ELUCIDATING THE DESIGN CHOICE OF PROBABILITY PATHS IN FLOW MATCHING FOR FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching has recently emerged as a powerful paradigm for generative modeling and has been extended to probabilistic time series forecasting in latent spaces. However, the impact of the specific choice of probability path model on forecasting performance remains under-explored. In this work, we demonstrate that forecasting spatio-temporal data with flow matching is highly sensitive to the selection of the probability path model. Motivated by this insight, we propose a novel probability path model designed to improve forecasting performance. Our empirical results across various dynamical system benchmarks show that our model achieves faster convergence during training and improved predictive performance compared to existing probability path models. Importantly, our approach is efficient during inference, requiring only a few sampling steps. This makes our proposed model practical for real-world applications and opens new avenues for probabilistic forecasting.

1 INTRODUCTION

Generative modeling has achieved remarkable success in recent years, especially for generating high-dimensional objects by learning mappings from simple, easily-sampled reference distributions, π_0, to complex target distributions, π_1. In particular, diffusion models have pushed the capabilities of generating realistic samples across various data modalities, including images (Ho et al., 2020; Song et al., 2020b; Karras et al., 2022), videos (Ho et al., 2022; Blattmann et al., 2023; Gupta et al., 2023), and spatio-temporal scientific data like climate and weather patterns (Pathak et al., 2024; Kohl et al., 2024). Despite their impressive performance, diffusion models often come with high computational costs during training and inference. Additionally, they typically assume a Gaussian reference distribution, which may not be optimal for all data types and can limit modeling flexibility.

One promising alternative is flow matching, where the mappings are learned via a stochastic process that transforms π_0 into π_1 through a random ordinary differential equation (ODE), approximating its marginal vector flow (Lipman et al., 2022; Albergo et al., 2023; Liu et al., 2022; Tong et al., 2023; Pooladian et al., 2023). While score-based generative models (Song & Ermon, 2019; Song et al., 2020a;b; Ho et al., 2020) are specific instances of flow matching, with Gaussian transition densities, the general framework allows for a broader class of interpolating paths. This flexibility can lead to deterministic sampling schemes that are faster and require fewer steps (Zhang & Chen, 2022). Recent work has demonstrated the remarkable capabilities of flow matching models for generating high-dimensional images (Esser et al., 2024) and discrete data (Gat et al., 2024).

Building on this, flow matching in latent space has recently been applied to forecasting spatio-temporal data (Davtyan et al., 2023) — predicting future frames in videos. This approach leverages latent representations to capture the complex dynamics inherent in temporal data. However, spatio-temporal forecasting, especially for video and dynamical systems data, presents unique challenges. A video prediction model capable of generalizing to new, unseen scenarios must implicitly "understand" the scene: detecting and classifying objects, learning how they move and interact, estimating their 3D shapes and positions, and modeling the physical laws governing the environment (Battaglia et al., 2016). Similarly, accurate climate and weather forecasting requires capturing intricate physical processes and interactions across multiple scales (Dueben & Bauer, 2018; Schultz et al., 2021).

We observe that, in the context of spatio-temporal forecasting, the performance of flow matching is highly sensitive to the choice of the probability path model, an important topic which has not been widely explored within a unified framework. Different probability paths can significantly impact the

accuracy and convergence of forecasting models, particularly when dealing with complex dynamical systems characterized by partial differential equations (PDEs) and chaotic behaviors. Motivated by this, we propose a novel probability path model specifically designed for probabilistic forecasting of dynamical systems. Our model leverages the continuous dynamics intrinsic to spatio-temporal data by interpolating between consecutive sequential samples. This approach ensures better alignment with the constructed flow, leading to improved predictive performance, more stable training, and greater inference efficiency. Existing probability path models often fail to fully capture the continuous nature of spatio-temporal data, resulting in a misalignment with flow-based methods and suboptimal outcomes. Our proposed model addresses these limitations directly.

Building on previous approaches, we provide a theoretical framework and efficient algorithms tailored to probabilistic forecasting using flow matching in latent space. Within this framework, we demonstrate that our probability path model outperforms existing methods across several forecasting tasks involving PDEs and other dynamical systems, achieving faster convergence during training and requiring fewer sampling steps during inference. These advances enhance the practicality of flow matching approaches for real-world applications, particularly in scenarios where computational resources and time are critical constraints. Our main contributions are the following.

- **Theoretical Framework and Efficient Algorithms:** We present a theoretical framework and efficient algorithms for applying flow matching in latent space to the probabilistic forecasting of dynamical systems (see Algorithms 1-2), extending the approach of Lipman et al. (2022) and Davtyan et al. (2023). Our approach is specifically tailored for time series data, enabling effective modeling of complex temporal dependencies inherent in dynamical systems.
- **Novel Probability Path Model:** We propose a new probability path model (Eq. (9)), specifically designed for modeling dynamical systems data. We provide intuitions to understand why our model leads to smoother training loss curve and faster convergence when compared to other models. We also provide theoretical insights to show that the variance of the vector field (VF) generating our proposed path can be lower than that of the optimal transport VF proposed by Lipman et al. (2022) for sufficiently correlated spatio-temporal samples (see Theorem 3 in App. C).
- **Empirical Validation:** We provide extensive empirical results to demonstrate that our proposed probability path model can outperform other flow matching models on several forecasting tasks involving PDEs and other dynamical systems (see Section 6). Our results demonstrate that the proposed probability path model outperforms existing flow matching models, achieving faster convergence during training and improved predictive performance.

2 Related Work

Generative models have gained significant attention in learning complex data distributions, particularly through the use of score-based diffusion models and flow-based models. These models have shown promise in generating realistic high-dimensional data such as images (Song et al., 2020b; Karras et al., 2022; Esser et al., 2024), videos (Davtyan et al., 2023; Shrivastava & Shrivastava, 2024) and time series (Meijer & Chen, 2024), and dynamical systems (Pathak et al., 2024; Kohl et al., 2024; Ren et al., 2024). The recent work of Rasul et al. (2021); Biloš et al. (2023); Ruhe et al. (2024); Kollovieh et al. (2024); Rühling Cachay et al. (2024) focuses on modeling time series with score-based diffusion models, which often require using many steps for sampling, whereas our proposed method requires as few as 10 steps. Related to our work is (Chen et al., 2024), which proposed a stochastic differential equation (SDE) framework using stochastic interpolants (Albergo et al., 2023) for probabilistic forecasting of time series, but their SDE based sampler typically requires many steps during generation. Another relevant work is (Tamir et al., 2024), which proposed a flow matching framework for time series modeling within the data space, but it focuses on small ODE datasets and does not consider forecasting tasks.

3 Flow Matching for Probabilistic Forecasting

In this section, we first introduce the objective of probabilistic forecasting; and then we discuss how flow matching can be used for learning conditional distributions in latent space.

Probabilistic forecasting framework. Suppose that we are given a training set of n trajectories, with each trajectory of length m, $S_n = \{(\mathbf{x}^{1:m})^{(i)}\}_{i=1,\dots,n}$, where $(\mathbf{x}^{1:m})^{(i)} = ((x^1)^{(i)}, \dots, (x^m)^{(i)})$, (with the $(x^l)^{(i)} \in \mathbb{R}^d$), coming from an underlying continuous-time dynamical system. For simplicity, we denote the trajectories as $\mathbf{x}^{1:m} = (x^1, \dots, x^m)$ unless there is a need to specify the corresponding n. The trajectories are observed at arbitrary time points $\mathbf{t}_{1:m} = (t_1, \dots, t_m)$ such that $x^i := x(t_i) \in \mathbb{R}^d$ and $(x(t))_{t\in[t_1,t_m]}$ are the observed states of the ground truth system. In practice, we may have access to only few trajectories, i.e., n is small or even $n = 1$, and the trajectories themselves may be observed at different time stamps.

The goal of probabilistic forecasting is to predict the distribution of the upcoming l elements given the first k elements, where $m = l + k$:

$$q(x^{k+1}, \dots, x^{k+l}|x^1, \dots, x^k) = \prod_{i=1}^{l} q(x^{k+i}|x^1, \dots, x^{k+i-1}). \tag{1}$$

We propose to model each one-step predictive conditional distribution in Eq. (1) via a probability density path. Instead of using score-based diffusion models to specify the path, we choose latent flow matching, a simpler method to train generative models. With flow matching, we directly work with probability paths, and we can simply avoid reasoning about diffusion processes altogether.

Flow matching in latent space. Let $z^\tau = \mathcal{E}(x^\tau)$ for $\tau = 1, \dots, m$, where $\mathcal{E}$ denotes a pre-trained encoder that maps from the data space to a lower dimensional latent space. Working in the latent space, our goal is to approximate the ground truth distribution $q(z^\tau|x^1, \dots, x^{\tau-1})$ by the parametric distribution $p(z^\tau|z^{\tau-1})$, which can then be decoded as $x^\tau = \mathcal{D}(z^\tau)$. The latent dynamics can be modeled by an ODE, $\dot{Z}_t = u_t(Z_t)$, where u_t is the vector field describing the instantenous rate of change of the state at time t. Learning the dynamics of the system is equivalent to approximating the vector field u_t by regressing a neural network using the mean squared error (MSE) loss.

Following the idea of flow matching, we infer the dynamics of the system generating $\mathbf{z}$ from the collection of latent observables by learning a time-dependent vector field $v_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, $t \in [0, 1]$, such that the ODE

$$\dot{\phi}_t(Z) = v_t(\phi_t(Z)), \quad \phi_0(Z) = Z, \tag{2}$$

defines a time-dependent diffeomorphic map (called a flow), $\phi_t(Z) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, that pushes a reference distribution $p_0(Z)$ towards the distribution $p_1(Z) \approx q(Z)$ along some probability density path $p_t(Z)$ and the corresponding vector field $u_t(Z)$. In other words, $p_t = [\phi_t]_* p_0$, where $[\cdot]_*$ denotes the push-forward operation. Here, q is the ground truth distribution, p denotes a probability density path, i.e., $p : [0, 1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$, and $\int p_t(Z)dZ = 1$. We also write $Z_t = \phi(Z)$; and thus the ODE can be written as $\dot{Z}_t = v_t(Z_t), Z_0 = Z$. Typically the reference distribution p_0 is chosen to be the standard Gaussian (Lipman et al., 2022; Liu et al., 2022).

In other words, the main goal of flow matching is to learn a deterministic coupling between p and q by learning a vector field v_t such that the solution to the ODE (2) satisfies $Z_0 \sim p$ and $Z_1 \sim q$. When $\mathbf{Z} = (Z_t)_{t\in[0,1]}$ solves Eq. (2) for a given function v_t, we say that $\mathbf{Z}$ is a flow with the vector field v_t. If we have such a vector field, then (Z_0, Z_1) is a coupling of (p, q). If we can sample from p, then we can generate approximate samples from the coupling by sampling $Z_0 \sim p$ and numerically integrating Eq. (2). This can be viewed as a continuous normalizing flow (Chen et al., 2018).

If one were given a pre-defined probability path $p_t(Z)$ and the corresponding vector field $u_t(Z)$ that generates the path, then one could parametrize $v_t(Z)$ with a neural network $v_t^\theta(Z)$, with θ the learnable parameter, and solve the least square regression by minimizing the flow matching loss:

$$\min_\theta \mathcal{L}_{fm}(\theta) := \mathbb{E}_{t,p_t(Z)}\omega(t)\|v_t^\theta(Z) - u_t(Z)\|^2, \tag{3}$$

where $t \in \mathcal{U}[0, 1]$, $Z \sim p_t(Z)$ and $\omega(t) > 0$ is a weighting function. We take $\omega(t) = 1$ unless specified otherwise. However, we do not have prior knowledge for choosing p_t and u_t, and there are many choices that can satisfy $p_1 \approx q$. Moreover, we do not have access to a closed form u_t that generates the desired p_t. We shall follow the approach of Lipman et al. (2022) and construct a target probability path by mixing simpler conditional probability paths. This probability path is the marginal probability path:

$$p_t(Z) = \int p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z}, \tag{4}$$

obtained by marginalizing the conditional probability density paths $p_t(Z|\tilde{Z})$ over observed latent trajectories $\tilde{Z}$, with $p_0(Z|\tilde{Z}) = p(Z)$ and $p_1(Z|\tilde{Z}) = \mathcal{N}(Z|\tilde{Z}, \epsilon^2 I)$ for a small $\epsilon > 0$. Doing so gives us a marginal probability p_1 which is a mixture distribution that closely approximates q. Then, assuming that $p_t(Z) > 0$ for all Z and t, we can also define a marginal vector field as:

$$u_t(Z) = \int u_t(Z|\tilde{Z}) \frac{p_t(Z|\tilde{Z})q(\tilde{Z})}{p_t(Z)} d\tilde{Z}, \tag{5}$$

where $u_t(Z|\tilde{Z})$ is a conditional vector field (conditioned on the latent trajectory $\tilde{Z}$). It turns out that this way of mixing the conditional vector fields leads to the correct vector field for generating the marginal probability path (4). We can then break down the intractable marginal VF into simpler conditional VFs which depends on a single sample.

To deal with the intractable integrals in Eq. (4)-(5) which complicates computation of an unbiased estimator of $\mathcal{L}_{fm}$, we shall minimize the conditional loss proposed by Lipman et al. (2022):

$$\min_\theta \mathcal{L}_{cfm}(\theta) := \mathbb{E}_{t, p_t(Z|\tilde{Z}), q(\tilde{Z})} \omega(t) \| v_t^\theta(Z) - u_t(Z|\tilde{Z}) \|^2, \tag{6}$$

where $t \in \mathcal{U}[0, 1]$, $\tilde{Z} \sim q(\tilde{Z})$, $Z \sim p_t(Z|\tilde{Z})$ and $u_t(Z|\tilde{Z})$ is the vector field defined *per sample* $\tilde{Z}$ that generates the conditional probability path $p_t(Z|\tilde{Z})$. Importantly, one can show that the solution of (6) is guaranteed to converge to the same result in (3); see Theorem 1 in App. C. Therefore, the conditional flow matching loss can match the pre-defined target probability path, constructing the flow that pushes p_0 towards p_1. Since both the probability path and VF are defined per sample, we can sample unbiased estimates of the conditional loss efficiently, particularly so with suitable choices of conditional probability paths and VFs.

4 Probability Path Models for Probabilistic Forecasting

In this section, we describe the family of probability paths that we consider for flow matching, and we propose an improved model for probabilistic forecasting of spatio-temporal data.

4.1 Common Probability Path Models

The family of Gaussian conditional probability paths gives us tractable choices to work with since the relevant quantities in Eq. (6) and thus the conditional flow can be defined explicitly. Therefore, we will work with Gaussian probability paths. Moreover, we are going to solve (6) over the dataset of all transition pairs $\mathcal{D} = \{(z^{\tau-1}, z^\tau)\}_{\tau=2,\dots,m}$, and use a pair of points for $\tilde{Z}$, setting $\tilde{Z} = (Z_0, Z_1) \in \mathcal{D}$. In particular, we consider the following class of models for the probability path:

$$p_t(Z|\tilde{Z} := (Z_0, Z_1)) = \mathcal{N}(Z|a_t Z_0 + b_t Z_1, c_t^2 I), \tag{7}$$

where a_t, b_t and c_t are differentiable time-dependent functions on $[0, 1]$, and I denotes the identity.

Table 1 provides five different choices of probability paths, including our proposed choice, that we study here. The optimal transport (OT) VF model described in Table 1 was initially proposed by Lipman et al. (2022), and setting $\epsilon_{min} = 0$ gives us the rectified flow model of Liu et al. (2022), which proposed connecting data and noise on a straight line. The stochastic interpolant model in Table 1 is the one considered by Chen et al. (2024). The VE and VP-diffusion conditional VFs (derived with Theorem 2) coincide with the vector fields governing the Probability Flow ODE for the VE and VP diffusion paths proposed in (Song et al., 2020b). It has been shown that combining diffusion conditional VFs with the flow matching objective leads to a training alternative that is more stable and robust when compared to existing score matching approaches (Lipman et al., 2022).

As remarked in (Lipman et al., 2022), there are many choices of vector fields that generate a given probability path. We shall use the simplest vector field that generates flow whose map is affine linear. Let $p_t(Z|\tilde{Z})$ be the Gaussian probability path defined in Eq. (7) and consider the flow map ψ_t defined as $\psi_t(Z) = a_t Z_0 + b_t Z_1 + c_t Z$ with $c_t > 0$. Then the unique vector field that defines ψ_t is (see Theorem 2 and the proof in App. C):

$$u_t(Z|\tilde{Z}) = \frac{c_t'}{c_t}(Z - (a_t Z_0 + b_t Z_1)) + a_t' Z_0 + b_t' Z_1, \tag{8}$$

Table 1: Choices of probability density paths that we study in this paper.

Model	a_t	b_t	c_t^2
VE-diffusion	1	0	σ_{1-t}^2, where σ_t is increasing in t, $\sigma_0 = 0$
VP-diffusion	$e^{-\frac{1}{2}T(1-t)}$	0	$1 - e^{-T(1-t)}, T(t) = \int_0^t \beta(s)ds$, β = noise scale
OT-VF / rectified flow	t	0	$(1 - (1 - \epsilon_{min})t)^2, \epsilon_{min} \geq 0$
Stochastic interpolant	$1 - t$	t or t^2	$\epsilon^2 t(1 - t)^2, \epsilon > 0$
Ours	$1 - t$	t	$\sigma_{min}^2 + \sigma^2 t(1 - t), \sigma_{min}, \sigma \geq 0$

where prime denotes derivative with respect to t, and $u_t(Z|\tilde{Z})$ generates the Gaussian path $p_t(Z|\tilde{Z})$.

In view of this, minimizing the conditional loss becomes:

$$\min_\theta \mathcal{L}_{cfm}(\theta) := \mathbb{E}_{t,z^\tau,z^{\tau-1},p_t(Z|z^\tau,z^{\tau-1})}\omega(t)\left\|v_t^\theta(Z) - \frac{c_t'}{c_t}(Z - (a_t z^\tau + b_t z^{\tau-1})) - a_t' z^\tau - b_t' z^{\tau-1}\right\|^2,$$

where $t \sim \mathcal{U}[0, 1]$, $Z \sim p_t(Z|z^\tau, z^{\tau-1})$ and $\mathbf{z} \sim q(\mathbf{z})$. We refer to this as the Flow Matching loss parametrization and work with this parametrization. There are other parametrizations: most popular ones are the Score Matching loss, Score Flow loss and DDPM loss. See App. B for a comparison of different loss parametrizations and App. A for connections to SDE based generative models.

4.2 A Novel Probability Path Model

We propose to choose $a_t = 1 - t$, $b_t = t$, $c_t^2 = \sigma_{min}^2 + \sigma^2 t(1 - t)$, in which case we have the probability path described by:

$$p_t(Z|\tilde{Z}) = \mathcal{N}(Z|tZ_1 + (1 - t)Z_0, (\sigma_{min}^2 + \sigma^2 t(1 - t))I), \tag{9}$$

which transports a Gaussian distribution centered around Z_0 with variance σ_{min}^2 at $t = 0$ to a Gaussian distribution centered around Z_1 at $t = 1$ with variance σ_{min}^2. Here $\sigma_{min}, \sigma \geq 0$ are tunable parameters. In the case when $\sigma_{min} = 0$, it describes a Brownian bridge that interpolates between Z_0 and Z_1 (Gasbarra et al., 2007). To ensure numerical stability when sampling $t \sim \mathcal{U}[0, 1]$, it is beneficial to use a small $\sigma_{min} > 0$. Note that σ^2 is a scale factor determining the magnitude of fluctuations around the path interpolating between Z_0 and Z_1. The variance c_t^2 is minimum with the value of σ_{min}^2 at the endpoints $t = 0$ and $t = 1$, and the maximum variance is $\sigma_{min}^2 + \sigma^2/4$ which occurs in the middle of the path at $t = 1/2$. The variance schedule is designed to balance exploration and stability. Low variance at the start ensures stable initialization, preventing the trajectory from deviating too far from the initial distribution. High variance in the middle allows the model to explore diverse paths in the latent space, avoiding mode collapse and enhancing diversity in the generated trajectories. Low variance at the end sharpens the trajectory, ensuring accurate reconstruction of the desired output. This strategy is inspired by findings in diffusion models that utilize a forward noising process and a backward denoising process (Ho et al., 2020; Song et al., 2020b), where such variance patterns have been shown to effectively manage the trade-off between exploration and refinement.

The corresponding vector field that defines the flow is then given by (applying Theorem 2):

$$u_t(Z|\tilde{Z}) = Z_1 - Z_0 + \frac{\sigma^2}{2}\frac{1 - 2t}{\sigma_{min}^2 + \sigma^2 t(1 - t)}(Z - (tZ_1 + (1 - t)Z_0)). \tag{10}$$

We remark that the conditional vector field $u_t(Z|\tilde{Z})$ is linear in Z and $\tilde{Z}$. This choice of the vector field corresponds to the choice of a linear model for the dynamics of the latent variable Z, and thus is reminiscent of the Koopman formalism (Koopman, 1931).

We expect that our proposed probability path model improves upon the other considered models, as it takes advantage of the inherent continuity and correlation in the spatio-temporal data. Intuitively, for time series samples whose underlying dynamics are continuous and obey a physical law, a Gaussian sample is typically further from the time series samples, so the distance between a Gaussian sample and a time series sample should generally be larger than the distance between consecutive time series samples (which can be highly correlated). Therefore, using a probability path that connects consecutive time series samples could lead to faster convergence and more stable training, when compared to using a path that simply connects a time series sample to a Gaussian sample, since the resulting flow model in the former case can better capture the true system dynamics with less effort.

Algorithm 1 Flow matching for spatio-temporal data

Input: Dataset of sequences D, number of iterations M
for i in range(1, M) **do**
  Sample a sequence $\mathbf{x}$ from the dataset D
  Encode it with a pre-trained encoder to obtain $\mathbf{z}$
  Choose a random target element $z^\tau, \tau \in \{3, \dots |\mathbf{z}|\}$, from $\mathbf{z}$
  Sample a step $t \sim U[0, 1]$
  Sample a noisy observation $Z \sim p_t(Z \mid z^\tau, z^{\tau-1})$, where p_t is given by Eq. (7)
  Compute $u_t(Z \mid z^\tau, z^{\tau-1})$
  Sample a condition frame $z^c, c \in \{1, \dots \tau - 2\}$
  Update the parameters θ via gradient descent

$$\nabla_\theta \| v_t^\theta(Z \mid z^{\tau-1}, z^c, \tau - c) - u_t(Z \mid z^\tau, z^{\tau-1}) \|^2 \tag{11}$$

end for
Return: A learned vector field, $v_t^{\theta^*}$

Algorithm 2 One-step ahead forecasting with forward Euler

Input: A sequence $(x^1, \dots, x^{T-1})$ containing the previous elements, number of integration steps N, grid $s_0 = 0 < s_1 < \cdots < s_N = 1$, a learnt vector field $v_s^{\theta^*}$ for $s \in [0, 1]$
Set $\Delta s_n = s_{n+1} - s_n$ for $n = 0, \dots, N-1$
Sample $Y_0^T \sim \mathcal{N}(\mathcal{E}(x^{T-1}), \sigma_{sam}^2 I)$, $\sigma_{sam} \geq 0$
for n in range(0, $N-1$) **do**
  Sample $c \sim \mathcal{U}(2, \dots, T-1)$
  $y^{T-c} = \mathcal{E}(x^{T-c})$
  $Y_{n+1}^T = Y_n^T + \Delta s_n v_{s_n}^{\theta^*}(Y_n^T | Y_0^T, y^{T-c}, T-c)$
end for
Return: An estimate of x^T, $\hat{x}^T = \mathcal{D}(Y_N^T)$

Moreover, if the consecutive samples are sufficiently correlated, then the variance of the vector field corresponding to our proposed probability path model can be lower than the variance of the vector field corresponding to the other choices of probability paths. We refer to Theorem 3 in App. C.3 for such comparison result for our proposed model and the optimal transport VF model of Lipman et al. (2022) (see also the discussions in Section C.3).

5 An Efficient Probabilistic Forecasting Algorithm

In this section, we present efficient algorithms for training and inferencing the flow matching model.

Recently Davtyan et al. (2023) proposed an efficient algorithm for latent flow matching for the task of video prediction, using the probability path generated by the optimal transport VF of Lipman et al. (2022). To enable efficient training, we shall follow Davtyan et al. (2023) and leverage the iterative nature of sampling from the learned flow and use a single random conditioning element from the past at each iteration. However, our method differs from Davtyan et al. (2023) as we shall use different probability paths and target vector fields.

Training. We set Z_1 to be the target element and Z_0 to be the reference element chosen to be the previous element before the target element, i.e., if $Z_1 = z^\tau$, then $Z_0 = z^{\tau-1}$, for $\tau = 2, \dots, m$. Note that this differs from Davtyan et al. (2023), where $\tilde{Z} = z_1$ (i.e., they do not use a reference element to define their probability path, whereas we use a pair of elements (Z_0, Z_1)). In this way, our probability path model maps a distribution centered around a previous state to the distribution of the current states, which is more natural from the point of view of probabilistic forecasting whose goal is to obtain an ensemble of forecasts. Algorithm 1 summarizes the training procedure of our method. Both the autoencoder and the VF neural network can also be jointly trained in an end-to-end manner, but our results show that separating the training can lead to improved performance. Moreover, doing so allows us to better assess the impact of using different probability paths.

Inference. We use an ODE sampler during inference to generate forecasts. The ODE sampler is described as follows. Let $(Y_i^\tau)_{i=0,\dots,N}$ denote the generation process, where N is the number of

integration steps and the superscript τ denotes the time index for which the generation/forecast is intended for. Given the previous elements $(x^1, \ldots, x^{T-1})$ of a time series sample, in order to generate the next element (i.e., the T-th element), we start with sampling the initial condition Y_0^T from $\mathcal{N}(z^{T-1}, \sigma_{sam}^2 I)$ for some small $\sigma_{sam} \geq 0$, where $z^{T-1} = \mathcal{E}(x^{T-1})$. This is in contrast to the procedure of Davtyan et al. (2023), which simply uses a mean-zero Gaussian sample instead. We then use an ODE solver to integrate the learned vector field along the time interval $[0, 1]$ to obtain Y_N^T. We use $\mathcal{D}(Y_N^T)$ as an estimate of x^T, and forecasting is done autoregressively. Algorithm 2 summarizes this procedure when the sampling is done using the forward Euler scheme. Note that we can also use computationally more expensive numerical schemes such as the Runge-Kutta (RK) schemes.

6 Empirical Results

In this section, we present our main empirical results to elucidate the design choice of probability paths. We focus on PDE dynamics forecasting tasks here (additional results can be found in App. D). We test the performance of our probability path model, i.e., Eq. (9) with $a_t = 1 - t$, $b_t = t$ and $c_t = \sqrt{\sigma_{min}^2 + \sigma^2 t(1-t)}$ on these tasks. We pick $\sigma_{min} = 0.001$, and treat σ and σ_{sam} as tunable parameters. We compare our proposed model with four other models of probability paths:

- **RIVER** (Davtyan et al., 2023): RIVER uses the OT-VF model in Table 1, i.e., $a_t = 0$, $b_t = t$, $c_t = 1 - (1 - \epsilon_{min})t$, choosing $\epsilon_{min} = 10^{-7}$.
- **VE-diffusion** in Table 1: We use $\sigma_t = \sigma_{min}\sqrt{\left(\frac{\sigma_{max}}{\sigma_{min}}\right)^{2t} - 1}$ with $\sigma_{min} = 0.01$, $\sigma_{max} = 0.1$, and sample t uniformly from $[0, 1 - \epsilon]$ with $\epsilon = 10^{-5}$ following Song et al. (2020b).
- **VP-diffusion** in Table 1: We use $\beta(s) = \beta_{min} + s(\beta_{max} - \beta_{min})$ where $\beta_{min} = 0.1, \beta_{max} = 20$ and t is sampled from $\mathcal{U}[0, 1 - \epsilon]$ with $\epsilon = 10^{-5}$, following (Song et al., 2020b). Thus, $T(s) = s\beta_{min} + \frac{1}{2}s^2(\beta_{max} - \beta_{min})$.
- **The stochastic interpolant path** in Table 1: We consider the path proposed by Chen et al. (2024) and use the suggested choice of $a_t = 1 - t$, $b_t = t^2$ and $c_t = \epsilon(1-t)\sqrt{t}$ (see Eq. (2) in (Chen et al., 2024) and note that $Var((1-t)W_t) = (1-t)^2 t$ for the standard Wiener process W_t). We choose $\epsilon = 0.01$. This is a path that is similar to ours, but with the variance c_t^2 chosen such that the maximum occurs at $t = 1/\sqrt{3}$ instead of at the middle of the path at $t = 1/2$. Later, we will see that different forms of variance can lead to vastly different performance in the considered tasks.

Evaluation metrics. We evaluate the models using the following metrics. First, we use the standard mean squared error (MSE) and the relative Frobenius norm error (RFNE) to measure the difference between predicted and true snapshots. Second, we compute metrics such as the peak signal-to-noise ratio (PSNR), and the structural similarity index measure (SSIM) to further quantify the quality and similarity of the generated snapshots (Wang et al., 2004). Third, we use the Pearson correlation coefficient to assess the correlation between predicted and true snapshots. Forth, we use the Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976) to measure the compatibility of the cumulative distribution function of the forecasts with the targets (see App. E.4).

Training details. We use an autoencoder (AE) to embed the training data into a low-dimensional latent space, which enables the model to capture the most relevant features of the data while reducing dimensionality (Azencot et al., 2020); see App. E for further discussion of the motivation. We then train a flow matching model in this latent space. Training generative models in latent space has also been considered by Vahdat et al. (2021) for score matching models and by Dao et al. (2023) for flow matching models. To train the AE, we minimize a loss function that consists of reconstruction error, in terms of MSE, between the input data and its reconstructed version from the latent space. The choice of the AE architecture is tailored to the complexity of the dataset (see App. E for details).

6.1 Probabilistic Forecasting of Dynamical Systems

We evaluate the performance of our proposed probability path model on challenging dynamical systems to demonstrate its effectiveness in forecasting complex continuous dynamics. Specifically, we consider the following tasks (for details see App. E.1):

- **Fluid Flow Past a Cylinder:** This task involves forecasting the vorticity of a fluid flowing past a cylinder. The model conditions on the first 5 frames and predicts the subsequent 20 frames at a resolution of 64×64 with 1 channel representing vorticity.
- **Shallow-Water Equation:** This dataset models the dynamics of shallow-water equations (Takamoto et al., 2022), capturing essential aspects of geophysical fluid flows. We use the first 5 frames for conditioning and predict the next 15 frames at a resolution of 128×128 with 1 channel representing horizontal flow velocity.
- **Diffusion-Reaction Equation:** This dataset models the dynamics of a 2D diffusion-reaction equation (Takamoto et al., 2022). We use the first 5 frames for conditioning and predict 15 future frames at a 128×128 with 2 channels representing velocity in the x and y directions.
- **Incompressible Navier-Stokes Equation:** As a more challenging benchmark, we consider forecasting the dynamics of a 2D incompressible Navier-Stokes equation (Takamoto et al., 2022). We use the first 5 frames for conditioning and predict the next 20 frames at a resolution of 512×512 with 2 channels representing velocity in the x and y directions.

Table 2 summarizes the performance of our model compared to other models across all tasks. It can be seen that our probability path model achieves the lowest test MSE and RFNE across all tasks, indicating more accurate forecasts. Moreover, the higher PSNR and SSIM scores indicate that our model better preserves spatial structures in the predictions. Our model also achieves the lowest CRPS on most tasks, showing that the predicted distribution of our model is closest to the true outcome. Despite the similarity of our proposed model with the stochastic interpolant of Chen et al. (2024), in that both models use consecutive samples to define the path, our model outperforms the stochastic interpolant model on most tasks, suggesting that choosing the maximum variance to occur at the middle of the path is a better choice. Importantly, our model is highly efficient during inference time since it requires only 10 sampling steps.

Figure 1 shows the Pearson correlation coefficients of the predicted snapshots over time for all models. Our model's predictions shows a slower decay of correlation coefficients compared to other models, indicating better temporal consistency and long-term predictive capability. Correlation coefficients about 95% indicate performance on par with physics-based numerical simulators.

Figure 2 compares the training loss curves of our model with others trained on the fluid flow past a cylinder and the shallow water equation task. Our method leads to faster convergence during training and smoother loss curves. This suggests that our model requires fewer iterations to generate high-quality samples when compared to other flow matching models.

Table 2: Results for forecasting dynamical systems using different probability path models for flow matching. The CRPSs are computed using 50 ensemble members, whereas the other results are averaged over 5 generations obtained with 10 sampling steps using RK4.

Task	Model	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)	CRPS (↓)
Flow past Cylinder	RIVER	3.05e-03	5.70e-02	42.66	0.98	1.76e-02
	VE-diffusion	2.75e-01	5.21e-01	26.81	0.52	2.43e-01
	VP-diffusion	3.20e-03	5.20e-01	42.60	0.97	2.64e-02
	Stoch. interpolant	3.39e-03	6.09e-02	41.82	0.97	3.36e-02
	Ours	**3.80e-04**	**2.30e-02**	**48.89**	**0.99**	**1.34e-02**
Shallow-Water	RIVER	9.29e-04	1.5e-01	34.90	0.91	**1.05e-02**
	VE-diffusion	1.23e-02	5.5e-01	28.76	0.57	4.42e-02
	VP-diffusion	1.31e-03	1.8e-01	34.33	0.88	**1.05e-02**
	Stoch. interpolant	1.06e-03	1.5e-01	35.57	0.88	1.41e-02
	Ours	**6.90e-04**	**1.3e-01**	**36.10**	**0.93**	1.08e-02
Diffusion-Reaction	RIVER	2.37e-03	2.07e-01	38.47	0.84	4.45e-02
	VE-diffusion	9.74e-02	1.7	32.83	0.35	1.01e-01
	VP-diffusion	1.72e-02	6.48e-01	34.65	0.53	6.70e-02
	Stoch. interpolant	6.17e-02	8.62e-01	**45.64**	0.76	1.27e-01
	Ours	**3.56e-04**	**1.16e-01**	34.34	**0.89**	**4.43e-02**
Navier-Stokes	RIVER	1.98e-04	2.5e-01	38.71	0.85	
	VE-diffusion	1.21e-04	2.4	29.53	0.27	
	VP-diffusion	7.62e-02	1.85e-01	40.73	0.93	
	Stoch. interpolant	7.26e-05	1.8e-01	37.81	0.88	
	Ours	**1.90e-05**	**9.8e-02**	**41.52**	**0.96**	

6.2 Ablation Study

To further assess our model, we conducted an ablation study focusing on the impact of various hyperparameters. Specifically, we study the impact of the values of σ, the choice of sampler, and the number of sampling steps during inference. For small σ_{sam}, we find that this parameter has negligible impact on test performance, so we fixed $\sigma_{\text{sam}} = 0$ for all experiments in this section.

Impact of σ on training stability. Figure 3 illustrates the effect of different σ values on the training loss curve for our method on the fluid flow past a cylinder task. We observed that larger values of σ (e.g., $\sigma = 0.1$) resulted in smoother loss curves and more stable convergence during training.

Effect of σ, sampler choice, and sampling steps on test performance. Table 3 investigates how different values of σ, the choice of sampler (Euler or RK4), and the number of sampling steps affect test performance. It can be seen, that even with as few as 5 sampling steps using the Euler scheme, our model perform reasonably well. However, increasing the number of sampling steps or employing the more computationally intensive RK4 sampler can help to lead to better results.

Table 3: Ablation study for the fluid flow past a cylinder task. The CRPSs are computed using 50 ensemble members, whereas the other results are averaged over 5 generations.

σ	sampler	sampling steps	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)	CRPS (↓)
0.0	Euler	5	1.78e-03	4.50e-02	44.12	0.98	2.42e-02
0.01	Euler	5	1.79e-03	4.50e-02	44.13	0.98	2.45e-02
0.1	Euler	5	1.94e-03	4.50e-02	44.26	0.98	2.40e-02
0.0	Euler	10	7.14e-04	2.90e-02	47.07	0.99	1.65e-02
0.01	Euler	10	3.81e-04	2.30e-02	48.73	0.99	1.25e-02
0.1	Euler	10	2.83e-03	5.10e-02	44.14	0.98	2.74e-02
0.0	RK4	10	3.89e-04	2.20e-02	49.22	0.99	1.27e-02
0.01	RK4	10	3.80e-04	2.30e-02	48.89	0.99	1.34e-02
0.1	RK4	10	6.49e-03	7.80e-02	40.70	0.97	4.03e-02
0.0	Euler	20	5.51e-04	2.70e-02	47.44	0.99	1.49e-02
0.01	Euler	20	7.32e-04	3.10e-02	46.85	0.99	1.71e-02
0.1	Euler	20	7.93e-04	3.10e-02	46.80	0.99	1.66e-02
0.0	RK4	20	9.31e-04	3.30e-02	46.42	0.99	1.82e-02
0.01	RK4	20	6.62e-04	2.80e-02	47.62	0.99	1.58e-02
0.1	RK4	20	6.58e-04	2.90e-02	47.14	0.99	1.61e-02

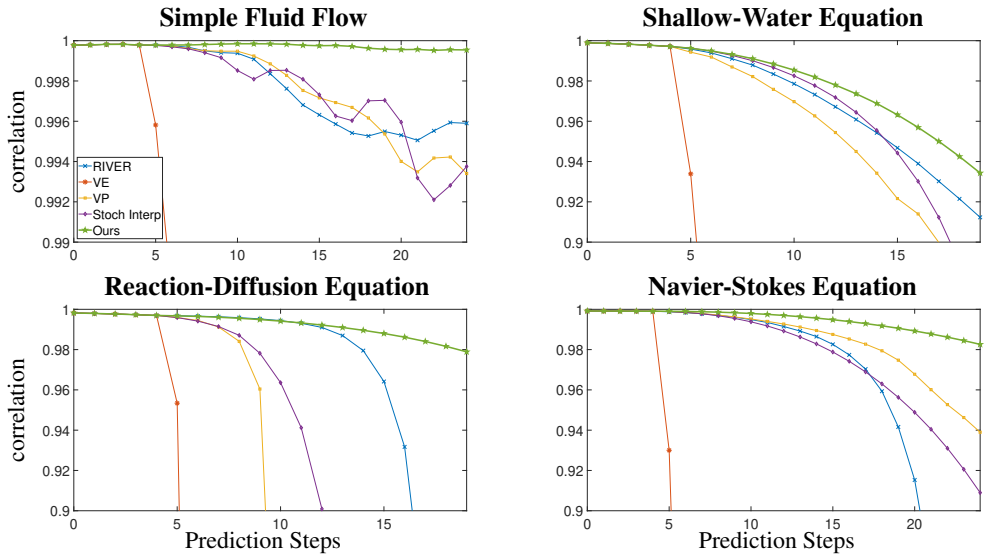

Figure 1: Pearson correlation coefficient to assess the correlation between predicted and true snapshots at various prediction steps for different probability path models. Our probability path model shows the best performance on all three tasks. Note that the first 5 snapshots are the conditioning snapshots.

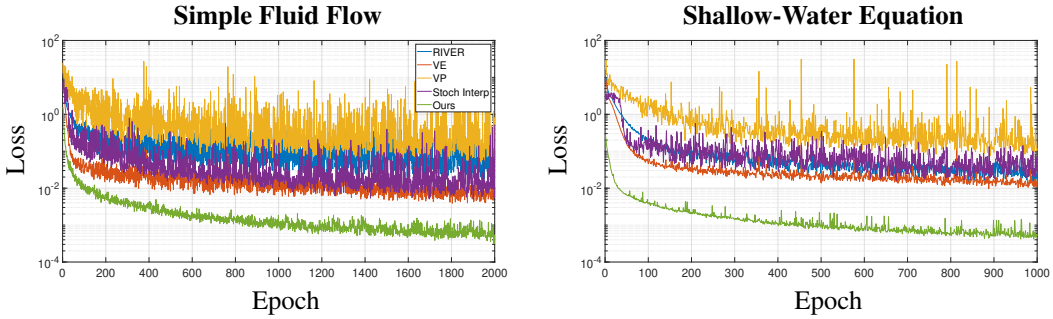

Figure 2: Training loss for different models of probability path for the fluid flow past a cylinder task. Our model leads to fastest convergence and smoothest loss curve among all models.

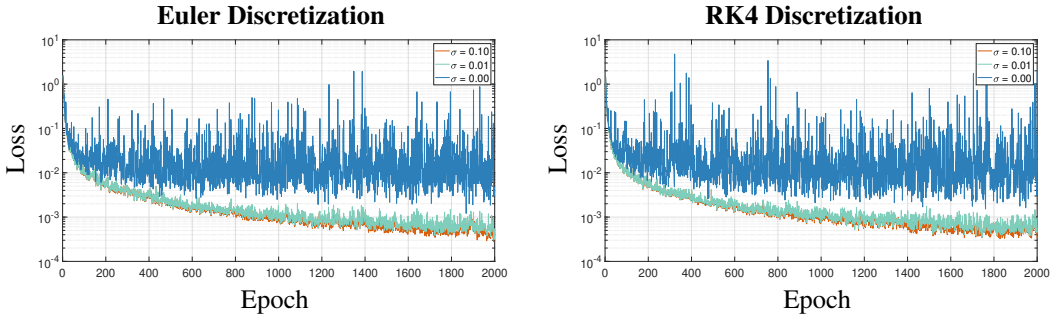

Figure 3: Training loss for different values of σ using our probability path model. The left subplot shows results for the Euler sampler, and the right subplot for the RK4 sampler. We see that the loss curve is sensitive to the choice of σ, with larger values of σ giving smoother loss curves.

6.3 Discussion

Our empirical results showed that our proposed model consistently outperformed other models across different forecasting tasks involving different types of dynamical systems. Our model shows improved training efficiency, with faster convergence reducing the computational resources and time required for model training. Moreover, our model is efficient during inference time since it only requires a few sampling steps, making it practical for real-world applications where computational efficiency is crucial. Additionally, the model maintained better temporal consistency, as indicated by a slower decay of Pearson correlation coefficients over longer prediction horizons, as well as improved probabilistic accuracy and reliability in forecasting, as reflected by lower CRPS values. These findings validate the effectiveness of our approach in modeling complex dynamical systems.

Our ablation study further validated the advantage of our proposed probability path model. We found that larger σ values not only contributed to smoother training loss curves but also enhanced the overall stability and efficiency of the model. Moreover, we saw that $\sigma_{\text{sam}} = 0$ can be fixed without compromising accuracy. The fact that our model achieved improved performance even with the simplest sampler (Euler scheme) and a minimal number of sampling steps (as few as five) validates its practical applicability, especially in scenarios where computational resources and time are limited.

7 Conclusion

In this work, we investigated the use of flow matching in latent space for probabilistic forecasting of spatio-temporal dynamics, providing a theoretical framework and an efficient algorithm. We demystified the critical role of the probability path design in this setting and proposed an improved probability path model. Our model leverages the inherent continuity and correlation in the spatio-temporal data, leading to more stable training and faster convergence. Our empirical evaluations on several PDE forecasting tasks demonstrated that our model performs favorably when compared to existing models. These findings highlight that while flow matching techniques hold great promise for probabilistic forecasting of spatio-temporal dynamics, it is important to select an appropriate probability path model to achieve optimal test performance. Future directions include extending the framework to go beyond Gaussian probability paths to better capture heavy-tailed distributions.

REFERENCES

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *International Conference on Machine Learning*, pp. 475–485. PMLR, 2020.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in Neural Information Processing Systems*, 29, 2016.

Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling temporal data as continuous functions with stochastic process diffusion. 2023.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

Yifan Chen, Mark Goldstein, Mengjian Hua, Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Probabilistic forecasting with stochastic interpolants and Föllmer processes. *arXiv preprint arXiv:2403.13724*, 2024.

Tim Colonius and Kunihiko Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197(25-28):2131–2146, 2008.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Aram Davtyan, Sepehr Sameni, and Paolo Favaro. Efficient video prediction via sparsely conditioned flow matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23263–23274, 2023.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped Langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

Peter D Dueben and Peter Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009, 2018.

N Benjamin Erichson, Lionel Mathelin, J Nathan Kutz, and Steven L Brunton. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(4):1867–1891, 2019.

N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Dario Gasbarra, Tommi Sottinen, and Esko Valkeila. Gaussian bridges. In *Stochastic Analysis and Applications: The Abel Symposium 2005*, pp. 361–382. Springer, 2007.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *arXiv preprint arXiv:2407.15595*, 2024.

Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

Diederik P Kingma and Ruiqi Gao. Understanding the diffusion objective as a weighted integral of ELBOs. *arXiv preprint arXiv:2303.00848*, 2023.

Georg Kohl, Liwei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. In *ICML 2024 AI for Science Workshop*, 2024.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Bernie Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

Caspar Meijer and Lydia Y Chen. The rise of diffusion models in time-series forecasting. *arXiv preprint arXiv:2401.03006*, 2024.

Jaideep Pathak, Yair Cohen, Piyush Garg, Peter Harrington, Noah Brenowitz, Dale Durran, Morteza Mardani, Arash Vahdat, Shaoming Xu, Karthik Kashinath, et al. Kilometer-scale convection allowing model emulation using generative diffusion modeling. *arXiv preprint arXiv:2408.10958*, 2024.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pp. 8857–8868. PMLR, 2021.

Pu Ren, Rie Nakata, Maxime Lacour, Ilan Naiman, Nori Nakata, Jialin Song, Zhengfa Bi, Osman Asif Malik, Dmitriy Morozov, Omri Azencot, et al. Learning physics for unveiling hidden earthquake ground motions via conditional generative modeling. *arXiv preprint arXiv:2407.15089*, 2024.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. *arXiv preprint arXiv:2402.09470*, 2024.

Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*, volume 10. Cambridge University Press, 2019.

Martin G Schultz, Clara Betancourt, Bing Gong, Felix Kleinert, Michael Langguth, Lukas Hubert Leufen, Amirpasha Mozaffari, and Scarlet Stadtler. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, 379(2194):20200097, 2021.

Gaurav Shrivastava and Abhinav Shrivastava. Video prediction by modeling videos as continuous multi-dimensional processes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7236–7245, 2024.

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

Ella Tamir, Najwa Laabid, Markus Heinonen, Vikas Garg, and Arno Solin. Conditional flow matching for time series modelling. In *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024.

Jakub M Tomczak. Latent variable models. In *Deep Generative Modeling*, pp. 57–127. Springer, 2021.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

APPENDIX

This appendix is organized as follows. In App. A, we provide some remarks on the connection of flow matching models to other generative models. In App. B, we provide commonly used loss parametrizations and compare them with our flow matching loss. In App. C, we provide theorems and proofs to justify the discussions in Section 3 (see Theorem 1-3). In App. D, we provide additional empirical results. In App. E, we provide the missing experimental details.

A CONNECTION TO SDE BASED GENERATIVE MODEL FRAMEWORKS

In this section, we provide some remarks on the connection between flow matching models and SDE based generative models (Song et al., 2020b).

Consider the following continuous-time Gaussian latent variable model (Tomczak, 2021): $Z_t = \mathcal{E}(X_t)$,

$$Z_t = A_t Z_0 + B_t Z_1 + L_t \epsilon, \quad t \in [0,1], \tag{12}$$

$$X_t = \mathcal{D}(Z_t), \tag{13}$$

where t is the continuous variable, $X_0 \in \mathbb{R}^d$ represent data samples, $Z \in \mathbb{R}^p$ is the latent variable, $\epsilon \sim \mathcal{N}(0, I)$ is independent of the random variables X_0, Z_0, Z_1. Here A_t, B_t and $L_t \geq 0$ are pre-specified coefficients which are possibly matrix-valued and time-dependent, $\mathcal{D}$ and $\mathcal{E}$ denote the decoder and encoder map respectively, and $\mathcal{D} \circ \mathcal{E} = I$. Note that Z_0 and Z_1 are initial and terminal point of the path $(Z_t)_{t\in[0,1]}$ in the latent space.

The above latent variable model can be identified (up to equivalence in law for each t) with the linear SDE of the form:

$$d\hat{Z}_t = F_t \hat{Z}_t dt + H_t Z_1 dt + G_t dW_t, \quad \hat{Z}_0 = Z_0, \quad t \in [0,1), \tag{14}$$

where $(W_t)_{t\in[0,1]}$ is the standard Wiener process. By matching the moments, we obtain

$$F_t = \dot{A}_t A_t^{-1}, \tag{15}$$

$$H_t = \dot{B}_t - \dot{A}_t A_t^{-1} B_t, \tag{16}$$

$$G_t G_t^T = \dot{L}_t L_t^T + L_t \dot{L}_t^T - \dot{A}_t A_t^{-1} L_t L_t^T - L_t L_t^T A_t^{-T} (\dot{A}_t)^T, \tag{17}$$

where the overdot denotes derivative with respect to t and A^T denotes the transpose of A.

Under the above formulation, various existing generative models such as DDPM (Ho et al., 2020), VP-SDE and VE-SDE of Song et al. (2020b;a), the critically damped SDE of Dockhorn et al. (2021), the flow matching models in (Lipman et al., 2022; Tong et al., 2023; Liu et al., 2022) and the stochastic interpolants of Albergo et al. (2023) can be recovered, and new models can be derived.

The following proposition establishes the connection between flow matching using our proposed probability path model, the Gaussian latent variable model (12) and the linear SDE model (14).

Proposition 1. *For every $t \in [0,1]$, the Z_t defined in Eq. (9) can be identified, up to equivalence in law, with the Z_t generated by the latent variable model (12) with $A_t = (1-t)I$, $B_t = tI$, $L_t = \sqrt{\sigma_{min}^2 + \sigma^2 t(1-t)}I$. For $t \in [0,1)$, it can also be identified with the solution $\hat{Z}_t$ of the linear SDE (14) with $F_t = -I/(1-t)$, $H_t = (1 + \frac{t}{1-t})I$ and $G_t = \sqrt{\sigma^2 + \frac{2\sigma_{min}^2}{1-t}}I$. Moreover, $\lim_{t\to 1} \hat{Z}_t =^d Z_1 + \sigma_{min}\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ and $=^d$ denotes equivalence in distribution.*

Proof. The identification follows from matching the moments of Z_t and $\hat{Z}_t$, i.e., applying Eq. (15)-(17).

To prove that $\lim_{t\to 1} \hat{Z}_t =^d Z_1 + \sigma_{min}\epsilon$, we use the explicit solution of the SDE:

$$\hat{Z}_t = \Phi(t,0) Z_0 + \int_0^t \Phi(t,s) H_s Z_1 ds + \int_0^t \Phi(t,s) G_s dW_s,$$

where $\Phi(t,s)$ is the fundamental solution of the homogeneous equation $d\Phi(t,s) = F_t \Phi(t,s) dt$ with $\Phi(s,s) = I$. For our $F_t = -I/(1-t)$, we have $\Phi(t,s) = \exp(-\int_s^t \frac{1}{1-u} du) I = (\frac{1-t}{1-s})I$.

Substituting this and the formula for H_t into the solution, we obtain $\hat{Z}_t = (1-t)Z_0 + \int_0^t (\frac{1-t}{1-s})(1 + \frac{s}{1-s})Z_1 ds + \int_0^t (\frac{1-t}{1-s})\sqrt{\sigma^2 + \frac{2\sigma_{min}^2}{1-s}} I dW_s$.

Now, let us examine each term as $t \to 1$. First, $(1-t)Z_0 \to 0$ as $t \to 1$ and $\int_0^t (\frac{1-t}{1-s})(1 + \frac{s}{1-s})Z_1 ds = tZ_1 \to Z_1$ as $t \to 1$. It remains to deal with the stochastic integral term $M_t := \int_0^t (\frac{1-t}{1-s})\sqrt{\sigma^2 + \frac{2\sigma_{min}^2}{1-s}} dW_s$. Note that M_t is an Itô integral that has zero mean, i.e. $\mathbb{E}M_t = 0$, and using Itô's formula (Särkkä & Solin, 2019),

$$\mathbb{E}M_t^2 = \int_0^t ((1-t)/(1-s))^2 \left(\sigma^2 + \frac{2\sigma_{min}^2}{1-s}\right) ds \tag{18}$$

$$= (1-t)^2 \left[\sigma^2 \int_0^t \frac{1}{(1-s)^2} ds + 2\sigma_{min}^2 \int_0^t \frac{1}{(1-s)^3} ds\right] \tag{19}$$

$$= (1-t)^2 \left[\sigma^2 \left(\frac{t}{1-t}\right) + \sigma_{min}^2 \left(\frac{1}{(1-t)^2} - 1\right)\right] \tag{20}$$

$$= (1-t)t\sigma^2 + \sigma_{min}^2 t(2-t), \tag{21}$$

which tends to σ_{min}^2 as $t \to 1$. Combining the above results, $\lim_{t\to 1} \hat{Z}_t \sim \mathcal{N}(Z_1, \sigma_{min}^2 I)$.

□

Remark 1. *The linear SDE in Proposition 1 is not the unique SDE that Eq. (9) can be identified with at each t. For instance, Z_t in Eq. (9) can also be identified with the solution $\hat{Z}_t$ of the SDE:*

$$d\hat{Z}_t = (Z_1 - Z_0)dt + \frac{\sigma^2}{2} \frac{1-2t}{\sqrt{\sigma_{min}^2 + \sigma^2 t(1-t)}} dW_t, \quad t \in [0,1]. \tag{22}$$

It is straightfoward to check that $\hat{Z}_t$ and Z_t are both Gaussian with the same mean and variance for all $t \in [0,1]$. Note that the probability path defined in Eq. (9) does not specify the covariance. However, since during training and inference, Z_t and Z_s are sampled using independent Gaussian samples, $Cov(Z_t, Z_s) = 0$ for $t \neq s$. On the other hand, the SDEs that we have constructed above specify the covariance. Thus, while $Z_t =^d \hat{Z}_t$ for all $t \in [0,1]$, the paths $(Z_t)_{t\in[0,1]}$ and $(\hat{Z}_t)_{t\in[0,1]}$ may not be equivalent in law as they may have different covariance. It is natural to ask how the choice of covariance structure affects the model performance. Note that generating the SDE path $(\hat{Z}_t)_{t\in[0,1]}$ requires simulation of the SDE. This beats the original purpose of flow matching, which champions a simulation-free approach that directly works with a pre-specified probability path.

B On Different Loss Parametrizations

In this section, we list popular choices of loss parametrization considered in the literature and connect them to our flow matching loss. We refer to (Kingma & Gao, 2023) for a more comprehensive discussion. Recall that the Gaussian path that we consider is: $Z_t = a_t Z_0 + b_t Z_1 + c_t \xi$, where $\xi \sim \mathcal{N}(0, I)$. In general, these loss parametrizations take the form of:

$$\mathcal{L}(\theta) := \mathbb{E}_{t, p_t(Z|\tilde{Z}), q(\tilde{Z})} \omega(t) \| m_t^\theta(Z) - m_t(Z, \tilde{Z}) \|^2, \tag{23}$$

where $\omega(t) > 0$ is a weighting function, $m_t(Z, \tilde{Z})$ is the object (conditioned on $\tilde{Z}$) to be learnt and m_t^θ is a neural network model used to learn the object of interest. Depending on which object one would like to learn/match, we have different loss parametrizations.

Flow matching. The flow matching loss that we focus in this paper is:

$$\mathcal{L}_{cfm}(\theta) := \mathbb{E}_{t, p_t(Z|\tilde{Z}), q(\tilde{Z})} \omega(t) \| v_t^\theta(Z) - u_t(Z|\tilde{Z}) \|^2, \tag{24}$$

where one aims to learn the flow generating vector field:

$$u_t(Z|\tilde{Z}) = \frac{c_t'}{c_t}(Z - (a_t Z_0 + b_t Z_1)) + a_t' Z_0 + b_t' Z_1, \tag{25}$$

Score matching. The score matching loss is:

$$\mathcal{L}_{sm}(\theta) := \mathbb{E}_{t,p_t(Z|\tilde{Z}),q(\tilde{Z})}\lambda(t)\|s_t^\theta(Z) - \nabla \log p_t(Z|\tilde{Z})\|^2, \tag{26}$$

where $\lambda(t) > 0$ is a weighting function and one aims to learn the score function:

$$\nabla \log p_t(Z|\tilde{Z}) = \frac{a_t Z_0 + b_t Z_1 - Z}{c_t^2}. \tag{27}$$

If $\lambda(t) = c_t^2$, then this reduces to the original score matching loss (Song & Ermon, 2019), whereas if $\lambda(t) = \beta(1-t)$, this becomes the score flow loss (Song et al., 2021).

Noise matching. The noise matching loss is:

$$\mathcal{L}_{nm}(\theta) := \mathbb{E}_{t,p_t(Z|\tilde{Z}),q(\tilde{Z})}\|\epsilon_t^\theta(Z) - \epsilon_t(Z|\tilde{Z})\|^2, \tag{28}$$

where one aims to learn the noise:

$$\epsilon_t(Z|\tilde{Z}) = \frac{Z - (a_t Z_0 + b_t Z_1)}{c_t}. \tag{29}$$

C Theoretical Results and Proofs

In this section, we provide theorems and proofs to justify the discussions in Section 3 and Section 4.

C.1 Connecting Flow Matching with Conditional Flow Matching

The following theorem justifies the claim that minimizing $\mathcal{L}_{fm}$ is equivalent to minimizing $\mathcal{L}_{cfm}$.

Theorem 1. *If the conditional vector field $u_t(Z|\tilde{Z})$ generates the conditional probability path $p_t(Z|\tilde{Z})$, then the marginal vector field u_t in Eq. (5) generates the marginal probability path p_t in Eq. (4). Moreover, if $p_t(Z) > 0$ for all t, Z, then $\mathcal{L}_{fm}$ and $\mathcal{L}_{cfm}$ are equal up to a constant independent of θ.*

Proof. The proof is a straightforward extension of the proofs of Theorem 1-2 in (Lipman et al., 2022) from conditioning on data samples to conditioning on latent samples and allowing an arbitrary weighting function $\omega(t)$.

Suppose that the conditional vector field $u_t(Z|\tilde{Z})$ generates the conditional probability path $p_t(Z|\tilde{Z})$, we would like to show that the marginal vector field u_t in Eq. (5) generates the marginal probability path p_t in Eq. (4). To show this, it suffices to verify that p_t and u_t satisfy the continuity equation:

$$\frac{d}{dt}p_t(Z) + div(p_t(Z)u_t(Z)) = 0, \tag{30}$$

where the divergence operator, div, is defined with respect to the latent variable $Z = (Z^1, \ldots, Z^d)$, i.e., $div = \sum_{i=1}^d \frac{\partial}{\partial Z^i}$.

We begin by taking derivative of $p_t(Z)$ with respect to time:

$$\begin{aligned}
\frac{d}{dt}p_t(Z) &= \frac{d}{dt}\int p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z} && (31)\\
&= \int \frac{d}{dt}p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z} && (32)\\
&= -\int div(u_t(Z|\tilde{Z})p_t(Z|\tilde{Z}))q(\tilde{Z})d\tilde{Z} && (33)\\
&= -div\left(\int u_t(Z|\tilde{Z})p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z}\right) && (34)\\
&= -div(u_t(Z)p_t(Z)). && (35)
\end{aligned}$$

In the third line, we use the fact that $u_t(\cdot|\tilde{Z})$ generates $p_t(\cdot|\tilde{Z})$. In the last line, we use Eq. (5). In the second and forth line above, the exchange of integration and differentiation can be justified by assuming that the integrands satisfy the regularity conditions of the Leibniz rule.

Next, we would like to show that if $p_t(Z) > 0$ for all t, Z, then $\mathcal{L}_{fm}$ and $\mathcal{L}_{cfm}$ are equal up to a constant independent of θ. We follow (Lipman et al., 2022) and assume that $q(Z)$ and $p_t(Z|\tilde{Z})$ are decreasing to zero sufficiently fast as $\|Z\| \to 0$, that $u_t, v_t, \nabla_\theta v_t$ are bounded, so that all integrals exist and exchange of integration order is justified via Fubini's theorem.

Using the bilinearity of the 2-norm, we decompose the squared losses as:

$$\|v_t^\theta(Z) - u_t(Z)\|^2 = \|v_t^\theta(Z)\|^2 - 2\langle v_t^\theta(Z), u_t(Z)\rangle + \|u_t(Z)\|^2, \tag{36}$$

$$\|v_t^\theta(Z) - u_t(Z|\tilde{Z})\|^2 = \|v_t^\theta(Z)\|^2 - 2\langle v_t^\theta(Z), u_t(Z|\tilde{Z})\rangle + \|u_t(Z|\tilde{Z})\|^2. \tag{37}$$

Now,

$$\mathbb{E}_{p_t(Z)}\|v_t^\theta(Z)\|^2 = \int \|v_t^\theta(Z)\|^2 p_t(Z)dZ \tag{38}$$

$$= \int\int \|v_t^\theta(Z)\|^2 p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z}dZ \tag{39}$$

$$= \mathbb{E}_{q(\tilde{Z}),p_t(Z|\tilde{Z})}\|v_t^\theta(Z)\|^2, \tag{40}$$

where we use Eq. (4) in the second equality above and exchange the order of integration in the third equality.

Next, we compute:

$$\mathbb{E}_{p_t(Z)}\langle v_t^\theta(Z), u_t(Z)\rangle = \int \left\langle v_t^\theta(Z), \frac{\int u_t(Z|\tilde{Z})p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z}}{p_t(Z)} \right\rangle p_t(Z)dZ \tag{41}$$

$$= \int \left\langle v_t^\theta(Z), \int u_t(Z|\tilde{Z})p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z} \right\rangle dZ \tag{42}$$

$$= \int\int \langle v_t^\theta(Z), u_t(Z|\tilde{Z})\rangle p_t(Z|\tilde{Z})q(\tilde{Z})d\tilde{Z}dZ \tag{43}$$

$$= \mathbb{E}_{q(\tilde{Z}),p_t(Z|\tilde{Z})}\langle v_t^\theta(Z), u_t(Z|\tilde{Z}), \tag{44}$$

where we first plug in Eq. (5) and then exchange the order the integration in order to arrive at the last equality.

Finally, noting that u_t are $\omega(t)$ independent of θ (and are thus irrelevant for computing the loss gradients), we have proved the desired result. □

C.2 Identifying the Vector Field that Generates the Gaussian Paths

Similar to Theorem 3 in (Lipman et al., 2022), we have the following result, which identifies the unique vector field that generates the Gaussian probability path.

Theorem 2. *Let $p_t(Z|\tilde{Z})$ be the Gaussian probability path defined in Eq. (7) and consider the flow map ψ_t defined as $\psi_t(Z) = a_t Z_0 + b_t Z_1 + c_t Z$ with $c_t > 0$. Then the unique vector field that defines ψ_t is:*

$$u_t(Z|\tilde{Z}) = \frac{c_t'}{c_t}(Z - (a_t Z_0 + b_t Z_1)) + a_t' Z_0 + b_t' Z_1, \tag{45}$$

where prime denotes derivative with respect to t, and $u_t(Z|\tilde{Z})$ generates the Gaussian path $p_t(Z|\tilde{Z})$.

Proof. Let $p_t(Z|\tilde{Z})$ be the Gaussian probability path defined in Eq. (7) and consider the flow map ψ_t defined as $\psi_t(Z) = a_t Z_0 + b_t Z_1 + c_t Z$. We would like to show that the unique vector field that defines ψ_t is:

$$u_t(Z|\tilde{Z}) = \frac{c_t'}{c_t}(Z - (a_t Z_0 + b_t Z_1)) + a_t' Z_0 + b_t' Z_1, \tag{46}$$

and $u_t(Z|\tilde{Z})$ generates the Gaussian path $p_t(Z|\tilde{Z})$.

We denote $w_t = u_t(Z|\tilde{Z})$ for notational simplicity. Then,

$$\frac{d}{dt}\psi_t(Z) = w_t(\psi_t(Z)). \tag{47}$$

Since ψ_t is invertible (as $c_t > 0$), we let $Z = \psi^{-1}(Y)$ and obtain

$$\psi_t'(\psi^{-1}(Y)) = w_t(Y), \tag{48}$$

where the prime denotes derivative with respect to t and we have used the apostrosphe notation for the derivative to indicate that ψ_t' is evaluated at $\psi^{-1}(Y)$.

Inverting $\psi_t(Z)$ gives:

$$\psi_t^{-1}(Y) = \frac{Y - \mu_t(\tilde{Z})}{c_t}, \tag{49}$$

where $\mu_t(\tilde{Z}) := a_t Z_0 + b_t Z_1$.

Differentiating ψ_t with respect to t gives $\psi_t'(Z) = c_t' Z + \mu_t'(\tilde{Z})$.

Plugging the last two equations into Eq. (48), we obtain:

$$w_t(Y) = \frac{c_t'}{c_t}(Y - \mu_t(\tilde{Z})) + \mu_t'(\tilde{Z}) \tag{50}$$

which is the result that we wanted to show.

□

C.3 Comparing the Variance of Different Vector Field Models

We show that under reasonable assumptions, the variance of the vector field corresponding to our proposed probability path model is lower than the variance of the vector field corresponding to the other choices of probability paths. Here the variance is taken with respect to the randomness in the samples z^τ and the Gaussian samples drawn during gradient descent updates. To simplify our analysis and to facilitate discussion, we only compare our probability path model to the model generated by the optimal transport VF of Lipman et al. (2022), setting $\epsilon_{min} := 0$ (rectified flow) without loss of generality.

We start with the following lemma.

Lemma 1. *Let A, B, C, D be random vectors where C and D are independent, both A and B are independent of C and D (but A and B could be dependent). If $Cov(A, B) \geq (Var(C) - Var(D) + Var(B))/2$, then*

$$Var(A + D) \geq Var(A - B + C) = Var(B - A + C), \tag{51}$$

where $A \geq B$ means that $A - B$ is positive semidefinite.

Proof. We compute:

$$\begin{aligned}
Var(A + D) &= Var(A - B + C + B + D - C) && (52)\\
&= Var(A - B + C) + Var(B + D) + Var(C) + 2Cov(A - B + C, B + D) \\
&\quad - 2Cov(A - B + C, C) - 2Cov(B + D, C) && (53)\\
&= Var(A - B + C) + Var(B + D) - Var(C) + 2Cov(A - B + C, B) \\
&\quad + 2Cov(A - B + C, D) && (54)\\
&= Var(A - B + C) + Var(B + D) - Var(C) + 2Cov(A, B) - 2Var(B) && (55)\\
&= Var(A - B + C) - Var(C) + Var(D) + 2Cov(A, B) - Var(B), && (56)
\end{aligned}$$

where we have simply rearranged the terms in the first equality, used the formula $Var(A+B+C) = Var(A)+Var(B)+Var(C)+2Cov(A, B)+2Cov(A, C)+2Cov(B, C)$, bilinearity of covariance, the facts that $Cov(A, A) = Var(A)$ and $Var(cA) = c^2 A$ for a scalar c, as well as the assumption that both A and B are independent of C, D in the last four equalities.

Therefore, if $-Var(C) + Var(D) + 2Cov(A, B) - Var(B) \geq 0$, then we have $Var(A + D) \geq Var(A - B + C)$.

□

Now, we consider the vector fields that correspond to the above two models of probability path, denoting the vector field that corresponds to our model as u_t and the vector field that corresponds to the rectified flow model as $\tilde{u}_t$. They generate the probability path $Z_t = tz^\tau + (1-t)z^{\tau-1} + c_t\xi$ and $\tilde{Z}_t = tz^\tau + (1-t)\eta$ respectively, where $\xi, \eta \sim \mathcal{N}(0, I)$, $c_t = \sqrt{\sigma_{min}^2 + \sigma^2 t(1-t)}$ and $t \in [0, 1]$.

Applying Eq. (8), we have:

$$u_t(Z_t|z^{\tau-1}, z^\tau) = z^\tau - z^{\tau-1} + c_t'\xi, \tag{57}$$

$$\tilde{u}_t(\tilde{Z}_t|z^{\tau-1}) = z^{\tau-1} - \eta. \tag{58}$$

Theorem 3. *Suppose that $(z^\tau)_{\tau=1,\dots,m}$, with the $z^\tau \in \mathbb{R}^d$, is a discrete-time stochastic process with nonzero correlation in time and let $t \in [0, 1]$ be given. If $Cov(z^{\tau-1}, z^\tau) \geq \frac{1}{2}\left(\left(\frac{\sigma^4}{4\sigma_{min}^2} - 1\right) I + Var(z^\tau)\right)$, then $Var(\tilde{u}_t(\tilde{Z}_t|z^{\tau-1})) \geq Var(u_t(Z_t|z^{\tau-1}, z^\tau))$.*

Proof. Note that $c_t' = \frac{\sigma^2(1-2t)}{2\sqrt{\sigma_{min}^2+\sigma^2 t(1-t)}}$, $Var(c_t'\xi) = (c_t')^2 I$ and $Var(-\eta) = I$. Therefore, using these and applying Lemma 1 with $A := z^{\tau-1}$, $B := z^\tau$, $C := c_t'\xi$ and $D := -\eta$, allow us to establish the claim that $Var(\tilde{u}_t(Z_t|z^{\tau-1})) \geq Var(u_t(Z_t|z^{\tau-1}, z^\tau))$ if

$$Cov(z^{\tau-1}, z^\tau) \geq \frac{1}{2}\left(\frac{\sigma^4(1-2t)^2 I}{4(\sigma_{min}^2 + \sigma^2 t(1-t))} + Var(z^\tau) - I\right). \tag{59}$$

Since the function $f(t) := \frac{\sigma^4(1-2t)^2}{4(\sigma_{min}^2+\sigma^2 t(1-t))}$ is maximized at the endpoints $t = 0, 1$ with the maximum value of $\sigma^4/4\sigma_{min}^2$, the desired result stated in the theorem follows. □

Theorem 3 implies that if the consecutive latent variables z^τ, $z^{\tau-1}$ are sufficiently correlated and σ is chosen small enough relative to σ_{min}, then the variance of the vector field that corresponds to our probability path model is lower than that corresponds to the rectified flow model.

Discussions. Let us denote $v_t^\theta(Z) := v_t^\theta(z^\tau, z^{\tau-1}, \xi)$ to show the explicit dependence of the vector field neural net on the random samples t, z^τ, $z^{\tau-1}$ and $\xi \sim \mathcal{N}(0, I)$ drawn during each update of gradient descent during training. During each gradient descent update, our model involves computation of

$$\nabla_\theta \mathcal{L}_{cfm}(\theta; t, \xi, z^\tau, z^{\tau-1}) = 2\nabla_\theta v_t^\theta(z^\tau, z^{\tau-1}, \xi)^T \cdot (v_t^\theta(z^\tau, z^{\tau-1}, \xi) - u_t(Z_t|z^\tau, z^{\tau-1})) \tag{60}$$

$$=: 2\nabla_\theta v_t^\theta(z^\tau, z^{\tau-1}, \xi)^T \cdot \Delta_t^\theta(z^{\tau-1}, z^\tau, \xi), \tag{61}$$

with $t \sim \mathcal{U}[0, 1]$, $\xi \sim \mathcal{N}(0, I)$ and the latent samples $z^\tau, z^{\tau-1}$ drawn randomly.

Similarly, for the rectified flow model let us denote $\tilde{v}_t^\theta(Z) := v_t^\theta(z^{\tau-1}, \eta)$ to show the explicit dependence of the vector field neural net on the random samples t, $z^{\tau-1}$ and $\eta \sim \mathcal{N}(0, I)$ drawn during each update of gradient descent during training. Each update of gradient descent using the rectified flow model involves computation of

$$\nabla_\theta \tilde{\mathcal{L}}_{cfm}(\theta; t, \eta, z^{\tau-1}) = 2\nabla_\theta \tilde{v}_t^\theta(z^{\tau-1}, \eta)^T \cdot (\tilde{v}_t^\theta(z^{\tau-1}, \eta) - \tilde{u}_t(\tilde{Z}_t|z^{\tau-1})) \tag{62}$$

$$=: 2\nabla_\theta \tilde{v}_t^\theta(z^{\tau-1}, \eta)^T \cdot \tilde{\Delta}_t^\theta(z^{\tau-1}, \eta), \tag{63}$$

with $t \sim \mathcal{U}[0, 1]$, $\eta \sim \mathcal{N}(0, I)$ and the latent sample $z^{\tau-1}$ drawn randomly.

Lower gradient variance results in smoother training loss curve and potentially faster convergence, so it is useful to compare the variances of the loss gradient for the two models. However, the variances are highly dependent on $\nabla_\theta v_t^\theta$, $\nabla_\theta \tilde{v}_t^\theta$ and their covariance with the other random vectors appearing in Eq. (60) and Eq. (62), making such comparison challenging without strong assumptions. Heuristically, the difference in the variances of the loss gradient during each update for the two models is primarily determined by the difference between $Var(\Delta_t^\theta(z^{\tau-1}, z^\tau, \xi))$ and $Var(\tilde{\Delta}_t^\theta(z^{\tau-1}, \eta))$ if $\nabla_\theta v_t^\theta$ and $\nabla_\theta \tilde{v}_t^\theta$ are relatively stable. In this case, we have $Var(\Delta_t^\theta(z^{\tau-1}, z^\tau, \xi)) \leq Var(\tilde{\Delta}_t^\theta(z^{\tau-1}, \eta))$ if we suppose the assumptions in Theorem 3, $Var(\tilde{v}_t^\theta) \geq Var(v_t^\theta)$ and $Cov(v_t^\theta, u_t) \geq Cov(\tilde{v}_t^\theta, \tilde{u}_t)$.

The implications of Theorem 3 together with the heuristics above could partially explain why our probability path model leads to smoother loss curve and faster convergence (see Figure 2) compared

to the RIVER method of Davtyan et al. (2023). On the other hand, the dependence of the lower bound in the theorem on σ and σ_{min} suggests that using values of σ that is relatively large enough might not keep the variance of the vector field low, which could partially explain the phenomenon displayed in Figure 2, where using $\sigma = 0.1$ and $\sigma_{min} = 0.001$ leads to large loss fluctuations.

D Additional Empirical Results

In this section, we provide additional experimental results.

Figure 4 provides visual results of the predicted snapshots by our model for each task. The visualizations highlight our model's ability to capture complex flow patterns and dynamics.

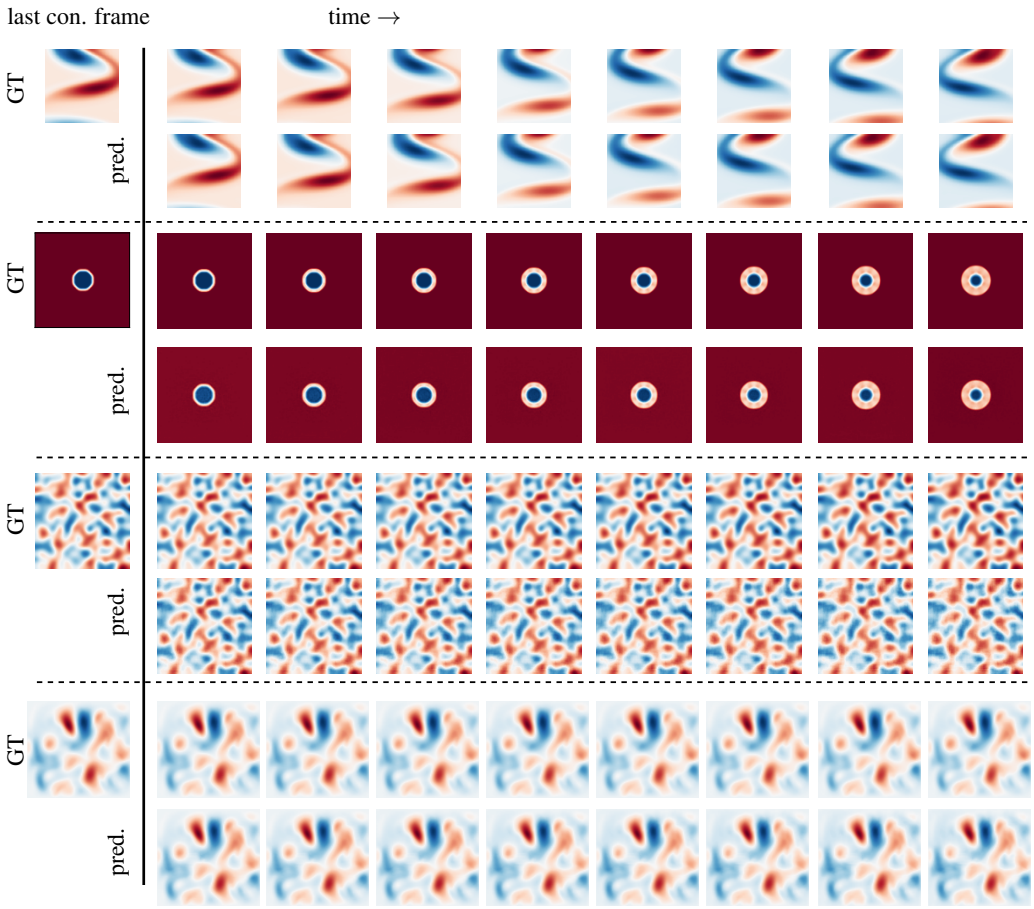

Figure 4: Visualization of predicted frames using our model of probability path for the considered tasks. From top to bottom: fluid flow past a cylinder, shallow-water equation, diffusion-reaction equation, and incompressible Navier-Stokes equation. In each case, GT indicates the ground truth frames and pred. indicates the predicted frames.

E Experimental Details

In this section, we provide the experimental details for the tasks considered in Section 6.

E.1 Details on the Datasets

Fluid flow past a cylinder. We use the fluid flow past a stationary cylinder at a Reynolds number of 100 as a simple test problem. This fluid flow is a canonical problem in fluid dynamics characterized

by a periodically shedding wake structure (Erichson et al., 2020; 2019). The flow dynamics are governed by the two-dimensional incompressible Navier–Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u},$$

$$\nabla \cdot \mathbf{u} = 0,$$

where $\mathbf{u} = (u, v)$ is the velocity field, p is the pressure, ρ is the fluid density, and ν is the kinematic viscosity. The vorticity field ω is obtained from the velocity field via:

$$\omega = \nabla \times \mathbf{u},$$

providing insights into the rotational characteristics of the flow.

For simulating the data, the Immersed Boundary Projection Method (IBPM) has been used (Colonius & Taira, 2008). The flow tensor has dimensions $199 \times 449 \times 151$, representing 151 temporal snapshots on a 449×199 spatial grid. We crop and spatially subsample the data which results in a 64×64 spatial field.

Shallow-water equation. The shallow-water equations, derived from the compressible Navier-Stokes equations, can be used for modeling free-surface flow problems. We consider the 2D equation used in (Takamoto et al., 2022), which is the following system of hyperbolic PDEs:

$$\partial_t h + \nabla h\mathbf{u} = 0, \quad \partial_t h\mathbf{u} + \nabla \left(\mathbf{u}^2 h + \frac{1}{2} g_r h^2 \right) = -g_r h \nabla b, \tag{64}$$

where $\mathbf{u} = u, v$ being the velocities in the horizontal and vertical direction respectively, h describes the water depth, and b describes a spatially varying bathymetry. $h\mathbf{u}$ can be interpreted as the directional momentum components and g_r describes the gravitational acceleration. The mass and momentum conservation properties can hold across shocks in the solution and thus challenging datasets can be generated. This equation finds application in modeling tsunamis and flooding events.

We use the dataset generated and provided by PDEBench (Takamoto et al., 2022). The data file (`2D_rdb_NA_NA.h5`) can be downloaded from `https://github.com/pdebench/PDEBench/tree/main/pdebench/data_download`. The data sample is a series of 101 frames at a 128×128 pixel resolution and come with 1 channel. The simulation considered in (Takamoto et al., 2022) is a 2D radial dam break scenario. On a square domain $\Omega = [-2.5, 2.5]^2$, the water height is initialized as a circular bump in the center of the domain:

$$h(t = 0, x, y) = \begin{cases} 2, & \text{for } r < \sqrt{x^2 + y^2}, \\ 1, & \text{for } r \geq \sqrt{x^2 + y^2}, \end{cases}$$

with the radius r randomly sampled from $\mathcal{U}(0.3, 0.7)$. The dataset is simulated with a finite volume solver using the *PyClaw* package. We apply standardization and then normalization to the range of $[-1, 1]$ to preprocess the simulated data.

Incompressible Navier-Stokes equation. The Navier-Stokes equation is the incompressible version of the compressible Navier-Stokes equation, and it can be used to model hydromechanical systems, turbulent dynamics and weather. We use the inhomogeenous version of the equation (which includes a vector field forcing term $\mathbf{u}$) considered by (Takamoto et al., 2022):

$$\nabla \cdot \mathbf{v} = 0, \quad \rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \Delta \mathbf{v} + \mathbf{u}, \tag{65}$$

where ρ is the mass density, $\mathbf{v}$ is the fluid velocity, p is the gas pressure and η is shear viscosity. The initial conditions $\mathbf{v}_0$ and inhomogeneous forcing parameters $\mathbf{u}$ are each drawn from isotropic Gaussian random fields with truncated power-law decay τ of the power spectral density and scale σ, where $\tau_{v_0} = -3$, $\sigma_{v_0} = 0.15$, $\tau_u = -1$, $\sigma_u = 0.4$. The domain is taken to be the unit square $\Omega = [0, 1]^2$ and the viscosity $\eta = 0.01$. The equation is numerically simulated using *Phiflow*. Boundary conditions are taken to be Dirichlet to clamp the field velocity to zero at the perimeter.

We use the dataset generated and provided by PDEBench (Takamoto et al., 2022). The data file (`ns_incom_inhom_2d_512-0.h5`) can be downloaded from

https://github.com/pdebench/PDEBench/tree/main/pdebench/data_download. The data sample is a series of 1000 frames at a 512×512 pixel resolution and come with 2 channels. We do not apply any data preprocessing procedure here.

Diffusion-reaction equation. We use the 2D extension of diffusion-reaction equation of Takamoto et al. (2022) which describes two non-linearly coupled variables, namely the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. The equation is given by:

$$\partial_t u = D_u \partial_{xx} u + Du \partial_{yy} u + R_u, \tag{66}$$

$$\partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v + R_v, \tag{67}$$

where D_u and D_v are the diffusion coefficient for the activator and inhibitor respectively, $R_u = R_u(u, v)$ and $R_v = R_v(u, v)$ are the activator and inhibitor reaction function respectively. The domain of the simulation includes $x \in (-1, 1)$, $y \in (-1, 1)$, $t \in (0, 5]$. This equation can be used for modeling biological pattern formation.

The reaction functions for the activator and inhibitor are defined by the Fitzhugh-Nagumo equation as: $R_u(u, v) = u - u^3 - k - v$, $R_v(u, v) = u - v$, where $k = 5 \times 10^{-3}$, and the diffusion coefficients for the activator and inhibitor are $D_u = 1 \times 10^{-3}$ and $D_v = 5 \times 10^{-3}$ respectively. The initial condition is generated as standard Gaussian noise $u(0, x, y) \sim \mathcal{N}(0, 1.0)$ for $x \in (-1, 1)$ and $y \in (-1, 1)$. We take a no-flow Neumann boundary condition: $D_u \partial_x u = 0$, $D_v \partial_x v = 0$, $D_u \partial_y u = 0$, and $D_v \partial_y v = 0$ for $x, y \in (-1, 1)^2$.

We use a downsampled version of the dataset provided by PDEBench (Takamoto et al., 2022). The data file (2D_diff-react_NA_NA.h5) can be downloaded from https://github.com/pdebench/PDEBench/tree/main/pdebench/data_download. The data sample is a series of 101 frames at a 128×128 pixel resolution and come with 2 channels. The sample frames are generated using the finite volume method for spatial discretization, and the time integration is performed using the built-in fourth order Runge-Kutta method in the *scipy* package. We do not apply any data preprocessing procedure here.

E.2 Details on Pre-Training the Autoencoder

We provide details on pre-training the autoencoder here. The choice of first pre-training an autoencoder is motivated by the computational challenges of working directly with the high-dimensional spatial resolution of PDE datasets. Training directly in the ambient space requires substantial GPU memory and computational resources, making it impractical for large-scale or high-resolution datasets. By leveraging a latent-space representation, we achieve significant dimensionality reduction while preserving the essential structure of the data, enabling efficient training and inference with standard hardware configurations. For these datasets, latent-space modeling provides a critical balance between computational efficiency, scalability, and performance.

We use the same architecture for the encoder and decoder for all the tasks, with the architecture parameters chosen based on the complexity of the task.

The encoder. The encoder first applies a 2D convolution (conv_in) to the input frame, which reduces the number of channels from in_channels to mid_channels, and processes the spatial dimensions. Then, a series of ResidualBlock layers, which progressively process and downsample the feature map, making it smaller in spatial dimensions but more enriched in terms of features, are applied. After the residual blocks, the feature map undergoes an attention process via a multi-head attention layer. This layer helps the encoder focus on important parts of the input, learning relationships between spatial positions in the image. For the post-attention step, the feature map is further processed by residual blocks and normalized, preparing it for the final convolution. The output of the encoder is obtained by applying a final 2D convolution (out_conv), which maps the processed feature map to the desired number of output channels (out_channels).

The decoder. The decoder takes the encoded feature map and transforms it back into an output with similar spatial dimensions as the input. Similar to the encoder, the decoder starts with a convolution that adjusts the number of channels from in_channels to mid_channels. Then, an attention mechanism (similar to the encoder) is applied to focus on important aspects of the encoded features. Next, a series of ResidualBlock layers, combined with UpBlock layers, are used to progressively increase the spatial dimensions of the feature map (upsampling), undoing the compression applied by

the encoder. After the upsampling, the output is normalized and passed through a final convolution (`out_conv`), mapping the internal feature representation to the desired number of output channels (`out_channels`).

Table 4 summarizes the architecture parameters used for the considered tasks.

Table 4: Parameters chosen for the encoder (decoder) architecture.

Task	Fluid flow	Shallow-water eq.	Navier-Stokes eq.	Diffusion-reaction eq.
`in_channels`	1 (1)	1 (1)	2 (2)	2 (2)
`out_channels`	1 (1)	1 (1)	2 (2)	2 (2)
`mid_channels`	64 (128)	128 (256)	128 (256)	128 (256)

Training details. We train the autoencoder using AdamW with batch size of 32, no weight decay and $\beta = (0.9, 0.999)$. We use the cosine learning rate scheduler with warmup. For the fluid flow past a cylinder task, we train for 2000 epochs and use learning rate of 0.001. For the Navier-Stokes task, we train for 500 epochs and use learning rate of 0.0001. For the other two tasks we train for 5000 epochs and use learning rate of 0.0005. Our implementation is in PyTorch, and all experiments are run on an NVIDIA A100-SXM4 GPU with 40 GB VRAM belonging to an internal SLURM cluster.

E.3 Details on Training the Flow Matching Models

Architecture. The vector field regressor is a transformer-based model designed to process latent vector fields and predict refined outputs with spatial and temporal dependencies. It uses key parameters like `depth` and `mid_depth`, which control the number of transformer encoder layers in the input, middle, and output stages. The `state_size` and `state_res` parameters define the number of channels and spatial resolution of the input data, while `inner_dim` sets the embedding dimension for processing. The model uses learned positional encodings and a time projection to incorporate spatial and temporal context into the input, which can include `input_latents`, `reference_latents`, and `conditioning_latents`. The input is projected into the inner dimension and passed through a series of transformer layers, with intermediate outputs from the input blocks concatenated with the output layers to refine predictions. Finally, the model projects the processed data back to the original spatial resolution and channel size using BatchNorm, producing the final vector field output.

Table 5 summarizes the architecture parameters used for the considered tasks.

Table 5: Parameters chosen for the vector field neural network.

Parameter	Fluid flow	Shallow-water eq.	Navier-Stokes eq.	Diffusion-reaction eq.
`state_size`	4	4	8	4
`state_res`	[8,8]	[16, 16]	[64, 64]	[16, 16]
`inner_dim`	512	512	512	512
`depth`	4	4	4	4
`mid_depth`	5	5	5	5

Training details. For all the considered tasks, we train the regressor using AdamW with batch size of 32, learning rate of 0.0005, no weight decay and $\beta = (0.9, 0.999)$. We use the cosine learning rate scheduler with warmup. For the fluid flow past cylinder, we train for 2000 epochs, for the shallow-water equation and diffusion-reaction task we train for 1000 epochs, and for the Navier-Stokes task we train for 100 epochs. Our implementation is in PyTorch, and all experiments are run on an NVIDIA A100-SXM4 GPU with 40 GB VRAM belonging to an internal SLURM cluster.

E.4 Details on the Evaluation Metrics

In addition to the standard mean squared error and relative Frobenius norm error (RFNE), we use the Pearson correlation coefficient to measure the linear relationship between the forecasted frames

and the target frames. The range of this coefficient is $[-1, 1]$, with zero implying no correlation. Correlations of -1 or $+1$ imply an exact linear relationship. Positive correlations imply that as x increases, so does y. Negative correlations imply that as x increases, y decreases. In addition, we use peak signal-to-noise ratio (PSNR) to evaluate the quality of signal representation against corrupting noise, and structural similarity index measure (SSIM) (Wang et al., 2004) to assess perceptual results. The presented results are computed by averaging over batch size and number of sample generations.

To better quantify probabilistic forecasting performance, we use the Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976) to compare the predicted cumulative distribution function (CDF) to the actual observed value. More precisely, CRPS measures the compatibility of a CDF F with an scalar-valued observation x as:

$$CRPS(F, x) = \int_{\mathbb{R}} (F(z) - 1_{\{x \leq z\}})^2 dz, \tag{68}$$

where $1_{\{x \leq z\}}$ is the indicator function which is one if $x \leq z$ and zero otherwise. CRPS is a proper scoring function, attaining its minimum when the predictive distribution and the data distribution matches. Therefore, a lower CRPS value indicates that the predicted distribution is closer to the true outcome. For high-dimensional arrays (such as forecasts for multiple variables or at multiple spatial locations), the CRPS can be extended by treating the multidimensional forecasts as multivariate distributions. In our experiments, we use 50 ensemble members to compute the empirical CDF, and the CRPS values presented are obtained by averaging over batch size and data dimensions.

E.5 Standard Deviations for the Presented Results

Table 6-10 provide the standard deviation of the results presented in the main paper.

Table 6: Standard deviation results for the fluid flow past a cylinder task using different choices of probability paths for flow matching. Results are averaged over 5 generations.

Model	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)
RIVER	1.33e-03	3.09e-02	1.31	5.93e-03
VE-diffusion	1.75e-02	3.60e-01	3.25e-01	2.31e-02
VP-diffusion	1.75e-02	3.60e-01	3.25e-01	2.31e-02
Stochastic interpolant	5.45e-05	3.51e-02	7.74e-02	3.75e-04
Ours ($\sigma = 0.01$, σ_{sam} = 0, Euler)	6.66e-06	4.49e-03	5.34e-02	3.48e-05

Table 7: Standard deviation for the results of ablation study for the fluid flow past a cylinder task. Results are averaged over 5 generations.

σ	sampler	sampling steps	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)
0.0	Euler	5	3.57e-05	2.42e-02	3.98e-02	2.51e-04
0.01	Euler	5	2.89e-05	2.45e-02	3.01e-02	1.80e-04
0.1	Euler	5	1.64e-05	2.73e-02	2.11e-02	7.21e-05
0.0	Euler	10	9.08e-06	1.25e-02	4.08e-02	5.73e-05
0.01	Euler	10	6.66e-06	4.49e-03	5.34e-02	3.48e-05
0.1	Euler	10	2.84e-05	4.00e-02	3.76e-02	2.21e-04
0.0	RK4	10	3.36e-06	7.69e-03	3.08e-02	2.33e-05
0.01	RK4	10	3.14e-06	5.43e-03	2.56e-02	2.22e-05
0.1	RK4	10	5.88e-05	5.83e-02	2.03e-2	1.85e-04
0.0	Euler	20	6.35e-06	7.36e-03	3.90e-02	5.90e-05
0.01	Euler	20	9.64e-06	1.21e-02	4.30e-02	5.94e-05
0.1	Euler	20	5.44e-06	1.39e-02	1.39e-02	1.03e-04
0.0	RK4	20	1.09e-05	1.65e-02	3.23e-02	8.08e-05
0.01	RK4	20	8.35e-06	1.32e-02	3.29e-02	6.54e-05
0.1	RK4	20	3.44e-06	1.07e-02	1.08e-02	4.08e-05

Table 8: Standard deviation results for the shallow-water equation task using different choices of probability paths for flow matching. Results are averaged over 5 generations.

Model	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)
RIVER	2.81e-05	8.72e-02	1.01e-01	1.29e-03
VE-diffusion	5.01e-04	3.16e-01	3.50e-01	3.25e-03
VP-diffusion	1.56e-04	1.36e-02	2.61e-01	1.06e-01
Stochastic interpolant	7.25e-06	1.03e-01	3.45e-02	4.61e-04
Ours (σ = 0.01, σ_{sam} = 0, RK4)	1.54e-06	7.30e-02	3.54e-03	2.60e-04

Table 9: Standard deviation results for the diffusion-reaction equation task using different choices of probability paths for flow matching. Results are averaged over 5 generations.

Model	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)
RIVER	4.93e-04	2.27e-01	7.01e-01	1.82e-02
VE-diffusion	6.65e-03	1.04	5.59e-01	9.11e-03
VP-diffusion	6.75e-04	5.67e-01	1.62	2.13e-02
Stochastic interpolant	7.92e-04	1.14	9.58e-02	1.20e-03
Ours (σ = 0, $\sigma_{sam} = 0$, RK4)	1.02e-06	4.76e-02	7.96e-03	1.11e-04

Table 10: Standard deviation results for the Navier-Stokes equation task using different choices of probability paths for flow matching. Results are averaged over 5 generations.

Model	Test MSE (↓)	Test RFNE (↓)	PSNR (↑)	SSIM (↑)
RIVER	5.79e-05	2.46e-01	6.14e-01	1.13e-02
VE-diffusion	1.21e-04	1.36	8.82e-02	1.12e-03
VP-diffusion	2.55e-06	1.42e-01	8.28e-02	6.43e-04
Stochastic interpolant	2.03e-07	1.29e-01	9.54e-03	1.95e-04
Ours (σ = 0.01, σ_{sam} = 0, RK4)	5.76e-08	4.88e-02	9.32e-03	7.42e-05