Bridging Neural Operator and Flow Matching for a Generative PDE Foundation Model

Zituo Chen
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
zituo@mit.edu

Sili Deng*
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
silideng@mit.edu

Abstract

Pretraining on large-scale collections of PDE-governed spatiotemporal trajectories has recently shown promise for building generalizable models of dynamical systems. Yet, most existing PDE foundation models rely on deterministic Transformer architectures, which demand substantial computational resources and lack generative flexibility. In contrast, generative models can capture uncertainty, making them well-suited for probabilistic forecasting, data assimilation, and scientific design. In this work, we introduce a generative PDE foundation model that bridges neural operator learning with flow matching. By jointly sampling the noise level and the physical timestep between adjacent states, the model learns a unified velocity field that transports a noisy current state toward its clean successor. Alongside the core framework, we introduce novel architectural strategies that achieve up to 15× greater computational efficiency than full-length diffusion models, enabling large-scale pretraining at substantially reduced cost. Our framework combines autoregressive Transformers and latent diffusion in a two-stage training pipeline, yielding scalable, accurate, and extensible generative modeling of PDE systems. We curate a training corpus of $\sim$2M trajectories across 12 distinct PDE families and release a suite of pretrained autoencoders and generative latent models of varying parameter scales. For downstream evaluation, we benchmark on previously unseen Kolmogorov turbulence with few-shot adaptation, and show long-term rollout stability of our model compared to its deterministic counterparts.

1 Introduction

Generative models can capture uncertainty through sampling, which is vital for scientific and engineering applications where forecasts [37, 23], machine and material design [46, 50], and safety margins [7, 18] depend on an ensemble of predictions rather than single point predictions. PDE foundation models [33, 5, 49] are large, pre-trained neural operators that learn generalizable representations of spatiotemporal dynamics, enabling zero-shot prediction, control, and design across diverse physical systems. However, most foundation models for PDEs have emphasized deterministic mapping from the current state to future states. This creates a gap between the application need and the capability of current large pretrained models.

This gap motivates a fundamental question about regression vs. sampling approaches in physical dynamics (e.g., generative ensemble-regression contrasts point-wise regression [47]): when should we learn a deterministic operator, and when must we sample from a conditional distribution over future fields? More importantly, can we unify these regimes in a single scalable framework that

*Corresponding Author

Accepted as poster in the AI for Science workshop (NeurIPS 2025).

offers the speed of neural operators while retaining the fidelity and extensive applications of modern generative models?

Training large-scale generative models for PDEs is challenging on two fronts, efficiency and data source. High-resolution fields are expensive to denoise or transport step-by-step; naively applying diffusion or flow-based methods in pixel space leads to prohibitive memory and computation cost. Meanwhile, scientific data are often multi-fidelity, heterogeneous across systems, and scarce at high accuracy; a practical foundation model must leverage broad-but-imperfect corpora while remaining physically plausible and stable in long-term rollouts.

Hence, we propose a latent diffusion-based generative PDE foundation model that unifies deterministic operators and stochastic flows within one training and inference stack. Our framework is efficient and scalable, while being capable to accommodate diverse types of dynamical systems.

Our approach has two components: a Pretrained Physics Variational Autoencoder (P2VAE) and a Flow Marching Transformer (FMT). P2VAE compresses static physical field snapshots into a compact latent grid (e.g., 16×16) to slash the cost of generative training and inference, and is trained across heterogeneous PDE datasets. FMT introduces a flow marching algorithm that bridges deterministic neural operator and stochastic flow matching through a bridge parameter k. For $k = 1$, FMT behaves like a neural operator; for $k = 0$, it reduces to a flow-matching–style stochastic sampler. In between, it learns transport on mixtures that combine data-driven drift with controlled stochasticity. We train the latent velocity field with a numerically stable frame-interpolation objective:

$$\|(1-t)\mathbf{g}(\mathbf{x}_t^k, t) - (\mathbf{x}_1 - \mathbf{x}_t^k)\|^2, \tag{1}$$

which follows directly from the continuity equation for the induced interpolation path. To support long-horizon rollouts, we introduce a diffusion forcing scheme that adaptively injects small stochastic increments during autoregressive prediction, mitigating error accumulation without sacrificing stability. Finally, a latent temporal pyramid executes coarse-to-fine transport, cutting training and inference cost while improving long-range consistency.

The key contributions of this paper are summarized as follow:

- We propose a latent generative PDE foundation model that unifies deterministic and stochastic modeling through a single transport field trained with a principled, well-conditioned regression objective rooted in flow matching.
- We design a novel flow marching algorithm equipped with diffusion forcing scheme and latent temporal pyramids that allows large-scale pretraining and stable long-term rollouts.
- We sort and provide a heterogeneous dataset across public PDE datasets including FNO-v, PDEArena, PDEBench and The Well, totaling up to 233 Gigabyte and consisting of 2.5 million trajectories.

2 Related works

2.1 Neural operator and PDE foundation models

Neural operators are data-driven models that build surrogate models for science and engineering applications. Existing methods include but not limited to FNO [25], DeepONet [31], PINO [26], OFormer [24], and UPT [1]. They excel at fast rollout and generalization to unseen inputs [20, 3, 21]. PDE foundation models based on Transformer architecture [45], such as ICON [48, 5], MPP [33], DPOT [13], PROSE [29, 41], and PITT [30], learn the PDE-governed spatiotemporal dynamics through large-scale pretraining across diverse spatiotemporal systems, and enable fast adaptation to new dynamics and in-context learning ability. However, they are mostly deterministic and lack the flexibility of generative modeling.

Diffusion-based generative models have been explored to solve PDE equations recently [15, 4, 35]. However, they follow the paradigm of generating new state out of pure noise conditioned on the previous states, which is different from our flow marching method; also, they target at a single dynamics, which is different from the concept of PDE foundation models.

2.2 Flow matching

Preliminaries Flow matching (FM) trains a time-dependent vector field by regressing to conditional velocities that deterministically transport white noise to data along a prescribed probability path [27, 28, 43]. Large-scale studies [10] further show FM can match or surpass diffusion on high-resolution image synthesis while retaining fast ODE sampling, motivating FM as a practical training principle for modern generative backbones.

Pyramidal flow matching Recent work proposes Pyramidal Flow Matching (PFM) [19], which reinterprets the denoising/transport trajectory as a multi-stage spatial pyramid, where only the final stage runs at full resolution while earlier stages operate coarser and are linked through a renoising technique to preserve continuity. This yields notable efficiency gains (especially for video) and pairs naturally with temporal pyramids for autoregressive history compression. These ideas directly inspire our latent temporal pyramid and coarse-to-fine training/inference strategy.

2.3 Diffusion forcing

Diffusion Forcing [6] blends autoregressive (AR) prediction with diffusion-style denoising: a causal next-token (or next-segment) model is trained to produce future content while simultaneously denoising a set of tokens with independent per-token noise levels. Compared to pure AR, diffusion forcing lets the model get access to partially noised data distributions; compared to pure diffusion, it preserves causal structure and AR efficiency. Interleaving causal prediction with one to three lightweight denoising refinements reduces exposure bias [2] while preserving autoregressive efficiency. In practice, it improves long-horizon stability, temporal coherence, and ensemble calibration with minimal latent-space overhead, which is suitable for the PDE condition propagation during an autoregressive generation process.

3 Methods

3.1 Flow marching

Let $(\mathbf{x}_0, \mathbf{x}_1) \sim \pi$ be consecutive states of a dynamical system. We synthesize intermediate training particles via a location-scale interpolation kernel in Fig. 1

$$\mathbf{x}_t^k = \mu_t + \sigma_t \mathbf{z}, \mu_t = t\mathbf{x}_1 + k(1-t)\mathbf{x}_0, \sigma_t = (1-t)(1-k), \mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I}), \tag{2}$$

with $t, k \sim \text{Unif}(0, 1)$.

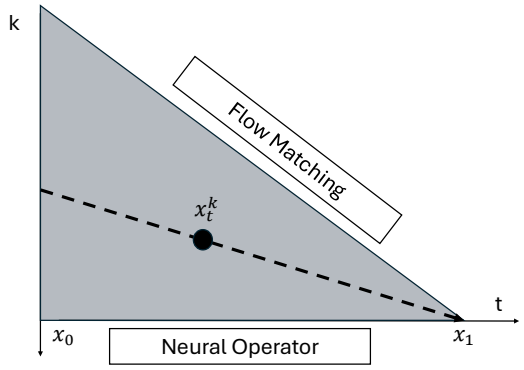

Figure 1: Location-scale interpolation kernel for flow marching

Conditionally,

$$q_t^k(\mathbf{x}_t^k | \mathbf{x}_0, \mathbf{x}_1, k) \sim \mathcal{N}(t\mathbf{x}_1 + k(1-t)\mathbf{x}_0, (1-t)^2(1-k)^2\boldsymbol{I}). \tag{3}$$

Two limits are informative: (i) $k = 0$ recovers a flow-matching kernel $\mathbf{x}_t^0 \sim \mathcal{N}(t\mathbf{x}_1, (1-t)^2\boldsymbol{I})$; (ii) $k = 1$ gives the deterministic neural operator interpolation $\mathbf{x}_t^1 = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$. Thus, k continuously bridges stochastic flow transport and deterministic operator regression.

Using the equivalent form

$$\mathbf{x}_t^k = \mathbf{x}_0 + t(\mathbf{x}_1 - \mathbf{x}_0) - (1-t)(1-k)(\mathbf{x}_0 - \mathbf{z}). \tag{4}$$

Differentiation gives the sample-wise velocity

$$\mathbf{u}_t^k = \frac{d}{dt}\mathbf{x}_t^k = (1-k)(\mathbf{x}_0 - \mathbf{z}) + \mathbf{x}_1 - \mathbf{x}_0 = \frac{\mathbf{x}_1 - \mathbf{x}_t^k}{1-t}. \tag{5}$$

Because the kernel is Gaussian local-scale, its conditional score at the sampled point $\mathbf{x} = \mathbf{x}_t^k$ is

$$\nabla_{\mathbf{x}} \log q_t^k(\mathbf{x}_t^k) = -\frac{\mathbf{x}_t^k - \mu_t}{\sigma_t^2} = -\frac{\mathbf{z}}{(1-t)(1-k)}. \tag{6}$$

We substitute Eq. 6 into Eq. 5, and get the score-velocity decomposition:

$$\mathbf{u}_t^k = (\mathbf{x}_1 - k\mathbf{x}_0) + (1-t)(1-k)^2 \nabla_{\mathbf{x}} \log q_t^k(\mathbf{x}_t^k). \tag{7}$$

Therefore, the transport velocity is a well-posed learnable target: its random part is aligned with an intrinsic geometric direction of the conditional density (the score), and its deterministic part is fixed by the pair $(\mathbf{x}_0, \mathbf{x}_1)$ and the bridge parameter k. Regressing a model $\mathbf{g}$ to $\mathbf{u}_t^k$ theoretically recovers the posterior-mean transporting field $\mathbf{g}^*$ (see A.1).

With the above justification, our setting trains a neural network approximation $\mathbf{g}$ by regressing to the oracle velocity:

$$\mathcal{R}_{\text{FM}}(\mathbf{g}) = \mathbb{E}_{(\mathbf{x}_0,\mathbf{x}_1),k,t}\left[\frac{1}{2}||\mathbf{g} - \mathbf{u}_t^k||^2\right]. \tag{8}$$

We notice that sampling both k and t to be the input of $\mathbf{g}$ is empirically slow to converge, and one of the most apparent failure mode of using k in the denoising process is that an intermediate denoised state would deviate from the original assigned bridge parameter k thus introduce an accumulating error. In practice, we adopted $\mathbf{u}_t^k = (\mathbf{x}_1 - \mathbf{x}_t^k)/(1-t)$ in Eq. 5 as the training objective, so that $\mathbf{x}_t^k$ and t are sufficient as inputs. This k-free objective can also be intuitively understood as the linear vector pointing to the end state $\mathbf{x}_1$ from the current state $\mathbf{x}_t^k$. This frame-interpolation view – "predict the missing bridge $(\mathbf{x}_1 - \mathbf{x}_t^k)$"– makes the training interface minimal: once $\mathbf{x}_t^k$ is constructed offline, the supervision depends only on $(\mathbf{x}_1 - \mathbf{x}_t^k)$.

The form is numerical stiff near $t \to 1$. We therefore precondition the target by $(1-t)$ and obtain the flow marching objective:

$$\mathcal{L}_{\text{FM}} = \frac{1}{2}\mathbb{E}\left[||(1-t)\mathbf{g}_\theta(\mathbf{x}_t^k, t) - (\mathbf{x}_1 - \mathbf{x}_t^k)||^2\right]. \tag{9}$$

Minimizers of $\mathcal{L}_{\text{FM}}$ correspond to minimizers of $\mathcal{R}_{\text{FM}}$ (up to the benign $(1-t)$ scaling), and the regression is well-conditioned at late times.

3.2 Conditional flow marching through diffusion forcing

There exists a variety of different dynamics in the training dataset. To accommodate these dynamics in one PDE foundation model, we introduce the conditional form of flow marching and design an approach to condition the dynamics through past states.

The conditional flow marching target could be derived from a conditional probability:

$$q_t(\mathbf{x}, \mathbf{h}) = \mathbb{E}_{\mathbf{h},(\mathbf{x}_0,\mathbf{x}_1),k}\left[q_t^k(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1, \mathbf{h})\right]. \tag{10}$$

Following the same derivation, we have the conditional flow marching objective:

$$\mathcal{L}_{\text{CFM}} = \frac{1}{2}\mathbb{E}\left[||(1-t)\mathbf{g}_\theta(\mathbf{x}_t^k, t, \mathbf{h}) - (\mathbf{x}_1 - \mathbf{x}_t^k)||^2\right]. \tag{11}$$

To derive the condition $\mathbf{h}$, we follow the style of diffusion forcing (DF) [6] to design a forcing scheme, which makes use of history states with different noise levels to induce the filtered PDE condition $\mathbf{h}_s$

(denote the physical timestep as $s = 1, 2, \dots T$). Specifically, a DF keeps and updates a compressed latent state $\mathbf{h}$ to be the condition at each step to inform the denoising process. We adopt a simplistic RNN parametrized by ϕ like original DF paper to evolve latent state $\mathbf{h}_s \sim p_\phi(\mathbf{h}_s|\mathbf{h}_{s-1}, \mathbf{x}^{k_s}_{s,t_s}, t_s)$, because governing equation and coefficients of PDE dynamics are usually kept the same as time evolves.

In conclusion, the training objective for FMT is

$$\mathcal{L}_{\text{CFM}} = \frac{1}{2} \mathop{\mathbb{E}}_{\substack{t_s, \mathbf{x}_s, \mathbf{x}_{s+1}, \mathbf{h}_s \\ \mathbf{h}_s \sim p_\phi(\mathbf{h}_s|\mathbf{h}_{s-1}, \mathbf{x}^{k_s}_{s,t_s}, t_s)}} \sum_{s=1}^{T} \left[\left\| (1-t_s)\mathbf{g}_\theta(\mathbf{x}^{k_s}_{s,t_s}, t_s, \mathbf{h}_{s-1}) - (\mathbf{x}_{s+1} - \mathbf{x}^{k_s}_{s,t_s}) \right\|^2 \right], \quad (12)$$

where $\mathbf{x}^{k_s}_{s,t_s} = \mathbf{x}_s + t_s(\mathbf{x}_{s+1} - \mathbf{x}_s) - (1-t_s)(1-k_s)(\mathbf{x}_s - \mathbf{z})$, t_s, k_s are independently sampled at each physical timestep s.

3.3 Latent temporal pyramids

We introduce two techniques to improve the computational efficiency of our model.

Firstly, we introduce an autoencoder P2VAE parametrized by ω to compress the state $\mathbf{x}$ to lower token count $\mathbf{y}$ used in the following implementations.

$$\begin{aligned} \mathbf{y} = \mathcal{E}_\omega(\mathbf{x}), \quad \hat{\mathbf{x}} = \mathcal{D}_\omega(\mathbf{y}), \\ \mathcal{L}_{\text{VAE}} = \frac{1}{2}\mathbb{E}\left\|\mathbf{x} - \hat{\mathbf{x}}\right\|^2 + \beta \text{KL}(q_\omega(\mathbf{y}|\mathbf{x})||p(\mathbf{y})). \end{aligned} \quad (13)$$

To further simplify the computational complexity, we introduce temporal pyramids in PFM [19], which resonates with the fact that a physical dynamics system is mostly Markovian and prediction relies less on farther previous states.

For early s, we use compressed latent states to propagate the PDE condition $\mathbf{h}_s$. In practice, we always train the conditional flow marching model on 4 consecutive states $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, where the FMT model takes $(\mathbf{x}^{k_0}_{0,t_0}, \mathbf{x}^{k_1}_{1,t_1}, \mathbf{x}^{k_2}_{2,t_2}, \mathbf{x}^{k_3}_{3,t_3})$ as input to predict the flow marching velocities by latent temporal pyramids $(Down(\mathbf{y}^{k_0}_{0,t_0}, 8), Down(\mathbf{y}^{k_1}_{1,t_1}, 4), Down(\mathbf{y}^{k_2}_{2,t_2}, 2), \mathbf{y}^{k_3}_{3,t_3})$.

3.4 Prediction and generation processes

We use the Euler ODE sampler (discretization on t) to propagate an intermediate state $\mathbf{x}^k_t$ to $\mathbf{x}_1$. (t_0, t_1, t_2, t_3) are initialized to be 0, and are updated simultaneously during the flow marching process. The discretization is taken to be $N = 100$ throughout the evaluation phase, with $dt = 0.01$. In a deterministic prediction setting, we set (k_0, k_1, k_2, k_3) to be 1, meaning that the past states are not noisy. In a generation setting, we choose to set (k_0, k_1, k_2) to be 1 and k_3 less than 1. The smaller the k, the larger the uncertainty about the current state. This setting allows $\mathbf{h}_3$ to be passed down from a clean history, and generate possible $\mathbf{x}_4$ out of pure noise. (k_0, k_1, k_2)'s parametrization choice can be further explored.

4 Experiments

4.1 Setup

Dataset gathering We consider a combination of public benchmark datasets for PDE foundation models: FNO-v [25], PDEBench [42], PDEArena [11], and the Well [34] to form a heterogeneous dataset consisting of 12 distinct dynamical systems. All the dynamical systems are 2D intrinsically, and three physical fields are chosen at maximum. We compressed the aforementioned datasets to the format of 128×128 spatial resolution with 3 multiphysics channels (c3p128) with float16 precision to form a 233 GB dataset, consisting over 2.5M trajectories with length 4. We provide the exact compression ratio and dataset information in A.2.

Training dataset The heterogeneous dataset is partitioned into train, valid, and test sets according to the original settings of each sub-dataset first; under the cases that there is no partition, we use a ratio of 8:1:1. We train P2VAE and FMT on the train set. Datasets are sampled with equal probabilities according to the practice in DPOT [13]. P2VAE's AdamW optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.995$, cosine learning rate schedule with 10% of linear warm up, and a weight decay of 1e-4; FMT's AdamW optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.95$, cosine learning rate schedule with 10% of linear warm up, and a weight decay of 0.01. Base learning rates of 1e-4 for a 256 batch size are adjusted according to batch sizes and model sizes to balance convergence speed and training stability. We conduct a two stage training recipe. We trained 2 P2VAEs, 16M and 87M, for 100k steps with KL term's weight β =1e-3. Based on the 16M P2VAE (with frozen weights), we train 3 FMTs with size 6M, 42M, and 138M (Small, Base, and Large) on the same training dataset for another 100k steps.

Evaluation metrics To assess the reconstruction and prediction quality of our model, we employ both the L2 relative error (L2RE), which is a common practice of PDE foundation models, and the variance-normalized root mean square error (VRMSE), as suggested by [34].

Implementation details For P2VAE, we reuse the standard SD-VAE [38] architecture to compress each state from c3p128 to c16p16 (12× compression rate) following the recommendation in [12]. P2VAE-16M uses 64 as the base dimensions, while P2VAE-87M uses 128. For FMT, we use the AdaLN-Zero mechanism introduced in [36] to condition a SiT [32]. In the Transformer side, we adopt the modern architecture RMSNorm and SwiGLU introduced by Llama-2 [44]. Multi-head self-attention with head dim 64 is implemented with FlashAttention v2 [9]. FMT-S, FMT-B, and FMT-L have 256, 512, and 768 as the embedding dimensions, respectively. The RNN in the diffusion forcing scheme is a GRU [8] which shares the same internal dimension as the embedding dimension in SiT; the current state $\mathbf{x}_t^k$ is first compressed onto a single token by cross attention to update the latent state $\mathbf{h}$ to inform dynamics.

4.2 Efficiency

Compared to a vanilla video-diffusion model [14] that operates with bidirectional self-attention across 4 frames with 256 tokens each and quadratic attention complexity, the efficiency gain due to FMT could be estimated by

$$\eta = \frac{(4 \times 16^2)^2}{(2^2)^2 + (4^2)^2 + (8^2)^2 + (16^2)^2} = 15. \tag{14}$$

4.3 Baselines

Baseline methods include: UNet [39], FNO [25], CNextU-net [17], which are trained on individual dynamics; DPOT [13], MPP [33], VICON [5], which are PDE foundation models jointly trained on several dynamics. All of the above is based on a deterministic neural operator setting. The results are listed in Tab. 1. The entries with $*$ is the implementation provided in the Well benchmark [34]. We provide the reconstruction error in L2RE and VRMSE of our P2VAEs to demonstrate the compression loss level due to the autoencoder structure for further comparisons. Note that since we unified the sub-datasets to p128c3 and float16, the metrics taken from other papers could be different on our format-unified dataset.

Table 1: P2VAE reconstruction error compared to benchmark PDE models. The best (or better) results among the existing statistics are in bold.

L2RE	FNO-v5	FNO-v4	FNO-v3	PA-NS	PA-NSC	PA-SWE	PB-CNSL	PB-CNSH	PB-SWE	W-AM	W-GS	W-SWE	W-RB	W-SF	W-TR	W-VE
UNet	0.198	0.119	0.0245	0.102	0.337		0.463	0.313	0.0521							
FNO	0.116	0.0922	0.0156	0.210	0.384		0.153	0.130	0.00912							
DPOT-30M	**0.0553**	**0.0442**	**0.0131**	0.0991	0.316		**0.0153**	**0.0245**	**0.00657**							
MPP-116M				0.0617			0.164	0.209								
VICON-88M				0.111			0.1561	0.0597								
P2VAE-16M	0.0890	0.0850	0.124	0.0651	0.0604	0.1093	0.0267	0.0334	0.438	0.0466	0.0774	0.0629	0.105	0.0956	0.0401	0.0360
P2VAE-87M	0.0802	0.0732	0.115	**0.0582**	**0.0527**	**0.1039**	0.0266	0.0325	0.186	**0.0329**	**0.0400**	**0.0596**	**0.0802**	**0.0846**	**0.0374**	**0.0274**
VRMSE	FNO-v5	FNO-v4	FNO-v3	PA-NS	PA-NSC	PA-SWE	PB-CNSL	PB-CNSH	PB-SWE	W-AM	W-GS	W-SWE	W-RB	W-SF	W-TR	W-VE
UNet*										0.2489	0.2252	0.3620	1.4860	3.447	0.2418	0.4185
FNO*										0.3691	**0.1365**	0.1727	0.8395	1.189	0.5001	0.7212
CNextU-net*										**0.1034**	0.1761	0.3724	0.6699	0.8080	**0.1956**	0.2499
P2VAE-16M	0.2240	0.2457	0.2721	0.0936	0.0850	0.1135	0.6028	0.3386	0.6504	0.4064	0.4916	0.1126	0.2499	0.1718	0.2838	0.2962
P2VAE-87M	**0.1886**	**0.2192**	**0.1986**	**0.0828**	**0.0743**	**0.1074**	**0.4444**	**0.2714**	**0.2945**	0.2016	0.3298	**0.0951**	**0.1886**	**0.1453**	0.2324	**0.1568**

4.4 Downstream evaluation results

Adapting foundation model to isotropic Kolmogorov turbulence According to REPA-E [22], we finetune the pretrained model (P2VAE+FMT) to adapt to an unseen system with a stop-gradient operation after the generation of latent states $\mathbf{y}$, so that the conditional flow marching loss won't deteriorate the autoencoder. The end-to-end finetuning loss is derived as

$$\mathcal{L}(\theta, \phi, \omega) = \mathcal{L}_{\text{CFM}}(\theta, \phi) + \lambda_{\text{VAE}} \mathcal{L}_{\text{VAE}}(\omega). \tag{15}$$

We conduct the experiment on an isotropic Kolmogorov turbulence dataset with u and v fields at $Re = 222$ [40]. We finetuned our FMT-B-42M model on 200 of the training trajectories in the train set for 5k steps with $\lambda_{\text{VAE}} = 1$ and test the performance on 500 trajectories in the test set. The metrics are shown in Table. 2, and one exemplary vorticity ($\boldsymbol{\omega} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$) reconstruction and prediction case is shown in Fig. 2.

Table 2: Few-shot adaptation result on the Kolmogorov turbulence dataset

Model	L2RE	VRMSE
P2VAE-16M-FT	0.0243	0.0614
FMT-B-42M-FT	0.0836	0.1053

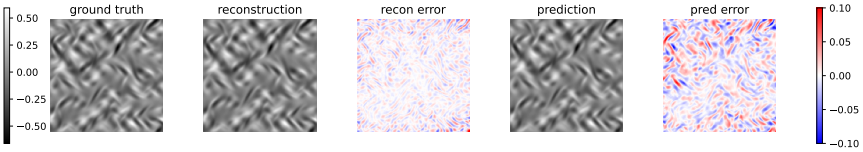

Figure 2: Reconstructed and predicted vorticity by the finetuned model

Long-term rollout We test the long-term rollout performance on the PDEArena-NS, PDEBench-CNS-Low, and PDEBench-CNS-High datasets of our model, and make the comparison with the statistics in VICON [5], as shown in Tab. 3. We plot sample trajectories based on FMT-B-42M in A.3.

Table 3: Comparison of long term rollout errors (in L2RE), with best results in bold.

L2RE	Case	FMT-S-6M	FMT-B-42M	FMT-L-138M	VICON-88M
Step 1	PA-NS	0.1060	0.0879	**0.0745**	0.1110
	PB-CNS-Low	0.0960	0.0796	**0.0557**	0.1561
	PB-CNS-High	0.0890	0.0450	**0.0411**	0.0597
Step 5	PA-NS	0.1889	0.1355	**0.1292**	0.2300
	PB-CNS-Low	0.0957	0.0958	**0.0872**	0.2456
	PB-CNS-High	0.1091	0.0992	**0.0797**	0.1973
Step 10	PA-NS	0.3318	0.2234	**0.2088**	0.3618
	PB-CNS-Low	0.1444	0.1119	**0.1004**	0.3747
	PB-CNS-High	0.1621	0.1218	**0.1198**	0.5788
Last step	PA-NS	0.5664	0.6134	**0.5271**	0.7781
	PB-CNS-Low	0.2820	**0.1298**	0.1311	0.3903
	PB-CNS-High	0.2130	0.1392	**0.1279**	0.7117
Average	PA-NS	0.4176	0.3859	**0.3048**	0.5627
	PB-CNS-Low	0.1480	0.1035	**0.0960**	0.2708
	PB-CNS-High	0.1497	0.1092	**0.0914**	0.3006

In DPOT [13], the authors noticed that injecting noise during training would benefit the long-term rollout capability because the distribution of model-predicted states can be partially taken into

account. However, they lack a systematic way to evaluate the noise level, which in turn demands a hyperparameter tuning process. In contrast, our model can deal with any noisy state because the distribution $q_t^k, k \in [0, 1]$ has been modelled, so that any misaligned predicted states are exposed during the training implicitly, which in turn minimize the exposure bias [2, 16] during long term prediction.

Generate ensemble of next states By tuning bridge parameter k_3 during the generation, we can effectively generate an ensemble of possible next state given a noisy initialization $k_3\mathbf{x}_3 + (1 - k_3)\mathbf{z}$ and concluded PDE condition $\mathbf{h}_3$ from clean past frames $(\mathbf{x}_0, \mathbf{x_1}, \mathbf{x_2})$.

We sampled one trajectory from PDEArena-NS and tested it on the FMT-B-42M model to generate a 32-batch size ensemble. The variance of the predicted ensemble is a decreasing function of prior noise level k_3 as shown in Fig. 3. Selected generated samples at different k_3 are displayed in A.4.

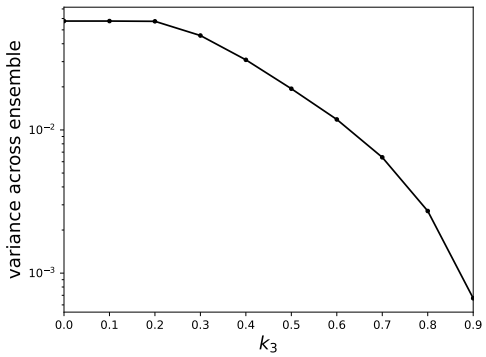

Figure 3: Average of batch-wise variation of $\mathbf{x}_4$ ensemble generated at different $\mathbf{x}_3$ noise levels k_3.

5 Conclusion

In this paper, we propose a conditional flow marching algorithm with a diffusion forcing scheme to construct a generative PDE foundation model that predicts future states given a series of past states. Empowered by a diverse training dataset, it displays excellent few-shot adaptation performance on unseen isotropic Kolmogorov turbulence; it allows accurate long-term rollout ability by reducing the exposure bias through generative modeling; and also allows generating reasonable and new physical dynamics data from noise.

We envision the current generative model to serve as a foundational tool for PDE-related applications that have a real-world impact. In the future, on the architecture side, we expect advanced Transformer models to enable better convergence of flow marching target; also we expect a stronger autoencoder would unlock the base performance bottleneck by minimizing the compression loss. On the application side, more generative applications could be explored based on the current model setting, including but not limited to conditional generation, data assimilation, uncertainty quantification, sparse reconstruction, etc.

Acknowledgments and Disclosure of Funding

This research was fully supported by the MIT GenAI Consortium (MGAIC) and was enabled in part by computational resources provided by the National Renewable Energy Laboratory (NREL) on the Kestrel system.

References

[1] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

[2] Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation, 2023. URL https://arxiv.org/abs/2204.01171.

[3] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

[4] Jan-Hendrik Bastek, WaiChing Sun, and Dennis M. Kochmann. Physics-informed diffusion models, 2025. URL https://arxiv.org/abs/2403.14404.

[5] Yadi Cao, Yuxuan Liu, Liu Yang, Rose Yu, Hayden Schaeffer, and Stanley Osher. Vicon: Vision in-context operator networks for multi-physics fluid dynamics prediction, 2025. URL https://arxiv.org/abs/2411.16063.

[6] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. URL https://arxiv.org/abs/2407.01392.

[7] Wei Wayne Chen, Doksoo Lee, Oluwaseyi Balogun, and Wei Chen. Gan-duf: Hierarchical deep generative models for design under free-form geometric uncertainty, 2022. URL https://arxiv.org/abs/2202.10558.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

[9] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

[11] Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling, 2022. URL https://arxiv.org/abs/2209.15616.

[12] Philippe Hansen-Estruch, David Yan, Ching-Yao Chung, Orr Zohar, Jialiang Wang, Tingbo Hou, Tao Xu, Sriram Vishwanath, Peter Vajda, and Xinlei Chen. Learnings from scaling visual tokenizers for reconstruction and generation, 2025. URL https://arxiv.org/abs/2501.09755.

[13] 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, 2024. URL https://arxiv.org/abs/2403.03542.

[14] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. URL https://arxiv.org/abs/2204.03458.

[15] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation, 2024. URL https://arxiv.org/abs/2406.17763.

[16] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion, 2025. URL https://arxiv.org/abs/2506.08009.

[17] Nabil Ibtehaz and Daisuke Kihara. Acc-unet: A completely convolutional unet model for the 2020s, 2023. URL https://arxiv.org/abs/2308.13680.

[18] Zhonglin Jiang, Qian Tang, and Zequn Wang. Generative reliability-based design optimization using in-context learning capabilities of large language models, 2025. URL https://arxiv.org/abs/2503.22401.

[19] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling, 2025. URL https://arxiv.org/abs/2410.05954.

[20] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[21] Boris Kramer, Benjamin Peherstorfer, and Karen E Willcox. Learning nonlinear reduced models from data with operator inference. *Annual Review of Fluid Mechanics*, 56(1):521–548, 2024.

[22] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers, 2025. URL https://arxiv.org/abs/2504.10483.

[23] Lizao Li, Robert Carver, Ignacio Lopez-Gomez, Fei Sha, and John Anderson. Generative emulation of weather forecast ensembles with diffusion models. *Science Advances*, 10(13): eadk4489, 2024. doi: 10.1126/sciadv.adk4489. URL https://www.science.org/doi/abs/10.1126/sciadv.adk4489.

[24] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning, 2023. URL https://arxiv.org/abs/2205.13671.

[25] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL https://arxiv.org/abs/2010.08895.

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

[27] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[28] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

[29] 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, 2024. URL https://arxiv.org/abs/2409.09811.

[30] Cooper Lorsung, Zijie Li, and Amir Barati Farimani. Physics informed token transformer for solving partial differential equations. *Machine Learning: Science and Technology*, 5(1):015032, 2024.

[31] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[32] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024. URL https://arxiv.org/abs/2401.08740.

[33] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for physical surrogate models, 2024. URL https://arxiv.org/abs/2310.02994.

[34] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J. Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Keaton Burns, Stuart B. Dalziel, Drummond B. Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich R. Kerswell, Suryanarayana Maddu, Jonah Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain

Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam H. Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning, 2025. URL https://arxiv.org/abs/2412.00568.

[35] Vivek Oommen, Aniruddha Bora, Zhen Zhang, and George Em Karniadakis. Integrating neural operators with diffusion models improves spectral representation in turbulence modeling, 2025. URL https://arxiv.org/abs/2409.08477.

[36] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

[37] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Gencast: Diffusion-based ensemble forecasting for medium-range weather, 2024. URL https://arxiv.org/abs/2312.15796.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

[40] Mohammed Sardar and Alex Skillen. Turbulent flow data as pytorch tensors for ml: Kolmogorov flow at re=222, and kelvin-helmholtz instability. Dataset, 2025. URL https://doi.org/10.48420/29329565.v1.

[41] Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Physical Review E*, 111(3):035304, 2025.

[42] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2024. URL https://arxiv.org/abs/2210.07182.

[43] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2024. URL https://arxiv.org/abs/2302.00482.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[46] Kazunari Wada, Katsuyuki Suzuki, and Kazuo Yonekura. Physics-guided training of gan to improve accuracy in airfoil design synthesis, 2023. URL https://arxiv.org/abs/2308.10038.

[47] Liu Yang, Constantinos Daskalakis, and George Em Karniadakis. Generative ensemble regression: Learning particle dynamics from observations of ensembles with physics-informed deep generative models, 2021. URL https://arxiv.org/abs/2008.01915.

[48] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[49] Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. Pdeformer: Towards a foundation model for one-dimensional partial differential equations, 2025. URL `https://arxiv.org/abs/2402.12652`.

[50] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, Han Yang, Hongxia Hao, Jielan Li, Ryota Tomioka, and Tian Xie. Mattergen: a generative model for inorganic materials design, 2024. URL `https://arxiv.org/abs/2312.03687`.

A Technical Appendices and Supplementary Material

A.1 Derivations for posterior mean $\mathbf{g}^*$ and continuity equation for location-scale interpolation kernel

We mentioned that regressing $\mathbf{g}$ to $\mathbf{u}_t^k = (\mathbf{x}_1 - \mathbf{x}_t^k)/(1-t)$ would recover the posterior-mean transporting field $\mathbf{g}^*$. The claim comes from the continuity equation.

Define the mixture marginal

$$q_t(\mathbf{x}) = \mathbb{E}_{(\mathbf{x_0}, \mathbf{x_1}, k)} \left[q_t^k(\mathbf{x} | \mathbf{x_0}, \mathbf{x_1}) \right]. \tag{16}$$

For any smooth test function ϕ,

$$\frac{d}{dt}\mathbb{E}_{q_t}\left[\phi(\mathbf{x}_t^k)\right] = \mathbb{E}_{q_t}\left[\nabla\phi(\mathbf{x}_t^k)\cdot\mathbf{u}_t^k\right] = \int \nabla\phi(\mathbf{x})\cdot \underbrace{\mathbb{E}\left[\mathbf{u}_t^k \,\middle|\, \mathbf{x}_t^k = \mathbf{x}, t\right]}_{:=\mathbf{g}^*(\mathbf{x},t)} q_t(\mathbf{x})d\mathbf{x} \tag{17}$$

Integrating by parts gives

$$\frac{d}{dt}\int \phi(\mathbf{x}) q_t(\mathbf{x}) d\mathbf{x} = -\int \phi(\mathbf{x}) \nabla_{\mathbf{x}} \cdot \left(q_t(\mathbf{x})\mathbf{g}^*(\mathbf{x},t)\right) d\mathbf{x}, \tag{18}$$

because $\int_{\partial\Omega} \phi(\mathbf{x})\mathbf{g}^*(\mathbf{x},t) q_t(\mathbf{x}) n(\mathbf{x}) d\mathbf{S} = 0, \phi \in \mathcal{C}_c^\infty(\Omega)$. Since this holds for all ϕ, q_t and $\mathbf{g}^*$ satisfy the continuity equation

$$\partial_t q_t(\mathbf{x}) + \nabla_{\mathbf{x}} \cdot \left(q_t(\mathbf{x})\mathbf{g}^*(\mathbf{x},t)\right) = 0, \tag{19}$$

The location-scale interpolation kernel is admissible – it induces a path of densities q_t that is transported by the velocity field equal to the posterior mean $\mathbf{g}^*$ of the sample-wise velocities $\mathbf{u}_t^k$; smoothness and integrability are valid.

A.2 Dataset description

All the data are compressed to float16 (half) precision to enable the Data Distributed Parallel training on a 4 H-100 GPU node.

FNO-v We upsampled original data from c1p64 to c3p128 (the 2nd and 3rd dimension are filled with zero). The dataset size is expanded from 11.1GB to 21GB. Trajectory count: FNO-v5 – 15.4k, FNO-v4 – 368k, FNO-v3 – 184k.

PDEArena For the PDEArena-NavierStokes(PA-NS) and PDEArena-NavierStokesCond(PA-NSC), the dataset size is compressed from 60GB to 25GB. For the PDEArena-ShallowWaterEquation(PA-SWE), it was compressed to 62GB from 76.6GB. Trajectory count: PA-NS – 48k, PA-NSC – 120k, PA-SWE – 470k.

PDEBench For the PDEBench-CompressibleNavierStokes(PB-CNS), unimportant physical fields are filtered. Thus, it becomes 65GB compressed from 551GB. For the PDEBench-ShallowWaterEquation(PB-SWE), it is compressed to 0.3GB from 6.2GB, the 2nd and 3rd dimension is filled with zero. Trajectory count: PB-CNS – 598k, PB-SWE – 77.6k.

The Well For the Well-GrayScott(W-GS), we fill the 3rd dimension with zero, ending up with a 5.3GB data set compressed from 153GB. For the Well-ActiveMatter(W-AM), we downsampled the data from c3p256 to c3p128, and obtained a compressed 1.1GB dataset from 51.3GB. For the Well-PlanetShallowWaterEquation(W-SWE), we downsampled the data from c3p256,512 to c3p128 and filtered out unimportant fields, so the data size is compressed to 9.3GB from 185.8GB. For the Well-RayleighBenard(W-RB), we downsampled the data from c3p512,128 to c3p128, and get a 26GB dataset from 342GB original data. For the Well-ShearFlow(W-SF), it is compressed to 14GB from 547GB by filtering out unimportant fields. For the Well-TurbulentRadiativeLayer2D(W-TR), it is downsampled from c3p128,384 to c3p128, thus compressed to 0.5GB from 6.9GB. For the Well-ViscoElasticInstability(W-VE), it is downsampled from c3p512 to c3p128, thus compressed to 0.5GB from 66GB. Trajectory count: W-GS – 92.2k, W-AM – 13.4k, W-SWE – 96.4k, W-RB – 266.6k, W-SF – 175.6k, W-TR – 7k, W-VE – 5.3k.

A.3 Rollout visualizations

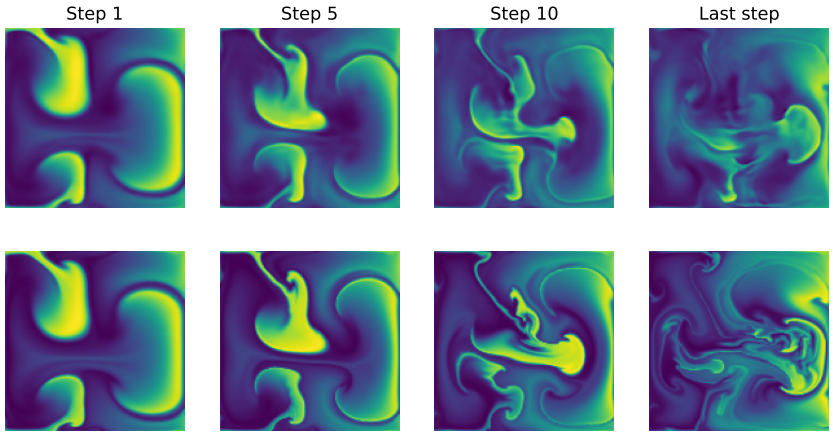

Figure A1: Sampled long-term rollout trajectories from PDEArena-NS by FMT-B-42M. Upper row: prediction. Bottom row: ground truth.

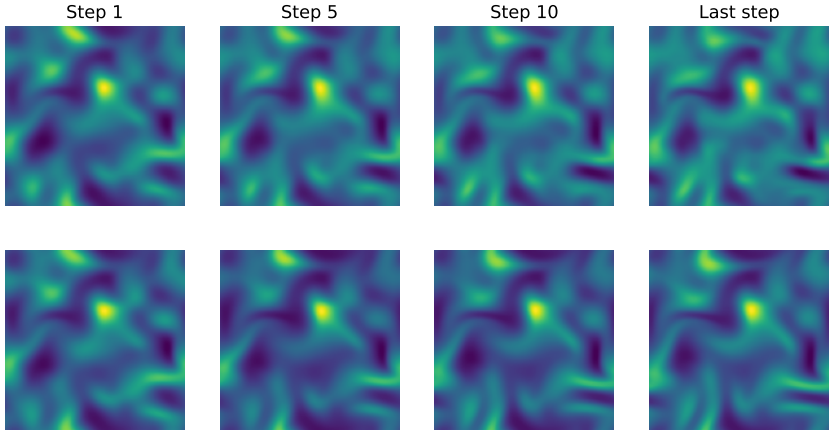

Figure A2: Sampled long-term rollout trajectories from PDEBench-CNS-Low by FMT-B-42M. Upper row: prediction. Bottom row: ground truth.

A.4 Generated ensemble at different k_3

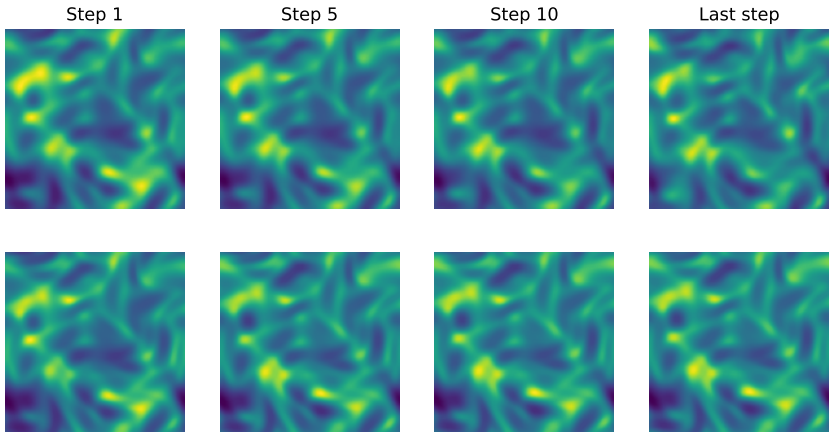

Figure A3: Sampled long-term rollout trajectories from PDEBench-CNS-High by FMT-B-42M. Upper row: prediction. Bottom row: ground truth.

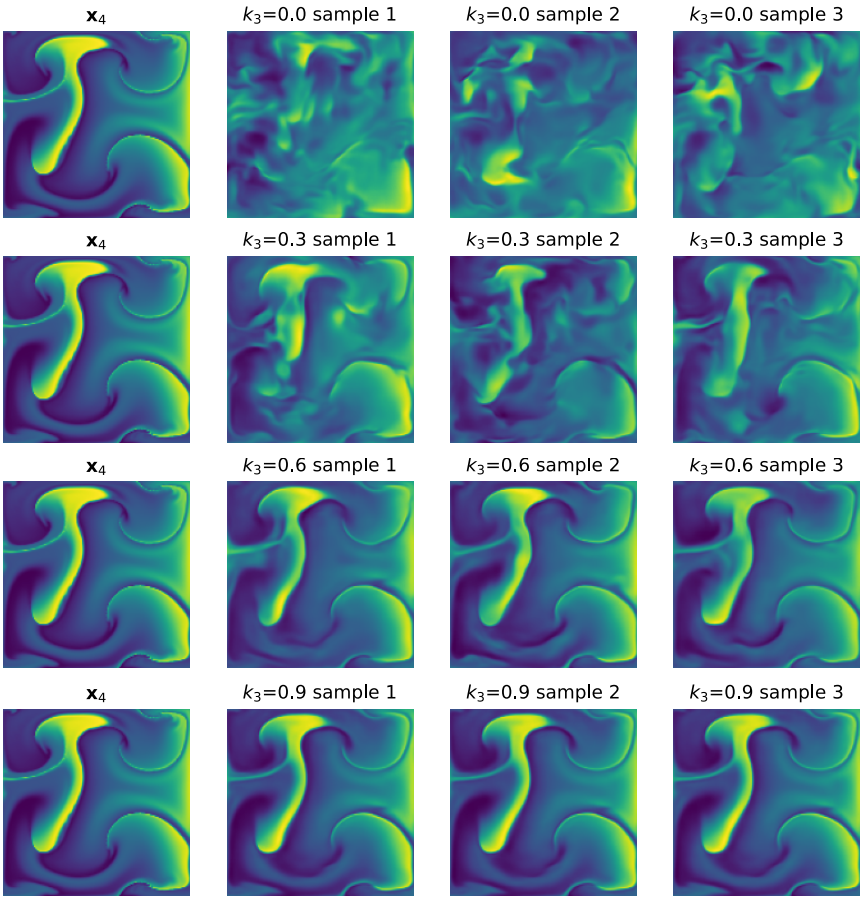

Figure A4: Generated ensembles at different k_3: $0, 0.3, 0.6, 0.9$.