask 50% of the patches and optimize a mean-squared reconstruction loss on the masked regions. The model is trained with batch size 256 using AdamW, with an initial learning rate of 5×10^{-4} decayed to 1.25×10^{-4}, weight decay 0, and $\beta = (0.9, 0.999)$, together with an exponential moving average with decay 0.9995. The MAE is trained for 400K steps, after which it is frozen and used as the feature encoder.

We keep the neural representation of the flow map and the training configuration the same as in subsection E.2. To reduce computational cost, each encoder variant is trained for 100K steps within the PI-LSFM framework. In the experiments, we evaluate five constant values of the coefficient $\bar{\lambda}$ from 0.1 to 10.0, which controls the relative strength between the data term and the physics term. For each encoder, we plot the trends of RE, WD, and JS as $\bar{\lambda}$ varies, as shown in Figure 8. These plots allow us to select the best configuration for each encoder and compare different encoders under their respective best-performing settings. We then report the corresponding RE, WD, JS, and per-step training time in Figure 4b. The selected configurations use $\bar{\lambda} = 3.0$ for MAE, $\bar{\lambda} = 1.0$ for Poseidon, and $\bar{\lambda} = 10.0$ for DPOT and MPP.

We observe that all encoders improve the performance of our model, demonstrating the effectiveness and robustness of feature-space training. Among them, MAE, Poseidon, and DPOT achieve the



(a) MAE. (b) Poseidon.

(c) DPOT. (d) MPP.

Figure 8: Effect of the coefficient $\bar{\lambda}$ for different feature encoders. Each plot shows RE, WD, and JS as $\bar{\lambda}$ varies.

strongest overall results. Since MAE provides the fastest training while maintaining strong performance, we use it as the default encoder in our subsequent experiments. Nevertheless, for settings where training a task-specific encoder is undesirable, pretrained encoders such as Poseidon and DPOT remain strong alternatives.

To construct the feature space, we avoid directly flattening the full encoder feature maps, as this would lead to prohibitively high dimensionality. Instead, following Deng et al. (2026), we aggregate the encoder outputs into a compact set of multi-scale statistics. For each feature map of size $H_i \times W_i \times C_i$, we retain location-wise features, global channel-wise mean and standard deviation, and channel-wise mean and standard deviation computed over spatial patches at multiple resolutions. This design preserves both local and global information while significantly reducing the dimensionality of the feature representation.

In Figure 11, we demonstrate the necessity of feature-space optimization: without a feature encoder, training fails, in contrast to the successful feature-space results in Figure 10.

E Experiment Details and Additional Results

E.1 Toy Example

Dataset To clearly demonstrate the advantage of our direct loss in jointly controlling the physics constraint and the data distribution, we consider a two-dimensional toy dataset for intuitive visualization. Specifically, we construct $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ from a mixture of three Gaussian distributions whose centers are approximately located on the unit circle. The underlying physical constraint is $\|(x, y)\|^2 = 1$.

To simulate a realistic setting where data samples are influenced by both the data distribution and the physical constraint, we do not fully project the samples onto the circle. Instead, we introduce controlled deviations by interpolating between the original Gaussian samples and their projections $x_{\text{final}} = (1 - \alpha)x_{\text{raw}} + \alpha x_{\text{proj}}$, where $\alpha = 0.9$ in all experiments. This produces training samples that approximately satisfy the constraint while preserving the multimodal structure of the data. An

Table 11: Constraint satisfaction on the toy example under different batch sizes. We report RE: physical Residual MSE $\times 10^4$.

Batch size	Ours	sCM ($\bar{\lambda}=0$)	sCM ($\bar{\lambda}=1$)	sCM ($\bar{\lambda}=3$)	sCM ($\bar{\lambda}=10$)	sCM ($\bar{\lambda}=30$)	sCM ($\bar{\lambda}=100$)
$B=3$	8.015	3996.970	220.986	176.003	190.425	773.622	240.384
$B=4$	3.040	574.162	198.419	155.516	146.720	142.761	98.074
$B=8$	0.722	366.265	158.029	142.076	68.378	38.710	33.033

example of this training dataset is shown in Figure 9a, while the fully projected dataset in Figure 9b serves as the desired physics-consistent distribution.

Training Details On this dataset, we compare our method with a single-stage consistency model (sCM Lu & Song (2024)). For sCM, we use the standard consistency objective $\mathcal{L}_{\mathrm{sCM}} = \mathbb{E}[\|F_\theta(x_{t_1},t_1) - \mathrm{sg}(F_\theta(x_{t_2},t_2))\|^2] + \bar{\lambda}\mathbb{E}[\|\mathcal{R}(f_\theta(x_t,t))\|^2]$, where F_θ denotes the raw neural-network output and f_θ denotes the corresponding consistency output used for generation. $\mathcal{R}(x,y) = x^2+y^2-1$ is the physics residual for the circle constraint. Except for the training loss, our method and sCM use the same model architecture and optimization setup. We use a MLP network as the backbone. The network has 8 layers with hidden dimension 128 and sinusoidal time embeddings of dimension 32. Both methods are trained for 40K steps using AdamW with learning rate 5×10^{-5}, weight decay 0, and $\beta = (0.9, 0.999)$. We use exponential moving average (EMA) with decay 0.9999.

Results We visualize the learned distributions in Figure 9 using a small batch size $B=3$, which highlights whether each method can simultaneously recover the multimodal distribution and satisfy the circle constraint. For sCM, we test multiple values of the physics weight λ for a fair comparison. As shown in Figures 9d–9i, physics-informed consistency models struggle to balance the data distribution and the constraint: small λ leads to poor constraint satisfaction, while large λ causes the physics term to dominate and often results in mode collapse. In contrast, our method in Figure 9c enforces the geometric constraint while preserving the three modes along the circle, demonstrating effective joint control of distributional fidelity and physical validity. We further report constraint satisfaction under different batch sizes, including $B=3,4,8$, in Table 11. Our method achieves significantly lower residual errors, outperforming sCM by up to approximately $30\times$ across different batch sizes.

E.2 Darcy Flow

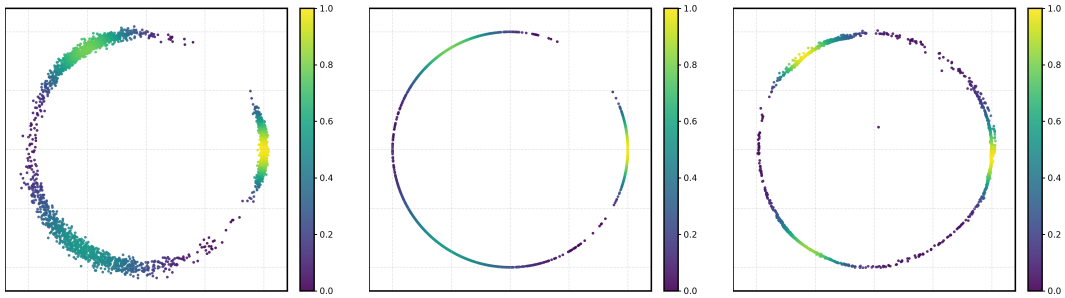
Dataset and constraints. We use the Darcy-flow dataset following Baldan et al. (2026) to enable direct comparison with prior physics-informed generative baselines. The task models steady-state flow in a heterogeneous porous medium on a 64×64 grid, where each sample contains a permeability field K and the corresponding pressure field p. The flow is governed by

$$\begin{aligned} u(x) &= -K(x)\nabla p(x), \qquad x\in\Omega,\\ \nabla\cdot u(x) &= f(x), \qquad x\in\Omega,\\ u(x)\cdot\hat{n}(x) &= 0, \qquad x\in\partial\Omega, \end{aligned} \tag{57}$$

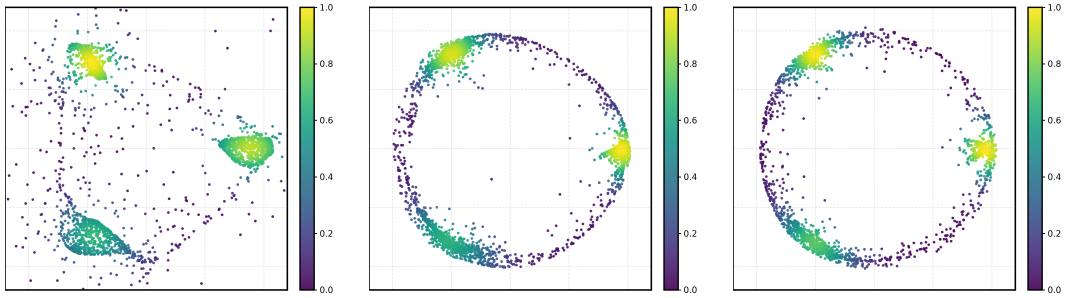
where u is the Darcy velocity, $\hat{n}$ is the outward unit normal on the boundary, and $x=(x_1,x_2)$ denotes the spatial coordinate. Equivalently, the pressure satisfies $\nabla\cdot(K\nabla p)+f=0$ with a no-flux boundary condition. Following Baldan et al. (2026), the forcing term is a localized source–sink pair:

$$f(x) = \begin{cases} r, & \text{if } |x_i - \frac{1}{2}w| \le \frac{1}{2}w, i=1,2\\ -r, & \text{if } |x_i - 1 + \frac{1}{2}w| \le \frac{1}{2}w, i=1,2\\ 0, & \text{otherwise,} \end{cases} \tag{58}$$

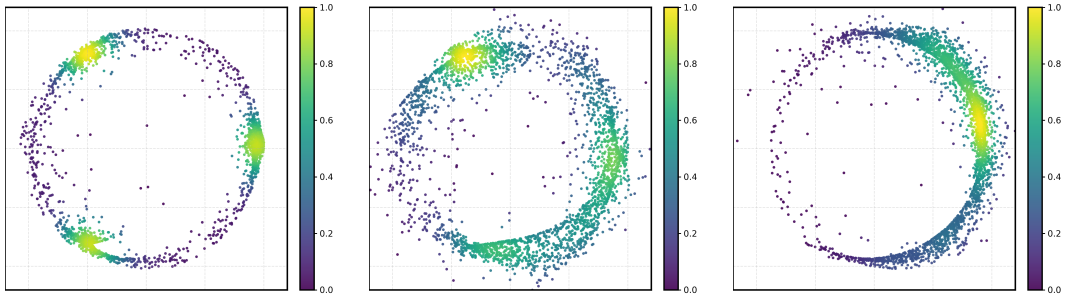
with $r=10$, $w=0.125$ and $x=(x_1,x_2)$ is the 2D coordination of x. The permeability field is generated as $K(x)=\exp(G(x))$, where $G(x)$ is a Gaussian random field. Given K and f, the pressure field p is obtained by solving a least-squares problem using a 64-term truncated Karhunen-Loève expansion, following Jacobsen et al. (2025). Since the no-flux boundary condition determines p only up to an additive constant, the pressure field is normalized by removing its spatial mean.



(a) Dataset (b) Full-projected dataset (c) Our method



(d) sCM ($\bar{\lambda} = 0.0$) (e) sCM ($\bar{\lambda} = 1.0$) (f) sCM ($\bar{\lambda} = 3.0$)



(g) sCM ($\bar{\lambda} = 10.0$) (h) sCM ($\bar{\lambda} = 30.0$) (i) sCM ($\bar{\lambda} = 100.0$)

Figure 9: Toy example visualization comparing the training dataset, the fully projected target distribution, our method, and physics-informed sCM with different values of $\bar{\lambda}$. Our method preserves the three-mode structure while satisfying the circle constraint, and achieves up to approximately $30\times$ lower physical residual error than sCM, as reported in Table 11.

For the physics constraint, we follow the residual-evaluation protocol of Baldan et al. (2026) and use both the PDE residual and the no-flux boundary residual:

$$\begin{aligned} \mathcal{R}_{\mathrm{pde}}(x) &= \nabla \cdot (K \nabla p)(x) + f(x), \\ \mathcal{R}_{\mathrm{bc}}(x) &= u(x) \cdot \hat{n}(x). \end{aligned} \tag{59}$$

In practice, the differential operators are evaluated using fixed finite-difference kernels implemented as 2D convolutional filters, and the forcing field is reconstructed with the same procedure used during dataset generation. The generated pressure field is likewise mean-normalized before evaluating the residuals. This benchmark is unconditional, so no external condition is provided to the generator.

Metrics. We evaluate both physical consistency and distributional fidelity. The residual error (RE) measures the mean squared violation of the governing equation and no-flux boundary condition:

$$\mathrm{RE} = \mathbb{E}_{x \in \Omega}[\mathcal{R}_{\mathrm{pde}}(x)^2] + \mathbb{E}_{x \in \partial\Omega}[\mathcal{R}_{\mathrm{bc}}(x)^2]. \tag{60}$$

For distributional fidelity, we follow Baldan et al. (2026) and report Wasserstein distance (WD) Villani et al. (2009) and Jensen-Shannon divergence (JS) Lin (2002), computed using the same evaluation

protocol and implementations with the SciPy library Gommers et al. (2025). We also report the number of function evaluations (NFE) and inference time (IT) to measure sampling efficiency.

Model and training. We use the basic architecture described in Bastek et al. (2024) without additional architectural optimization. The number of parameters is kept consistent with Baldan et al. (2026), which serves as our main baseline. For experiments with feature-space stabilization, the encoder is used only during training to compute the data target, while generation is still performed in the original data space. The training details of the feature encoder are provided in subsection D.3, and the architecture and training configuration of the main model after encoder pretraining are summarized in Table 12. Note that our method does not require intermediate time inputs t; therefore, for unconditional generation, we remove the time embedder and conditional modulation modules.

Visualization Results. Qualitative results are shown in Figure 10. Our method produces physically consistent samples with coherent pressure and permeability fields, while maintaining low residual errors.

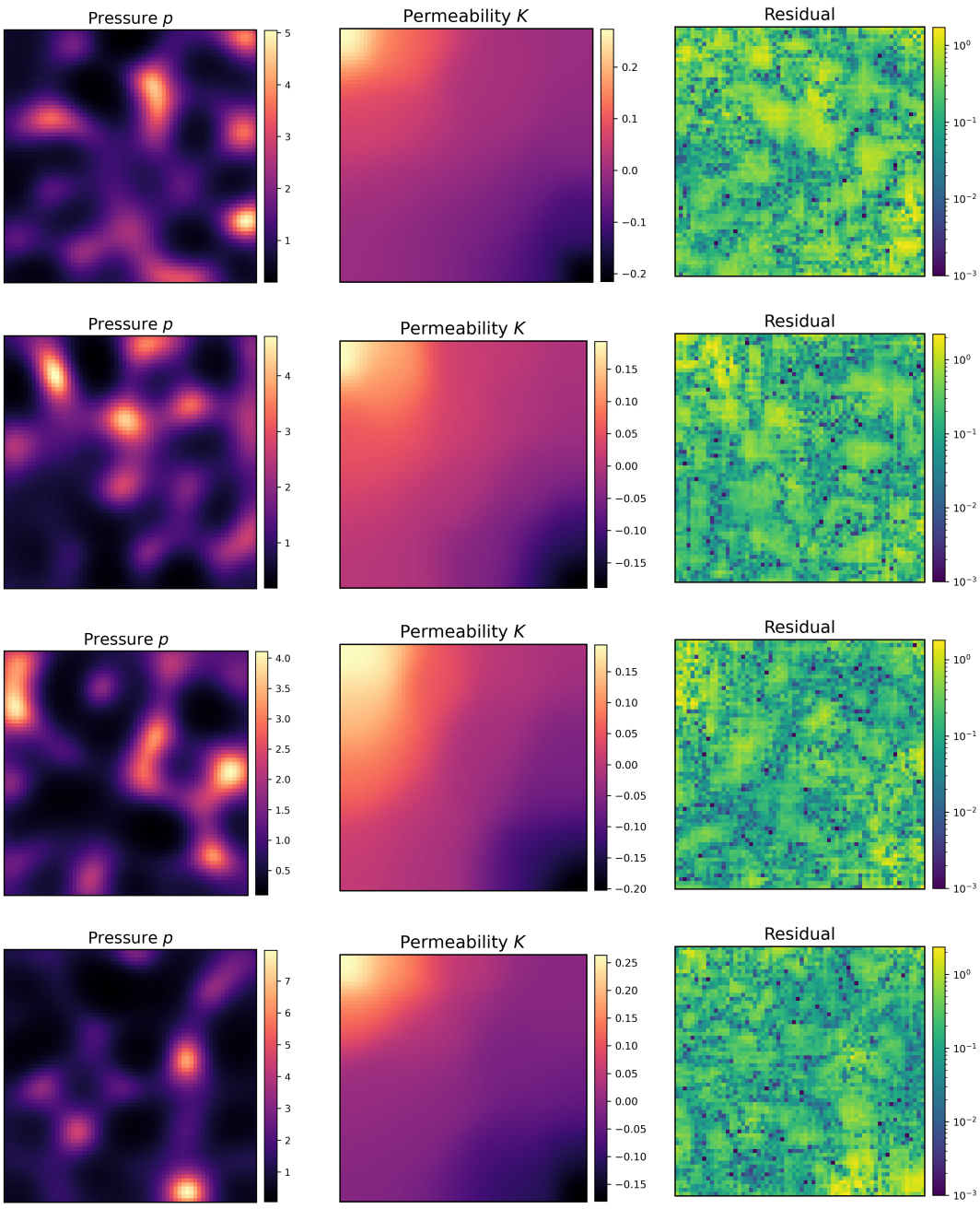


Figure 10: Generated Darcy flow samples. Each row shows one sample, with columns corresponding to pressure p, permeability K, and residual.

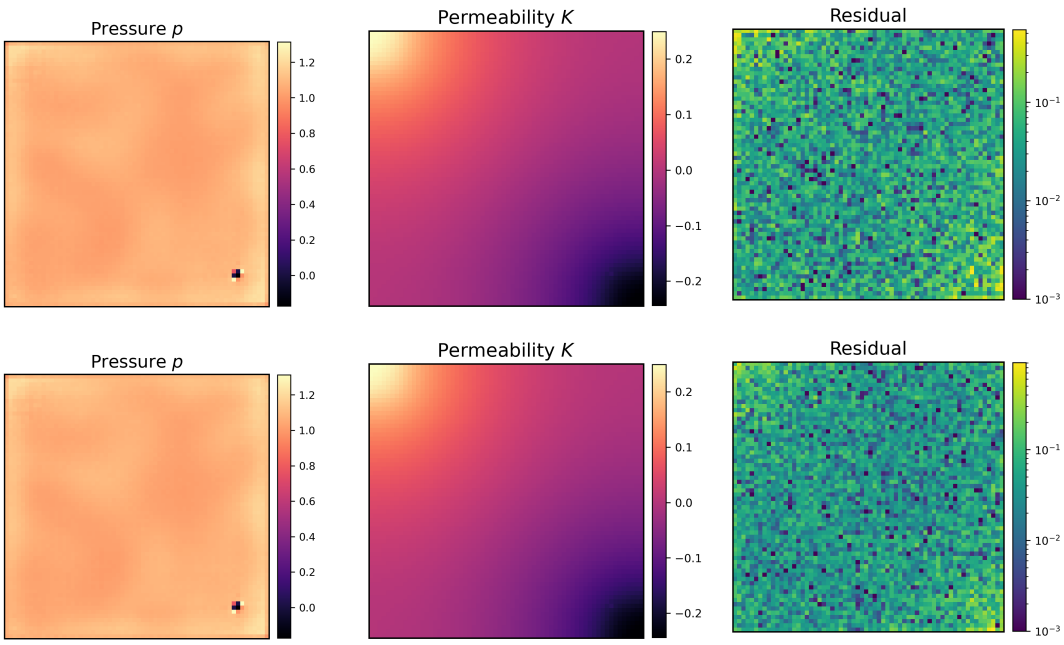


Figure 11: Failure cases without feature-space optimization on Darcy flow. Each row shows one generated sample, with columns corresponding to pressure p, permeability K, and residual. Without a feature encoder, optimization fails to produce physically meaningful results.

E.3 Kolmogorov Flow

Dataset and constraints. Kolmogorov flow is a standard benchmark for two-dimensional incompressible turbulence, where the dynamics are driven by a spatially periodic forcing. We follow the dataset construction in Baldan et al. (2026), where velocity fields are defined on a 128×128 periodic domain and conditioned on the Reynolds number, which determines the flow regime. The data are generated using a spectral solver based on the vorticity–streamfunction formulation of the incompressible Navier–Stokes equations. Specifically, the vorticity field is evolved in time, and the velocity field $U = (u, v)$ is recovered from the stream function, while the pressure is obtained by solving a Poisson equation. The simulations are performed for Reynolds numbers in the range $[100, 500]$, with 32 distinct values used for training and 16 unseen values reserved for evaluation. For each Reynolds number, multiple trajectories are simulated from perturbed initial conditions, and snapshots are collected after the flow reaches a statistically steady regime, resulting in 1024 samples per condition.

The governing physical constraint is incompressibility, corresponding to mass conservation:

$$\nabla \cdot U(x) = \partial_x u(x) + \partial_y v(x) = 0, \qquad x \in \Omega. \tag{61}$$

Because the domain is periodic, the velocity field satisfies periodic boundary conditions in both spatial directions. We therefore use circular padding in the neural-network convolutions, so the boundary conditions are respected without additional enforcement. For consistency with the data-generation process, the residual is computed using the same discretization scheme as the spectral solver, avoiding numerical discrepancies between training and evaluation.

This task is conditional: the Reynolds number specifies the operating regime, and the model is required to generate velocity fields that match both the conditional distribution and the incompressibility constraint.

Metrics. The physical residual error is defined by the divergence violation:

$$\mathrm{RE} = \mathbb{E}_{x \in \Omega}[(\nabla \cdot U(x))^2]. \tag{62}$$

Table 12: Architecture and training hyperparameters of Darcy Flow.

Hyperparameter	Value
Architecture hyperparameters	
In-, output channels	2, 2
ResNet blocks per down-/upsampling pass	2
ResNet block normalization	Group Normalization, 8 groups
ResNet block activation function	SiLU
Attention block normalization	Layer Normalization
Feature map resolutions	$64\times64 \to 32\times32 \to 16\times16 \to 8\times8$
Latent dimensions	$32 \to 64 \to 128 \to 256$
Attention head dimension	32
Number of attention heads	8
Parameters (M)	9.85
Conditioning size	None
Training hyperparameters	
Batch size (Darcy flow)	64
Iterations (Darcy flow)	400k
Learning rate	$5e^{-5}$
Optimization algorithm	Adam
EMA decay	0.9999
Dropout	none
Weight Decay	0

We use WD and JS to measure distributional fidelity as in Darcy flow (subsection E.2). In addition, we report the MSE of the conditional mean fields (MMSE) and the MSE of the conditional standard-deviation fields (SMSE):

$$\begin{aligned} \mathrm{MMSE} &= \mathbb{E}_c \|\mathbb{E}[U|c] - \mathbb{E}[U_{\mathrm{data}}|c]\|_2^2, \\ \mathrm{SMSE} &= \mathbb{E}_c \|\mathrm{Std}(U|c) - \mathrm{Std}(U_{\mathrm{data}}|c)\|_2^2. \end{aligned} \tag{63}$$

MMSE evaluates whether the generated samples match the conditional mean flow, while SMSE evaluates whether they reproduce the conditional fluctuation level. We also report the number of function evaluations (NFE) and inference time (IT) to measure sampling efficiency.

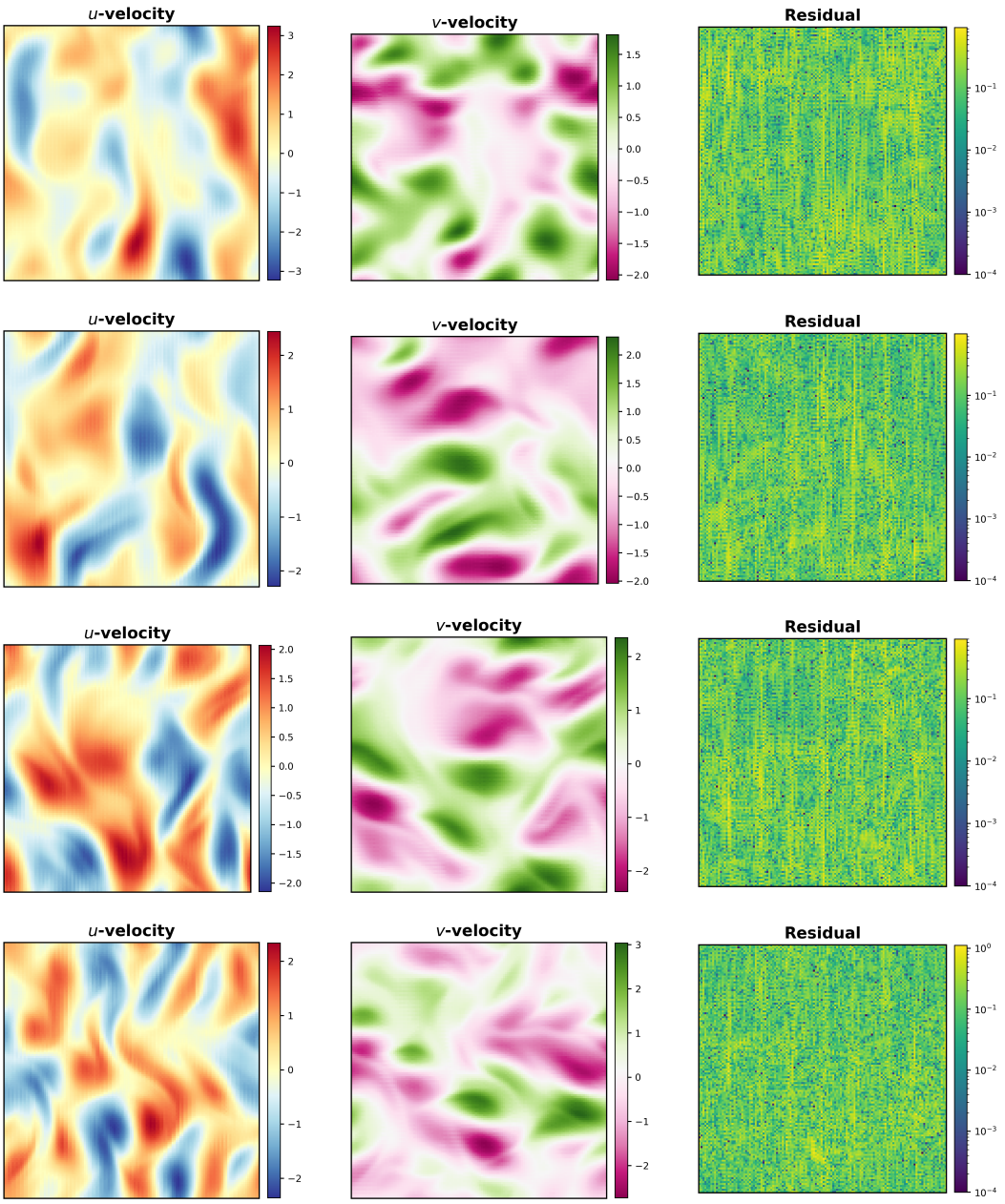


Figure 12: Generated Kolmogorov flow samples. Each row shows one sample, with columns corresponding to u-velocity, v-velocity, and divergence residual. From top to bottom, the Reynolds number increases.

Model and training. We use the same architecture as in Bastek et al. (2024) without additional architectural modifications. For experiments with feature-space stabilization, the encoder is used only during training to compute the data target, while generation is still performed in the original data space. The training details of the feature encoder are provided in subsection D.3, and the architecture and training configuration of the main model after encoder pretraining are summarized in Table 13.

Visualization Results. Qualitative results are shown in Figure 12. Our method produces coherent velocity fields while maintaining low divergence residuals.

Table 13: Architecture and training hyperparameters of Kolmogorov flow.

Hyperparameter	Value
Architecture hyperparameters	
In-, output channels	2, 2
Global conditioning dimension	1
ResNet blocks per down-/upsampling pass	2
ResNet block normalization	Group Normalization, 8 groups
ResNet block activation function	SiLU
Attention block normalization	Layer Normalization
Feature map resolutions	$128\times128 \to 64\times64 \to 32\times32 \to 16\times16 \to 8\times8$
Latent dimensions	$32 \to 64 \to 128 \to 256 \to 512$
Attention head dimension	32
Number of attention heads	8
Parameters (M)	36.2
Conditioning variable	1
Sigmoid output	False
Training hyperparameters	
Batch size (K-flow)	64
Iterations (K-flow)	400k
Learning rate	5×10^{-5}
Optimization algorithm	Adam
EMA decay	0.9999
Dropout	none
Weight Decay	0

E.4 Dynamic Stall

Dataset and constraints. Dynamic stall is a conditional generation benchmark for unsteady compressible flow around a sinusoidally pitching NACA0012 airfoil. We follow the dataset construction of Baldan et al. (2026), where the data are generated by solving the two-dimensional unsteady compressible Reynolds-averaged Navier-Stokes (RANS) equations on an O-type mesh around the airfoil. The simulations are performed with ANSYS Fluent ANSYS (2024) using an implicit finite-volume solver, and the resulting flow fields are downsampled to a 128×128 grid. The airfoil motion is prescribed by a sinusoidal pitching law, and each simulation is run until periodic convergence is reached, after which flow snapshots are collected over a full pitching cycle (see Baldan et al. (2026) for further details). Each sample contains six predicted fields, including absolute pressure P, temperature T, density ρ, skin friction τ_w, and tangential velocity gradients along the airfoil surface. The model is conditioned on four operating parameters: the free-stream Mach number, the mean angle of attack, the pitching amplitude, and the reduced frequency. Following Baldan et al. (2026), training and testing configurations are sampled from this four-dimensional design space, with additional perturbations applied to each nominal configuration. The training set contains 128 base configurations, each perturbed 32 times, and the test set contains 16 unseen configurations.

The physical constraints are imposed through two analytical point-wise residuals following Baldan et al. (2026). The first residual enforces the ideal gas relation:

$$\mathcal{R}_{\text{ig}} = P - \rho R T, \tag{64}$$

while the second constrains the skin friction using the temperature-dependent viscosity from Sutherland's law:

$$\begin{aligned} \mathcal{R}_\tau &= \tau_w - \mu(T)\sqrt{(\partial_x u)^2 + (\partial_y u)^2}, \\ \mu(T) &= \mu_0 \frac{T_0 + S}{T + S} |(\frac{T}{T_0}|)^{3/2}. \end{aligned} \tag{65}$$

Here, R is the gas constant, and μ_0, T_0, and S are the reference viscosity, reference temperature, and Sutherland constant. These constraints couple the generated thermodynamic variables and near-wall quantities.

Table 14: Architecture and training hyperparameters of dynamic stall.

Hyperparameter	Value
Architecture hyperparameters	
In-, output channels	6, 6
Global conditioning dimension	4
ResNet blocks per down-/upsampling pass	2
ResNet block normalization	Group Normalization, 6 groups
ResNet block activation function	SiLU
Attention block normalization	Layer Normalization
Feature map resolutions	$128\times128 \to 64\times64 \to 32\times32 \to 16\times16$
Latent dimensions	$12 \to 24 \to 48 \to 96$
Attention head dimension	32
Number of attention heads	8
Parameters (M)	1.9
Conditioning variables	4
Sigmoid output	False
Training hyperparameters	
Batch size (dynamic stall)	64
Iterations (dynamic stall)	400k
Learning rate	5×10^{-5}
Optimization algorithm	Adam
EMA decay	0.9999
Dropout	none
Weight Decay	0

Metrics. The physical residual error combines the two point-wise residuals:

$$\mathrm{RE} = \mathbb{E}[\mathcal{R}_{\mathrm{ig}}^2 + \mathcal{R}_{\tau}^2]. \tag{66}$$

We use the same distributional and statistical metrics as in Kolmogorov flow, including WD, JS, MMSE, and SMSE. These metrics evaluate whether the generated samples match both the physical constraints and the conditional statistics of the target flow fields. NFE and IT are also reported to compare sampling efficiency.

Model and training. We use the same basic architecture as in Bastek et al. (2024) without additional architectural modifications. The model is conditioned on the operating and pitching parameters of the dynamic-stall case. For experiments with feature-space stabilization, the encoder is used only during training to compute the data target, while generation is still performed in the original data space. The architecture and training configuration are summarized in Table 14.

Visualization Results. Qualitative results are shown in Figure 13. Our method generates coherent dynamic-stall fields across all predicted quantities.

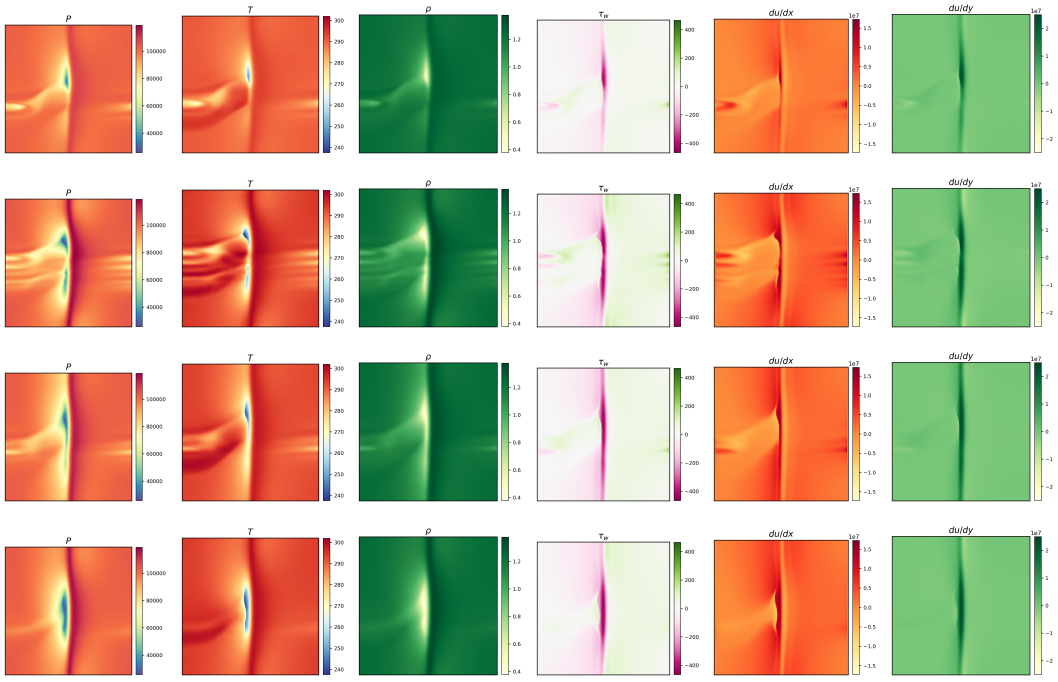


Figure 13: Generated dynamic-stall samples. Each row corresponds to one generated sample, and the columns show pressure P, temperature T, density ρ, wall shear stress τ_w, and tangential velocity gradients $\partial_x u$ and $\partial_y u$, respectively.

E.5 Topology Optimization

Dataset and constraints. Topology optimization is a conditional inverse-design benchmark, where the objective is to generate a material distribution that achieves high structural stiffness under prescribed loading and boundary conditions. We follow the setup of Bastek et al. (2024). Each sample is defined on a 65×65 grid representing a discretized design domain. The model is conditioned on the external load field $\mathbf{f} = (f_x, f_y)$, boundary conditions, auxiliary physical fields (e.g., strain energy and von Mises stress), and a target volume constraint $V_{\max}$. These conditioning fields are concatenated with noise and provided as input to the network, which predicts both the displacement field $u = (u_1, u_2)$ and the material density field ρ.

The problem corresponds to minimum-compliance topology optimization under linear elasticity Bastek et al. (2024). The objective is to minimize the total elastic energy

$$\min_{\rho} \int_{\Omega} \frac{1}{2}\, \sigma(\xi) : \varepsilon(\xi)\, d\Omega, \tag{67}$$

subject to the box constraint $\rho(\xi) \in [0, 1]$, the volume constraint $\int_\Omega \rho(\xi)\, d\Omega \le V_{\max}$, and the equilibrium condition $\nabla \cdot \sigma(\xi) + \mathbf{f}(\xi) = 0$. Dirichlet boundary conditions are imposed on fixed supports, while Neumann boundary conditions specify the applied loads. The stress and strain fields are related through a density-dependent constitutive model $\sigma = \mathbb{C}(\rho) : \varepsilon(u)$, where $\varepsilon = \frac{1}{2}(\nabla u + \nabla u^\top)$. Following Bastek et al. (2024), the material stiffness tensor is parameterized using the SIMP formulation, $\mathbb{C}(\rho) = \rho^p \mathbb{C}_0$ with $p > 1$, which encourages near-binary material distributions. The displacement field u is discretized on the underlying grid, leading to a finite-dimensional representation U. Under a standard finite-element formulation, the equilibrium equation is converted into a linear system

$$K(\rho)U = F, \tag{68}$$

where $K(\rho)$ is the global stiffness matrix assembled from element-wise contributions, U denotes the nodal displacement vector of u, and F is the external force vector of f. Consistent with Bastek et al. (2024), we evaluate the physical residual by assembling the FEM stiffness system directly on the grid

representation, rather than relying on finite-difference approximations, to ensure consistency with the data generation process.

The dataset is defined on a 65×65 nodal grid, which we interpolate to a 64×64 grid for compatibility with the convolutional architecture, following Bastek et al. (2024). We adopt the same data split as Bastek et al. (2024), where the test set is divided into in-distribution and out-of-distribution cases. The in-distribution setting contains samples with loading and boundary configurations similar to those seen during training, while the out-of-distribution setting introduces unseen configurations to evaluate generalization under novel physical conditions.

Metrics. We evaluate the generated solutions from three complementary perspectives: physical consistency, structural optimality, and volume-constraint satisfaction. For physical consistency, we measure the violation of the discretized equilibrium equation using the residual mean absolute error. Specifically, for each sample we compute the ℓ_1 norm of the FEM residual and report its median over the test set:

$$R_{\mathrm{MAE}} = \mathrm{median}|(\frac{1}{N}|\|K(\rho)\,U - F\|_1|), \tag{69}$$

where $K(\rho)$ denotes the global stiffness matrix assembled from the predicted density field, U is the corresponding nodal displacement obtained from the model, F is the external force vector, and N is the number of grid points. This metric quantifies how well the generated fields satisfy mechanical equilibrium. To assess structural performance, we report the median compliance error (CE). Following standard practice Bastek et al. (2024), the predicted density field is first binarized, and the corresponding displacement is recomputed by solving the FEM system. The compliance of the generated structure is then defined as

$$C_{\mathrm{gen}} = U_{\mathrm{gen}}^{\top} F, \tag{70}$$

and compared against the reference compliance C_{ref}, yielding

$$\mathrm{CE} = \frac{C_{\mathrm{gen}} - C_{\mathrm{ref}}}{C_{\mathrm{ref}}}. \tag{71}$$

Finally, we evaluate satisfaction of the volume constraint using the volume fraction error (VFE). After binarization, the average density is compared with the target volume fraction v_f:

$$\mathrm{VFE} = \frac{||\bar{\rho}_{\mathrm{bin}} - v_f||}{v_f}, \tag{72}$$

where $\bar{\rho}_{\mathrm{bin}}$ denotes the mean value of the binarized density field. Lower values of R_{MAE}, CE, and VFE indicate better physical consistency, structural optimality, and constraint satisfaction, respectively.

Model and training. We use the same basic architecture as in Bastek et al. (2024) without additional architectural modifications. The model is conditioned on the loading, boundary-condition, and volume-constraint information of the topology-optimization problem. For experiments with feature-space stabilization, the encoder is used only during training to compute the data target, while generation is still performed in the original data space. The architecture and training configuration are summarized in Table 15.

Visualization Results. Qualitative results are shown in Figure 14. Our method generates feasible topology designs while maintaining low equilibrium residuals.

Table 15: Architecture and training hyperparameters of topology optimization.

Hyperparameter	Value
Architecture hyperparameters	
In-, output channels	10, 3
ResNet blocks per down-/upsampling pass	2
ResNet block normalization	Group Normalization, 8 groups
ResNet block activation function	SiLU
Attention block normalization	Layer Normalization
Feature map resolutions	$64\times64 \to 32\times32 \to 16\times16 \to 8\times8$
Latent dimensions	$128 \to 256 \to 512 \to 1024$
Attention head dimension	32
Number of attention heads	8
Parameters (M)	131.2
Sigmoid output	True for the last channel
Training hyperparameters	
Batch size (topology optimization)	64
Iterations (topology optimization)	600k
Learning rate	5×10^{-5}
Optimization algorithm	Adam
EMA decay	0.9999
Dropout	none
Weight Decay	0

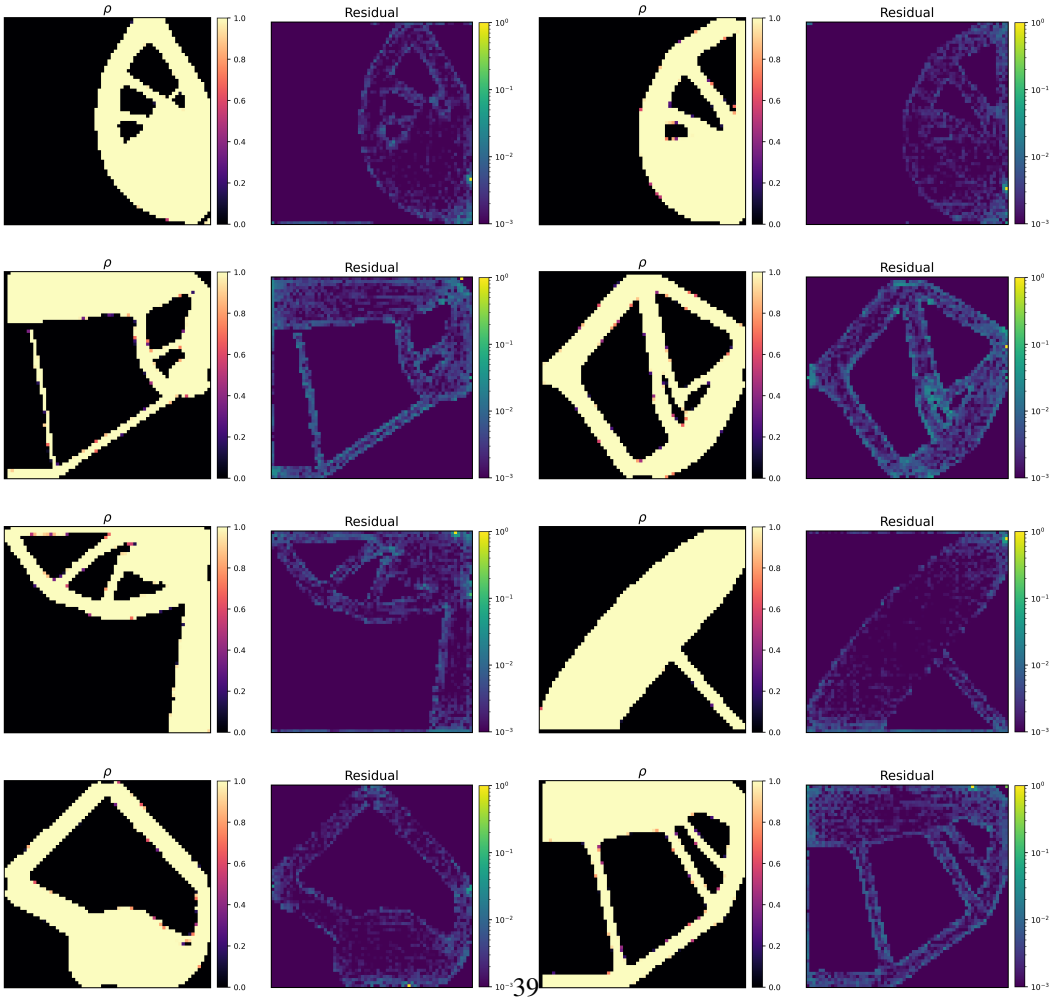


Figure 14: Generated topology-optimization samples. Columns 1 and 3 show the generated density fields ρ, while columns 2 and 4 show the corresponding equilibrium residuals.

F Additional Theoretical Background

F.1 Flow Map Learning

As discussed in subsection 2.2, flow-map learning aims to directly learn the flow map $\psi_{t\to r}$, so that one-step generation can be performed via $x_1 = \psi^{\theta}_{0\to 1}(x_0)$, where $x_0 \sim p_0$. Previously, similar to multi-step generative models, there is no easily computable closed-form target flow map $\psi_{t\to r}(x_0)$ available for direct supervision. Therefore, one cannot simply train the model with a direct objective of the form $\|\psi^{\theta}_{t\to r}(x_0) - \psi_{t\to r}(x_0)\|^2$.(In this work, we address this difficulty through a long-short flow-map formulation.)

Instead, flow-map learning must rely on regression-based surrogate objectives, much like multi-step methods such as Flow Matching. In Flow Matching, this is achieved by constructing a conditional path $p_t(x_t|x_1)$ and using the corresponding conditional velocity $u_t(x_t|x_1)$ to regress the marginal velocity $\mathcal{L}(\theta) = \mathbb{E}_{t,x_1\sim p_1, x_t\sim p_{t|1}(x_t|x_1)}[\|u^{\theta}(x_t,t) - u_t(x_t|x_1)\|^2]$. The difficulty for flow maps, however, is that unlike instantaneous velocity fields $u_t(x_t)$, they do not admit a self-consistent conditional counterpart Li et al. (2026).

Theorem F.1 ((Non-existence of conditional flow maps; Theorem 4.1 of Li et al. (2026)).)**.** *There exists no conditional flow map $\psi_{t\to r}(x|x_1)$ that simultaneously (i) is consistent with the conditional velocity $u(x|x_1)$ under Equation 3, and (ii) satisfies the consistency relation $\psi_{t\to r}(x) = \mathbb{E}_{x_1\sim p_t(x_1|x)}[\psi_{t\to r}(x|x_1)]$ with the marginal flow map. As a result, a self-consistent conditional cumulative field does not exist.*

Proof. Let the mappings $\psi_{t\to r}(x)$ defined by (1) and (2) be denoted by $\psi^{(1)}_{t\to r}(x)$ and $\psi^{(2)}_{t\to r}(x)$, respectively. Specifically, $\psi^{(1)}_{t\to r}(x) = \psi_{0\to r}(\psi^{-1}_{0\to t}(x))$ and $\psi^{(2)}_{t\to r}(x) = \mathbb{E}_{x_1\sim p_t(x_1|x)}[\psi_{t\to r}(x|x_1)]$.

It suffices to show that $\psi^{(1)}_{t\to r}(x) \neq \psi^{(2)}_{t\to r}(x)$. To do so, it is enough to prove that $\frac{d}{dr}\psi^{(1)}_{t\to r}(x) \neq \frac{d}{dr}\psi^{(2)}_{t\to r}(x)$. For $\psi^{(2)}_{t\to r}(x)$, we have

$$\begin{aligned}
\frac{d}{dr}\psi^{(2)}_{t\to r}(x) &= \frac{d}{dr}\int_{x_1} \psi_{t\to r}(x|x_1)\, p_t(x_1|x)\, dx_1 \\
&= \int_{x_1} \frac{d}{dr}\psi_{t\to r}(x|x_1)\, p_t(x_1|x)\, dx_1 \\
&= \int_{x_1} u_r(\psi_{t\to r}(x|x_1)|x_1)\, \frac{p_{\text{data}}(x_1)p_t(x|x_1)}{p_t(x)}\, dx_1 \\
&= \int_{x_1} u_r(\psi_{t\to r}(x|x_1)|x_1)\, p_{\text{data}}(x_1)\, dx_1 \\
&= \int_{x_1} (x_1 - x)\, p_{\text{data}}(x_1)\, dx_1 \\
&= \mathbb{E}_{x_1\sim p_{\text{data}}}[x_1] - x.
\end{aligned} \tag{73}$$

Consequently, if $\psi^{(1)}_{t\to r}(x) = \psi^{(2)}_{t\to r}(x)$, then we must have

$$\frac{d}{dr}\psi_{0\to r}(x) = \mathbb{E}_{x_1\sim p_{\text{data}}}[x_1] - x \tag{74}$$

which implies $\psi_{0\to r}(x) = (\mathbb{E}_{x_1\sim p_{\text{data}}}[x_1] - x)r + x$. In particular, at $r = 1$, this yields

$$p_{\text{data}} = (\psi_{0\to 1})_\sharp p_0 = \delta_{\mathbb{E}_{x_1\sim p_{\text{data}}}[x_1]}, \tag{75}$$

where δ denotes the Dirac distribution concentrated at a single point. This is a contradiction unless the data distribution is degenerate. Therefore, such a conditional flow map cannot exist. □

Because a self-consistent conditional flow map does not exist, flow-map learning is typically carried out through regression objectives built around trajectory consistency (Proposition Proposition 2.1). In particular, one may define the trajectory-consistency loss based on Proposition Proposition 2.1 as $\mathcal{L}^{\text{TC}}(\theta) = \mathbb{E}_{t,s,r,x_t\sim p_t}[\|\psi^{\theta}_{t\to r}(x_t) - \psi^{\theta}_{s\to r}(\psi^{\theta}_{t\to s}(x_t))\|_2^2$. However, $\mathcal{L}^{\text{TC}}$ does not contain dataset-level supervision, since every term in the loss is produced by the model itself. Therefore, $\mathcal{L}^{\text{TC}}$ can

only enforce internal consistency of the learned flow map, but cannot guarantee that the learned map generates the desired target distribution p_1.

To address this issue, existing flow-map learning methods mainly fall into two categories: *progressive extension* methods and *derivative-based* methods.

- **Progressive extension** (ShortCut Frans et al. (2025), PSD of FMM Boffi et al. (2025), Split-MeanFlow Guo et al. (2025)): these methods extend instantaneous dynamics to longer flow maps through consistency constraints. For example, ShortCut first learns the instantaneous velocity by regression loss with supervision from the dataset

$$\mathcal{L}(\theta) = \mathbb{E}_{x_1\sim p_1, x\sim p_{t|1}(\cdot|x_1)}[\|u^\theta_{t\to t+d}(x) - u_t(x|x_1)\|^2], \qquad u_t(x|x_1) = \frac{x_1 - x}{1-t}, \quad x_1 \in \mathcal{D}, \tag{76}$$

  and then extends to longer intervals through consistency relations such as

$$\mathcal{L}(\theta) = \mathbb{E}_{x_t\sim p_t}[\|u^\theta_{t\to t+2d}(x_t) - \frac{1}{2}(u^\theta_{t\to t+d}(x_t) + u^\theta_{t+d\to t+2d}(x_{t+d}))\|^2] \tag{77}$$

  where $x_{t+d} = d\,u_{t\to t+d}(x_t) + x_t$, and d is sampled uniformly at random from $\{\frac{1}{128}, \frac{1}{64}, \ldots, \frac{1}{2}\}$. As a result, the long-range map receives no direct supervision from the dataset $\mathcal{D}$; instead, it is learned recursively from shorter model outputs.

- **Derivative-based methods** (MeanFlow Geng et al. (2025), Consistency Models Song et al. (2023), ESD and LSD of FMM Boffi et al. (2025)): these methods relate the flow map $\psi_{t\to r}$ to the instantaneous velocity field $u_t = \frac{d}{dr}\psi_{t\to r}|_{r=t}$ through the continuous equations $\frac{d}{dt}\psi_{t\to r}(x) + u_t(x)\nabla\psi_{t\to r}(x) = 0$ or $\frac{d}{dr}\psi_{t\to r}(x) = u_r(\psi_{t\to r}(x))$ implied by trajectory consistency (Proposition Proposition 2.1), obtained by appropriate reformulations and by taking the limits $s \to t$ and $s \to r$, respectively. This connection allows them to supervise long-range flow maps $\psi_{t\to r}$ more directly using data-dependent conditional velocities $u_t(x|x_1)$. For example, MeanFlow uses an objective of the form

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim p_t}[|u^\theta_{t\to r}(x) - \mathrm{sg}[u_t(x|x_1) + (r-t)(u_t(x|x_1)\partial_x u^\theta_{t\to r}(x) + \partial_t u^\theta_{t\to r}(x))]|^2] \tag{78}$$

  where $u_t(x|x_1) = \frac{x_1 - x}{1-t}, x_1 \in \mathcal{D}$. This provides more direct supervision than progressive extension methods. However, it also leads to a more complicated regression target and typically requires Jacobian-vector products (JVPs), which can significantly increase memory usage, computational cost, and optimization instability.

Other methods, such as AlphaFlow Zhang et al. (2025), can be viewed as hybrids of these two categories. Overall, compared with multi-step generative models that only need to learn instantaneous fields, the regression objectives for flow-map learning are structurally more complicated. This further increases the difficulty of incorporating physical constraints into one-step or few-step flow-map learning.

F.2 Closed-Form Solution

In addition to optimizing indirect regression losses, prior work Bertrand et al. (2025) has also derived a closed-form expression for the Flow Matching velocity field $u_t(x) = \frac{\mathbb{E}_{x_1\sim p_1}[\frac{x_1 - x}{1-t} k_t(x_1, x)]}{\mathbb{E}_{x_1\sim p_1}[k_t(x_1, x)]}$. However, this closed form is mainly used for theoretical analysis rather than practical training. The main reason is that it involves the computation of a kernel $k_t(y, x) = e^{-\frac{\|ty - x\|^2}{2(1-t)^2}}$, whose value depends on the time-dependent coefficient $\frac{1}{(1-t)^2}$. When t is far from the endpoint 1, this coefficient makes the kernel relatively flat, as illustrated in Figure 15. As a result, accurately estimating the expectation $\mathbb{E}_{x_1\sim p_1}[\cdot k_t(x_1, x)]$ inside the kernel-weighted velocity field u_t requires a large number of samples, making direct training with this closed-form velocity ineffective.

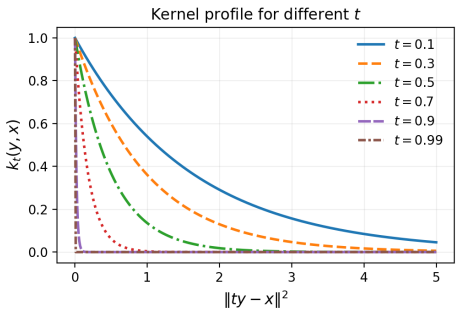


Figure 15: Kernel profiles for different values of t.

In our method, the long-short flow-map decomposition allows us to use the closed-form Flow Matching dynamics only in the near-endpoint regime $t \to 1$. In this regime, the kernel becomes sufficiently sharp, so estimating the expectation $\mathbb{E}_{x_1 \sim p_1}[\cdot\, k_t(x_1, x)]$ does not require a large number of samples, as discussed in subsection 3.3. Moreover, to unify the data-driven effect with the physics-driven effect, we do not use the closed-form velocity directly. Instead, we adopt the potential-based formulation in Theorem 3.1, which also avoids the singularity of the closed-form velocity field at $t = 1$.

G Additional Darcy Comparisons and Experimental Settings

G.1 Conflict-Free One-Step Comparisons

We compare PI-Consistency (implementing sCM), PI-MeanFlow, and PI-Shortcut on Darcy. Each retains its original training objective and procedure, with the same terminal physics loss added under matched architecture, data, batch size, residual, and training budget. Table 16 reports results with and without the conflict-free (CF) optimization strategy from PBFM Baldan et al. (2026). This control tests whether resolving data–physics gradient conflicts alone is sufficient to close the performance gap. CF updates improve the physical residuals of these baselines but do not recover the overall data–physics performance of PI-LSFM, which uses no CF update. The remaining gap supports the benefit of our direct data-and-physics objective beyond the gains obtained by conflict-free optimization of the tested indirect objectives.

Table 16: One-step Darcy baselines with and without conflict-free (CF) updates.

Method	CF	RE ↓	WD·10^2 ↓	JS·10^1 ↓
PI-Consistency	No	0.240	3.889	5.023
PI-Consistency	Yes	0.152	3.017	3.933
PI-MeanFlow	No	4603.573	4.243	5.457
PI-MeanFlow	Yes	4.973	4.253	5.212
PI-Shortcut	No	0.435	2.336	2.518
PI-Shortcut	Yes	0.276	2.184	2.363
PI-LSFM	No	0.055	0.108	0.171

G.2 Three-Seed Darcy Check

Table 17 reports three additional independent Darcy runs trained for 100K steps each. These runs provide a stability check at 100K steps, rather than three fully converged 400K-step runs. Our main benchmark results use the fixed seed 42 without seed selection.

Table 17: Three-seed Darcy stability check at 100K steps.

Seed	RE ↓	WD·10^2 ↓	JS·10^1 ↓
43	0.132	0.269	0.241
123	0.116	0.215	0.210
2026	0.127	0.308	0.259
Mean $\pm$ std	0.125 ± 0.008	0.264 ± 0.047	0.237 ± 0.025

G.3 Batch-Size Sensitivity

Table 18 reports the Darcy batch-size sweep, with each setting trained for 100K steps. Batch sizes 32 and 64 retain useful data–physics performance, while the results deteriorate at 16 and 8. The sweep evaluates unconditional Darcy generation.

Table 18: Darcy batch-size sensitivity after 100K training steps per setting.

Batch size	RE ↓	WD·10^2 ↓	JS·10^1 ↓
64 (default)	0.117	0.213	0.216
32	0.193	0.503	0.441
16	13.715	7.572	2.771
8	5.971	7.395	2.570

G.4 Default Settings and Selection Protocol

The data and physics discretizations are selected through the Darcy design studies and retained across all four benchmarks. The bandwidth $\epsilon = 0.1$ is shared across tasks. The Darcy physics weight comes from its design-study sweep; the other benchmarks reuse prior-work settings Baldan et al. (2026) without a new per-task search. Table 19 summarizes the relative physics weights $\bar{\lambda}$, encoders, and discretizations. Darcy and Kolmogorov use an MAE encoder; dynamic stall and topology optimization train without an encoder.

Table 19: Per-benchmark default settings used in the reported experiments.

Benchmark	$\bar{\lambda}$	ϵ	Encoder	Data	Physics
Darcy	3.0	0.1	MAE	TR	FE
Kolmogorov	1.0	0.1	MAE	TR	FE
Dynamic stall	1.0	0.1	None	TR	FE
Topology optimization	0.01	0.1	None	TR	FE