SplineFlow: Flow Matching for Dynamical Systems with B-Spline Interpolants

Santanu Subhash Rathod
CISPA Helmholtz Center for Information Security
santanu.rathod@cispa.de

Pietro Liò
Department of Computer Science and Technology, University of Cambridge
pl219@cam.ac.uk

Xiao Zhang
CISPA Helmholtz Center for Information Security
xiao.zhang@cispa.de

Abstract

Flow matching is an emerging, scalable generative framework for characterizing continuous normalizing flows with wide-range applications. However, state-of-the-art methods are not well-suited for modeling dynamical systems, as they construct conditional paths that are restricted to linear interpolants, a suboptimal supervision signal, which may not capture the system's underlying state evolution. Moreover, constructing unified paths to satisfy multi-marginal constraints across observations is challenging, since naïve higher-order polynomials tend to be unstable and oscillatory. To address these limitations, we introduce SplineFlow, a theoretically grounded flow matching algorithm that jointly models conditional paths across observations via B-spline interpolation. Specifically, SplineFlow exploits the smoothness and stability of B-spline bases to learn the complex underlying dynamics in a structured manner while ensuring the multi-marginal requirements are met. Comprehensive experiments across deterministic and stochastic dynamical systems under various configurations, as well as cellular trajectory inference tasks, demonstrate that SplineFlow outperforms existing baselines, especially when the underlying dynamics are of higher degree or in irregular sampling regimes.

1 Introduction

Flow matching [Lipman et al., 2022] is a generative framework with growing applications across a wide variety of domains, including materials science [Luo et al., 2025], language modeling [Gat et al., 2024], single-cell biology [Rohbeck et al., 2025], and life science [Li et al., 2025]. The wide adoption results from the simplicity, scalability, and generality of its training process compared with alternative frameworks, such as score-based diffusion [Song et al., 2020] and continuous normalizing flows (CNFs) [Chen et al., 2018b, Rubanova et al., 2019]. In particular, unlike CNF-based methods that require expensive simulations, flow matching trains the velocity field of the generative dynamics in a simulation-free manner. At inference time, sampling involves forward ODE integration steps, which are more efficient than the denoising steps of score-based diffusion. Since accurately characterizing the velocity field lies at the heart of modeling dynamical systems, flow matching naturally provides a scalable probabilistic framework for efficiently learning the dynamics of the underlying system.

The training objective of flow matching is a regression loss over conditional velocities, making the construction of conditional paths critical, as they resemble the *supervision signal*. These paths are typically constructed by linearly interpolating observed data points [Lipman et al., 2022, Zhang et al., 2024], which enables efficient generation of high-fidelity samples with far fewer function evaluations

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

[Pooladian et al., 2023]. In these works, the primary emphasis is on the quality of the generated samples rather than on whether the learned dynamics meaningfully capture the underlying ground truth. To faithfully model dynamical systems, the learned neural velocity field must closely match the underlying dynamics throughout the time horizon, rather than merely generating valid samples at observed time points. Therefore, linear interpolants are a restrictive design choice, particularly suboptimal when the underlying dynamics are nonlinear, high-curvature, or contain exponential or sinusoidal terms, necessitating more flexible conditional path designs. However, constructing higher-degree conditional paths that are stable, scalable, and easily integrated with existing frameworks remains challenging, since higher-degree polynomial paths that satisfy all marginal observations tend to be oscillatory, a behavior known as Runge's phenomenon Epperson [1987]. While prior work has improved the vanilla design for modeling cellular dynamics [Tong et al., 2024, Rathod et al., 2025], the emphasis has been on better designs of the coupling scheme while still relying on linear paths.

We present SplineFlow, a flow matching algorithm that constructs conditional probability paths using B-splines (Algorithm 1), a class of spline functions created using the Cox-de Boor recursion formula [De Boor and De Boor, 1978]. Specifically, we explain the limitations of linear conditional paths for modeling dynamical systems and highlight the advantages of B-spline interpolants, including lower approximation errors, greater flexibility, and improved stability (Section 3.3). Theoretically, we show how SplineFlow integrates B-spline conditional paths into the flow matching framework, ensuring the regressed velocity field satisfies the marginal distributions and a more accurate characterization of the underlying dynamics (Section 4). We also extend our spline-based method to model stochastic dynamics with additive noise by regressing on both the drift and the score using simulation-free Schrödinger Bridges (SF2M) [Tong et al., 2023]. Comprehensive experiments across simulations of deterministic and stochastic dynamics with varying complexities, as well as real-world unpaired cellular trajectories, demonstrate the advantages of SplineFlow over baselines, especially pronounced for nonlinear and oscillatory systems or when sampling is irregular (Section 5).

In summary, our contributions are as follows:

- Using B-splines to construct conditional paths, we develop SplineFlow, an efficient and effective flow matching framework for modeling deterministic dynamical systems. We also present a natural SF2M extension of SplineFlow for handling stochastic dynamics.
- By leveraging functional and first-derivative approximation error rates of B-splines, we analyze the theoretical advantages of SplineFlow in terms of asymptotic dynamics approximation error and sample generation quality compared to linear interpolants.
- The improvement of SplineFlow over baselines is most pronounced for nonlinear and oscillatory ODE systems under irregular sampling, whereas for SDEs, the gains are strongest for nonlinear systems. Compared with adjoint-based methods, SplineFlow preserves performance while being significantly faster. For cellular trajectory inference, SplineFlow outperforms baselines under interpolation while remaining competitive under extrapolation.

2 Related Work

Flow Matching. Flow matching [Lipman et al., 2022, Albergo and Vanden-Eijnden, 2022, Liu et al., 2022] is a probabilistic modeling framework for characterizing the generative dynamics by regressing a neural network on tractable conditional velocity fields, which bypasses expensive simulations as required in previous continuous normalizing flow techniques [Chen et al., 2018a, Papamakarios et al., 2021]. Linear velocity fields are typically used in flow matching, as they have been shown to achieve faster training convergence and greater efficiency in function evaluations than other choices, such as diffusion paths [Lipman et al., 2022, Liu et al., 2022]. Notably, linear paths between noise and observations using optimal-transport couplings have resulted in even better performances [Pooladian et al., 2023]. However, it is important to note that these techniques primarily optimize for endpoint generation fidelity and efficiency, and not for faithful characterization of intermediary dynamics.

Score Matching. In score-based diffusion [Song et al., 2020, Nichol and Dhariwal, 2021], stochastic transformations are modeled by reversing a tractable Gaussian-noising process, which results in a score-dependent denoising velocity field. While capable of generating complex, high-dimensional data, its application to modeling stochastic dynamics is limited by its reliance on the Gaussian-noising steps. Stochastically transforming two arbitrary distributions between each other is well known as the Schrödinger Bridge problem [Léonard, 2013, Schrödinger, 1932] with several applications in

generative modeling [De Bortoli et al., 2021, Vargas et al., 2021]. However, the iterative methods for their approximate solutions [De Bortoli et al., 2021, Bunne et al., 2023] suffer from instability and numerical issues, limiting their applicability. To bridge these gaps, Tong et al. [2023] proposed SF2M (score and flow matching), which models the stochastic dynamics by regressing probability flow velocity fields and probability scores over conditional probability paths modeled via Brownian bridges. Similar to flow matching, it again uses linear interpolants to construct the conditional paths.

Modeling Dynamical Systems. Although the underlying state transformations of deterministic systems can be modeled via CNFs [Chen et al., 2018a], these methods require computing an adjoint state via forward simulation at each iteration, which limits scalability. Similar requirements arise when modeling stochastic dynamics [Kidger et al., 2021] or dynamics from irregularly sampled observations [Rubanova et al., 2019]. Flow matching addresses the above scalability issue; however, most efforts on the dynamical front have been limited to solutions for modeling processes underlying unpaired datasets, such as transcriptomic trajectories Li et al. [2025]. For instance, Tong et al. [2024] constructed the conditional paths between temporal transcriptomic samples by linearly interpolating between minibatch optimal transport (OT) couplings, while Rathod et al. [2025] achieved improvements by incorporating biological priors using spatial omics data. Zhang et al. [2024] employed flow matching for modeling dynamical systems, but they still rely on piecewise linear conditional paths, with no provision for faithfully characterizing the intermediary system dynamics. Recently, [Rohbeck et al., 2025] proposed multi-marginal OT-couplings with cubic splines. However, while the method enables smooth interpolation, it is primarily motivated by minimum-curvature considerations rather than faithful approximation of the underlying system dynamics, and lacks rigorous guarantees regarding spline selection and the flexibility to convert to lower-order or higher-order polynomials when needed. Different from these works, SplineFlow is the first method to demonstrate the efficacy of modeling dynamics across degrees of complexity using well-motivated B-spline interpolants.

3 Motivating B-Splines for Dynamic Modeling

3.1 Problem Setup

Let x_t be the state variables of the underlying dynamics at time t governed either by an ODE $dx_t = u_t(x_t)dt$ if the system is deterministic, or by a SDE $dx_t = u_t(x_t)dt + g(t)dw_t$ if stochastic. Here, $u_t(\cdot)$ is the underlying velocity or drift, $g(t)$ stands for the diffusion schedule, and w_t is the standard Wiener process. Let $\{X_i\}_{i=0}^{N-1}$ be a collection of N training trajectories that follow the underlying system dynamics but with different initial values, where $X_i = [x_i(t_0^i), x_i(t_1^i), \ldots, x_i(t_{n_i}^i)]$ is the i-th trajectory, $x_i(t_j^i) \in \mathbb{R}^d$ denotes the j-observation, $x_i(t_0^i)$ stands for the initial value, and $0 = t_0^i < t_1^i < \cdots < t_{n_i}^i = 1$ denote the observed timestamps. We use $p \in [0, 1]$ to denote the degree of sampling irregularity (or sparsity), where a fraction p of observations are randomly masked. The task is to learn a velocity network u_θ (and also a score network s_ϕ for stochastic systems), such that for any testing trajectory X, it minimizes the MSE across any sampled time point t' between the ground-truth observation $x_{t'}$ and the predicted observations by integrating from the initial value x_{t_0}. Formally, if the underlying system is defined by an ODE, the goal is to minimize:

$$\min_\theta \mathbb{E}\left[\left\| x_{t_0} + \int_{t_0}^{t'} u_\theta(x_t, t)dt - x_{t'} \right\|^2\right]. \tag{1}$$

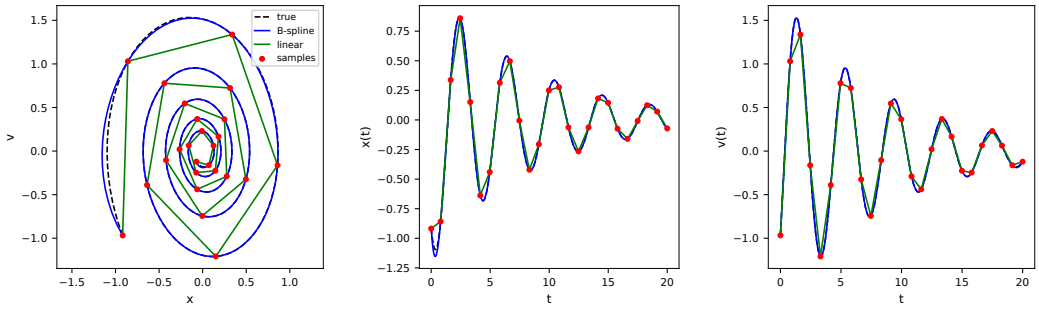
Otherwise, if the underlying system is stochastic and determined by an SDE, we minimize:

$$\min_{\theta,\phi} \mathbb{E}\left[\left\| x_{t_0} + \int_{t_0}^{t'} \left((u_\theta(x_t, t) + \frac{1}{2}g(t)^2 s_\phi(x_t, t))dt + g(t)dw_t\right) - x_{t'} \right\|^2\right]. \tag{2}$$

See Appendix A.3 for detailed derivations of why minimizing Equation 2 is desirable for SDEs.

3.2 Limitation of Linear Interpolants

While flow matching is primarily designed to optimize endpoint data generation (i.e., only considering $t = 1$), it has been extended to model the state evolution of dynamical systems [Zhang et al., 2024, Tong et al., 2024], where observations are made at multiple time points across different trajectories. In prior literature on flow matching, u_θ is typically trained by regressing on conditional velocity fields



(a) Phase Space (b) Position (c) Velocity

Figure 1: Visualizations of interpolated trajectories from a damped harmonic oscillator.

that generate Gaussian conditional paths, using *linear interpolants* for each consecutive observation pair. Specifically, for any $t \in [t_j^i, t_{j+1}^i)$, the target conditional probability path is constructed as:

$$p_t(x|z) = \mathcal{N}\big(x; \mu_t(z), \sigma^2 I\big)\ , \text{ where } \mu_t(z) = \frac{t_{j+1}^i - t}{t_{j+1}^i - t_j^i} x_i(t_j^i) + \frac{t - t_j^i}{t_{j+1}^i - t_j^i} x_i(t_{j+1}^i), \tag{3}$$

where $z = (x_i(t_j^i), x_i(t_{j+1}^i))$ denotes a latent variable, and $\sigma > 0$ is a small constant representing the standard deviation. According to the construction, the associated conditional velocity field takes a simple form of piecewise functions: for any $t \in [t_j^i, t_{j+1}^i)$, $u_t(x|z) = (x_i(t_{j+1}^i) - x_i(t_j^i))/(t_{j+1}^i - t_j^i)$. Flow matching algorithms are then designed to minimize the following regression loss:

$$\min_\theta\ \mathbb{E}_{t\sim\mathcal{U}(0,1), z\sim q(z), x\sim p_t(x|z)} \big\| u_\theta(t,x) - u_t(x|z) \big\|^2, \tag{4}$$

where $\mathcal{U}(0,1)$ is the uniform distribution over $[0,1]$, and $q(z)$ refers to the joint distribution, usually corresponding to either independent [Lipman et al., 2022] or OT-based couplings [Pooladian et al., 2023], for the consecutive time points (t_j, t_{j+1}) such that the sampled t falls inside. Full discussions on the preliminaries of flow matching and their extensions to SDEs are provided in Appendix A.

Since linear probability paths interpolate between observational points, the trained velocity field will induce a marginal distribution that is approximately similar to the observational distribution, thus modeling the underlying dynamics. However, while linear paths connect observations at endpoints, they cannot approximate the intermediary functional values well, and in particular, often miss out on curvature and induce significant bias, as illustrated in Figure 1.

3.3 Advantages of B-Spline Interpolants

B-Splines are widely used to model complex shapes and functions across several domains [Hasan et al., 2024, Biswas and Lovell, 2008, Li, 2020]. Notably, B-splines are known to be stable, smooth, and flexible with respect to the choice of polynomial degree, whereas other alternative choices of higher-order interpolants are prone to oscillatory behavior such as Runge's phenomenon Epperson [1987] and to non-smoothness (see Appendix B for detailed discussions). Let $[x_{t_0}, x_{t_1}, \cdots, x_{t_n}]$ be the $n+1$ points we need to interpolate over. The B-spline interpolant function is built over by recursively combining basis splines using the Cox-De Boor Recursion formula [De Boor and De Boor, 1978]. Specifically, the B-spline bases are defined recursively by:

$$\mathcal{B}_{j,m}(t) = \frac{t - t_j}{t_{j+m} - t_j}\, \mathcal{B}_{j,m-1}(t) + \frac{t_{j+m+1} - t}{t_{j+m+1} - t_{j+1}}\, \mathcal{B}_{j+1,m-1}(t),\ \ \forall m \geq 1, \tag{5}$$

where the 0-th degree B-spline basis is given by $\mathcal{B}_{j,0} = 1$ if $t \in [t_j, t_{j+1})$, and $\mathcal{B}_{j,0} = 0$ otherwise. The interpolant function $\mu(t)$ of degree m is then a linear combination of m-th degree B-splines, such that it recovers the $n+1$ observation points. Formally, $\mu(t)$ is defined as:

$$\mu(t) = \sum_{j=0}^{n} c_{j,m} \mathcal{B}_{j,m}(t), \quad \text{s.t. } \mu(t_j) = x_{t_j},\ \forall\, j \in \{0, 1, \ldots, n\}, \tag{6}$$

where $c_{j,m}$ is determined by solving the linear constraints. Note that $\mu(t)$ is continuous by construction and gets increasingly smooth with the degree of the chosen bases [De Boor and De Boor, 1978].

In practice, for data $x \in \mathbb{R}^d$, we calculate B-spline values at time t separately for each dimension d and concatenate them while using. The following theorem compares the approximation errors, both functional and first-derivative, of higher-degree B-splines with those of linear interpolants.

Theorem 3.1 (Informal)**.** *Assume $f \in \mathcal{C}^m[a,b]$ is a function with m-th bounded derivatives, and we observe the its value at $n+1$ equidistant points $\{t_j\}_{j=0}^n$. Let $\mu(t)$ be the B-spline interpolant of degree m and $p_1(t)$ be the linear interpolant that fits the observations. Then, we have*

$$\|f(t) - \mu(t)\|_\infty = \mathcal{O}(n^{-m}) \quad \textit{and} \quad \|f(t) - p_1(t)\|_\infty = \mathcal{O}(n^{-2}),$$
$$\|f'(t) - \mu'(t)\|_\infty = \mathcal{O}(n^{-m+1}) \quad \textit{and} \quad \|f'(t) - p_1'(t)\|_\infty = \mathcal{O}(n^{-1}).$$

Theorem 3.1 shows that if a function has bounded m-th derivatives, then B-spline interpolants can achieve much lower approximation error compared with linear interpolants. We provide the formal, complete statement and its detailed proof in Appendix D.1. The degree m is chosen based on the nature of the underlying data. For nonlinear dynamical systems, the best-fit is usually greater than 2. While Theorem 3.1 is stated for equidistant or regularly sampled data points, B-splines are known to be stable even in the case of irregular sampling [De Boor and De Boor, 1978, De Boor, 1976] as seen in Lemmas D.3-D.5, which are central results in our proofs and are not limited to regularity.

We conduct preliminary experiments to compare estimates obtained with B-spline and linear interpolants for trajectories sampled from a damped harmonic oscillator (see Appendix C.3 for its formal definition). Figure 1 visualizes the comparison results, where the practical implications of our theoretical results become quite clear: B-splines are much more adept at capturing the underlying curvature compared to linear interpolants. B-splines also retain the flexibility in terms of degree with which to model, unlike cubic interpolants, which are restricted to the cubic degree.

Theorem 3.2 (Informal)**.** *A linear interpolant is a special case of a B-spline interpolant of degree* 1.

Theorem 3.2 with a formal statement is proven in Appendix D.2. When the underlying function is linear in time, B-spline interpolants are expected to recover the performance of linear interpolants.

4 SplineFlow: Flow Matching with B-Spline Interpolation

Motivated by the efficacy, stability, and flexibility of B-splines in Section 3.3, we propose SplineFlow, which constructs conditional paths in flow matching frameworks via B-spline interpolation. Algorithm 1 in Appendix E.1 presents the pseudocode of SplineFlow for both deterministic and stochastic cases.

4.1 Modeling Deterministic Dynamics

When the dynamics are governed by an ODE, we model $p(x|z)$ using Gaussian distributions $\mathcal{N}(x; \mu_t(z), \sigma_t^2 I)$, where $\mu_t(z)$ is replaced by B-spline interpolants. The following theorem, proven in Appendix D.3, derives an error bound with respect to the neural velocity field learned by SplineFlow.

Theorem 4.1 (Informal)**.** *Let $f(t)$ be the ground-truth ODE dynamics of the system. Then, with equidistant observations, the mean Euclidean error between the learned velocity field $u_\theta(t,x)$ from SplineFlow and the true velocity field is upper bounded by:*

$$\mathbb{E}\big[\|u_\theta(t,x) - f'(t)\|\big] \leq \sqrt{\textit{Training loss}} + \mathcal{O}(n^{-m+1}),$$

where m denotes the degree of B-spline functions selected in SplineFlow.

Theorem 4.1 shows that SplineFlow approximates the ground-truth dynamics better than existing linear-interpolant-based approaches with an error term corresponding to $\mathcal{O}(n^{-1})$. In practice, the training loss usually converges to 0 (see Appendix G.2), and thus, better approximation properties of B-splines translate to better conditional paths whose derivatives are regressed over by the neural network u_θ, eventually leading to better prediction of the ground-truth dynamics.

Theorem 4.1 has implications for the sampling or generation process, in which SplineFlow generates samples closer to the ground truth than existing approaches, as shown in the corollary below.

Corollary 4.2 (Informal)**.** *Under the same setting as Theorem 4.1, the distance between ground truth x_t and generated samples $\hat{x}_t$ from a trajectory for SplineFlow at any $t \in [a,b]$ is upper bounded by*

$$\mathbb{E}\big[\|x_t - \hat{x}_t\|\big] \leq (b-a) \times \sqrt{\textit{Training loss}} + \mathcal{O}(n^{-m+1}).$$

We present the formal version of Corollary 4.2 and its detailed proofs in Appendix D.4. The following theorem proves that if we employ the B-spline-based construction, we can ensure that (i) the marginal distributions can be recovered from conditional probability paths, (ii) the conditional velocity field for the dynamic case, analogous to the static case as in Equation 13 satisfies the continuity equation, and (iii) the conditional velocity field that induces the B-spline probability path has an analytical form.

Theorem 4.3 (Informal)**.** *Let $\mu_t(\cdot)$ be the B-spline interpolant of degree m and the conditional probability path be $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t^2 I)$ with $\sigma_t(z) = \sigma \to 0$. Then, the induced marginal p_t recovers the data marginals and satisfies the continuity equation, with a conditional velocity field:*

$$u_t(x|z) = \sum_{j=1}^{n} m \cdot c_{j,m} \left(\frac{\mathcal{B}_{j,m-1}(t)}{t_{j+m} - t_j} - \frac{\mathcal{B}_{j+1,m-1}(t)}{t_{j+m+1} - t_{j+1}} \right).$$

We present the formal version of Theorem 4.3 and its proof in Appendix D.5. According to Theorem 4.3, one can show that $\nabla_\theta \mathcal{L}_{\text{CFM}} = \nabla_\theta \mathcal{L}_{\text{FM}}$ even in the dynamic case, analogous to the static case as shown in Lipman et al. [2022]. This enables us to finally train the network u_θ using the regression loss (Equation 12), with conditional velocity fields corresponding to a B-spline probability path.

4.2 Modeling Stochastic Dynamics

We further extend SplineFlow to the *flow and score matching* (SF2M) framework [Tong et al., 2023] for handling stochastic dynamics (see Appendix A.3 for full preliminaries on SF2M). When the underlying SDE has a constant diffusion schedule $g(t) = \sigma$, we can express the drift $u_t(x_t)$ based on the probability flow velocity field u_t^o and the probability score $\nabla_x \log(p_t(x_t))$:

$$u_t(x_t) = u_t^o(x_t) + \frac{\sigma^2}{2} \nabla_x \log(p_t(x_t)), \tag{7}$$

where $u_t^o(x_t)$ is formally defined in Equation 18. From Theorem 3.2 in Tong et al. [2023], we know that constructing a conditional probability flow $u_t^o(\cdot)$ and its induced probability path $p_t(\cdot)$ satisfying marginal constraints gives us a tractable way to train the probability flow velocity field approximation $u_\theta(t, x)$ and the score approximation $s_\phi(t, x)$ by optimizing over the SF2M regression loss:

$$\mathcal{L}_{\text{SF2M}}(\theta, \phi) = \mathbb{E}_{t,z,x} \Big[\|u_\theta(t, x) - u_t^o(x|z)\|^2 + \lambda(t)^2 \|s_\phi(t, x) - \nabla \log p_t(x|z)\|^2 \Big]. \tag{8}$$

We adapt our SplineFlow framework to construct $p_t(x|z)$ using B-spline interpolation and the induced $u_t^o(x|z)$ for modeling stochastic dynamics. The following proposition, proven in Appendix D.6, states the simplified optimization objective of SplineFlow for modeling SDE dynamics.

Proposition 4.4. *Assume the underlying SDE dynamics have a constant diffusion σ. Let $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t^2(z))$ be a Spline-Bridge with B-spline mean μ_t and standard deviation $\sigma_{t_i} \to 0$ at observed timestamps. Then, for $\lambda(t) = \sigma_t(z)$, the SF2M regression objective can be simplified as:*

$$\mathcal{L}_{\text{SplineFlow}}(\theta, \phi) = \mathbb{E} \Big[\|u_\theta(t, x) - (\epsilon\, \sigma_t' + \mu_t')\|^2 + \|\lambda(t) s_\phi(t, x) + \epsilon\|^2 \Big],$$

where $\epsilon \sim \mathcal{N}(0, I)$, σ_t' and μ_t' stand for the derivatives of σ_t and μ_t.

When modeling stochastic dynamical systems, we observed that the choice of $\sigma_t(z)$ can drastically affect performance. Apart from setting $\sigma_t(z) = \sigma$, we proposed a piecewise quadratic scheme:

$$\sigma_t^2(z) = \frac{\sigma^2}{(t_{j+1} - t_j)^2} (t - t_j)(t_{j+1} - t), \;\; \forall t \in [t_j, t_{j+1}), \tag{9}$$

which empirically works the best when modeling cellular dynamics. Algorithm 1 outlines all the necessary steps used to train the velocity network u_θ and the score network s_ϕ and to obtain the SDE drift term u_t (Equation 7). Together with the diffusion schedule σ, which can be estimated (see Appendix F for detailed procedure), this drift can then be integrated using the Euler-Maruyama SDE discretization scheme [Platen, 1999] to generate SDE trajectories from given initial values.

Table 1: Comparisons of different methods for modeling ODE dynamics in MSE across 6 deterministic systems and varying sampling irregularity p. The best performance is highlighted in bold.

p	Model	Exp.	Harm.	Damp.	LV	Hopper.	Lorenz
0	NeuralODE	9.99e−1	6.54	3.47	1.98e4	2.95	1.49
	LatentODE	1.00e−4	8.20e−4	1.20e−4	**1.56e−3**	1.89	1.16
	MMFM	1.40e−3	1.40e−2	2.00e−3	4.87e−1	1.82e+1	8.02e−1
	TFM	6.60e−4	1.81e−1	1.50e−2	1.84e−1	1.55	2.44
	SplineFlow	**8.90e−5**	**3.70e−4**	**9.50e−5**	8.60e−2	**1.41**	**6.39e−1**
0.25	LatentODE	1.00e−4	1.00e−3	3.40e−4	**9.40e−4**	1.61	–
	MMFM	2.00e−3	1.40e−2	1.80e−3	3.72e−1	3.13e+1	–
	TFM	6.90e−4	1.74e−1	1.50e−2	2.59e−1	1.51	–
	SplineFlow	**8.20e−5**	**6.50e−4**	**3.00e−5**	1.00e−1	**1.43**	–
0.5	LatentODE	1.60e−4	7.60e−4	7.30e−4	**1.55e−3**	**1.65**	–
	MMFM	1.70e−3	1.20e−2	3.00e−3	4.41	4.78e+1	–
	TFM	9.40e−4	1.77e−1	1.60e−2	4.26	1.72	–
	SplineFlow	**1.10e−4**	**5.50e−4**	**4.70e−5**	7.05e−1	1.83	–
0.75	LatentODE	9.60e−5	**7.90e−4**	9.70e−4	**2.90e−3**	3.56	–
	MMFM	2.40e−3	1.00e−2	4.00e−3	8.62	6.11e+1	–
	TFM	1.40e−3	2.05e−1	3.10e−2	1.98	**2.81**	–
	SplineFlow	**8.60e−5**	1.12e−3	**4.80e−5**	1.78	3.34	–
	Avg. Training Runtime (s)						
	LatentODE	2.80e+4	4.47e+4	4.38e+4	4.44e+4	1.21e+5	5.30e+4
	SplineFlow	**1.90e+2**	**2.73e+2**	**2.67e+2**	**2.65e+2**	**2.02e+2**	**2.79e+2**
	Speedup (×)	**147**	**163**	**164**	**167**	**597**	**190**

5 Experiments

We test the performance of SplineFlow on several ODE dynamical systems, including Exponential Decay, Harmonic Oscillator, Damped Harmonic Oscillator, Lotka–Volterra, and Lorenz System, along with their SDE counterparts (see Appendix C for their definitions). In addition, we evaluate SplineFlow on the HopperPhysics dataset from the MuJoCo Physics simulation engine, and on preprocessed, log-normalized longitudinal transcriptomic datasets, including the Embryoid Body temporal data from Moon et al. [2019] and a post-traumatic brain regeneration dataset from Wei et al. [2022], across two-dimensional PHATE and PCA embeddings with 10 principal components (PCs).

For ODE, we compare SplineFlow with adjoint-based NeuralODE [Chen et al., 2018a] and LatentODE [Rubanova et al., 2019], as well as Trajectory Flow Matching (TFM) [Zhang et al., 2024] and Multi-Marginal Flow Matching (MMFM) [Rohbeck et al., 2025]. We benchmark SplineFlow against SF2M [Tong et al., 2023] for SDE and, additionally, against Minibatch-OT Flow Matching (MOTFM) [Tong et al., 2024] for transcriptomics datasets. For ODE evaluation, we compute the mean squared error (MSE) between the predicted trajectories and the ground truth. For SDE, we use several metrics, including the 2-Wasserstein distance ($\mathcal{W}_2$), maximum mean discrepancy (MMD), and energy distance (Energy), which measure statistical fidelity, as well as MSE on SDE and PODE (probability flow ODE) trajectories. For all experiments, we report average statistics over 5 runs.

5.1 ODE Dynamics

Table 1 summarizes the comparisons for ODE dynamics, showing that SplineFlow outperforms the baselines in most settings. Compared to adjoint-based Neural ODE and Latent ODE, simulation-free methods (TFM, MMFM, and SplineFlow) are at least on par in MSE with regularly sampled observations ($p = 0$), if not better, while being significantly (over 100x) faster (see Appendix E.2 for detailed runtime analyses of these algorithms). Higher-degree interpolants in SplineFlow capture the underlying dynamics much better than linear interpolants in TFM, especially for nonlinear and oscillatory systems, such as Lotka–Volterra and the harmonic family. We discuss the experiments on the chaotic Lorenz systems in Section 5.3, and the ablations of B-spline degree in Section 6.

Table 2: Comparisons of SplineFlow and SF2M for modeling SDE dynamics across multiple datasets and metrics, where p denotes the degree of irregularity. The best results are highlighted in bold.

Dataset	p	Model	MSE (SDE)	$\mathcal{W}_2$	MMD
Exp-Decay	0	SF2M	0.477 ± 0.010	$\mathbf{0.305 \pm 0.005}$	$\mathbf{0.024 \pm 0.001}$
		SplineFlow ($m = 1$)	$\mathbf{0.456 \pm 0.008}$	0.331 ± 0.003	0.027 ± 0.002
	0.5	SF2M	$\mathbf{0.524 \pm 0.011}$	0.204 ± 0.012	0.009 ± 0.002
		SplineFlow ($m = 1$)	0.530 ± 0.007	$\mathbf{0.195 \pm 0.001}$	$\mathbf{0.007 \pm 3.3\text{e}{-4}}$
Damped Harmonic	0	SF2M	$\mathbf{0.967 \pm 0.033}$	0.430 ± 0.007	0.048 ± 0.003
		SplineFlow ($m = 1$)	0.970 ± 0.009	$\mathbf{0.410 \pm 0.004}$	$\mathbf{0.042 \pm 0.002}$
	0.5	SF2M	$\mathbf{0.973 \pm 0.025}$	$\mathbf{0.319 \pm 0.007}$	$\mathbf{0.022 \pm 0.003}$
		SplineFlow ($m = 1$)	0.981 ± 0.040	0.321 ± 0.028	0.022 ± 0.006
Lotka–Volterra	0	SF2M	0.461 ± 0.097	0.443 ± 0.050	0.125 ± 0.024
		SplineFlow ($m = 2$)	$\mathbf{0.273 \pm 0.014}$	$\mathbf{0.294 \pm 0.003}$	$\mathbf{0.042 \pm 0.003}$
	0.5	SF2M	0.355 ± 0.031	0.375 ± 0.037	0.080 ± 0.029
		SplineFlow ($m = 3$)	$\mathbf{0.285 \pm 0.025}$	$\mathbf{0.310 \pm 0.012}$	$\mathbf{0.044 \pm 0.004}$
Lorenz	0	SF2M	2.159 ± 0.008	0.632 ± 0.008	0.130 ± 0.003
		SplineFlow ($m = 4$)	$\mathbf{1.227 \pm 0.056}$	$\mathbf{0.180 \pm 0.015}$	$\mathbf{0.005 \pm 0.002}$

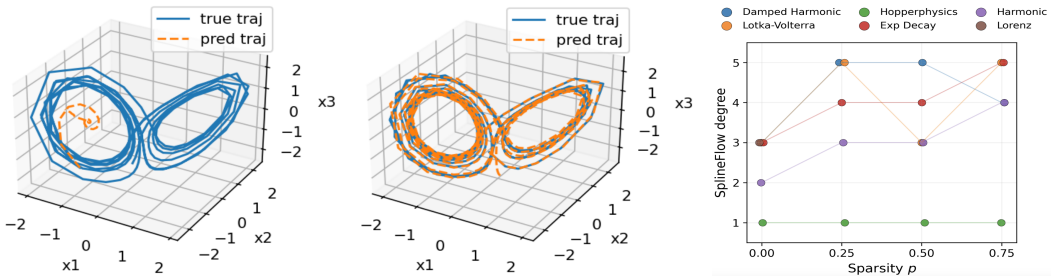
For irregularly sampled observations with $p > 0$, TFM with its linear interpolants performs poorly compared to SplineFlow, as observed not only from the metrics in Table 1 but also from the visualizations of the evolved dynamics in Appendix I. Higher-degree spline bases tend to better capture the underlying behavior under irregular sampling regimes. In scenarios where the underlying dynamics behave approximately linearly, such as in HopperPhysics, MMFM (owing to its nonlinear interpolants) performs poorly compared to SplineFlow, which is on par with TFM. This is expected, since linear interpolants are a special case of B-spline interpolants with $m = 1$ as shown in Theorem 3.2. Figure 7 in Appendix H further visualizes the relative improvement of SplineFlow in a heatmap.

5.2 SDE Dynamics

For SDE dynamical systems with *additive constant diffusion*, we observe that constructing conditional probability paths with a constant variance scheme $\sigma_t(\boldsymbol{z}) = \sigma$ performs better than the piecewise quadratic scheme as introduced in Equation 9. Table 2 summarizes the comparison results across different evaluation metrics between SplineFlow implemented with the constant scheme and the baseline SF2M. For regularly sampled observations, higher-degree B-spline interpolants used to model the conditional probability paths are particularly beneficial for nonlinear Lotka–Volterra and oscillatory damped harmonic systems, which aligns with Theorem 3.1. For irregularly sampled observations, SplineFlow outperforms the baseline for Lotka–Volterra, whereas SF2M with its linear interpolants is the better-performing model in other cases. Due to space limit, we present full SDE comparison results with $p \in \{0.25, 0.75\}$ and across other metrics in Table 14 of Appendix H.

5.3 Lorenz Systems

Chaotic dynamical systems exhibit distinctive evolution dynamics, with drastic shifts in their states at certain timepoints and highly nonlinear trajectories. For Lorenz, in particular, depending on the initial value, the state may converge onto one attractor basin for a period before shifting to another attractor basin, known as the *Butterfly Effect*. This behavior makes the system sensitive to masking and irregular sampling, since the state values at certain critical time points contain information about which attractor the trajectory will approach. Therefore, we restrict the configurations on Lorenz to regular sampling ($p = 0$). Our experiments show that higher-degree SplineFlow not only outperform TFM for the ODE case and SF2M for SDE in terms of metric performance (Tables 1 and 2), but also enable SplineFlow to capture the underlying structure of the dynamics much better than the baselines, including the adjoint-based methods (Figures 2a, 2b and 20–25). This aligns with our theoretical motivation for using SplineFlow to model nonlinear and higher-degree dynamical systems.



(a) TFM (b) SplineFlow (c) Degree vs. Irregularity

Figure 2: (a) and (b) compare the Lorenz ODE trajectories learned by SplineFlow with TFM, whereas (c) presents the ablations of spline degree m on ODE dynamics with varying sampling irregularity p.

Table 3: Comparison results of SplineFlow on transcriptomic datasets with PHATE embedding.

Dataset	Task	Model	MMD	$\mathcal{W}_2$	Energy
Brain Regeneration	Inter.	MOTFM	0.503 ± 0.016	0.794 ± 0.026	1.440 ± 0.062
		SF2M	0.457 ± 0.008	0.753 ± 0.026	1.310 ± 0.036
		SplineFlow ($m = 5$)	$\mathbf{0.285 \pm 0.007}$	$\mathbf{0.607 \pm 0.027}$	$\mathbf{0.815 \pm 0.021}$
	Extra.	MOTFM	1.010 ± 0.012	1.570 ± 0.005	3.890 ± 0.051
		SF2M	0.972 ± 0.009	$\mathbf{1.530 \pm 0.047}$	3.770 ± 0.072
		SplineFlow ($m = 4$)	$\mathbf{0.955 \pm 0.009}$	1.536 ± 0.019	$\mathbf{3.740 \pm 0.042}$
Embryoid Evolution	Inter.	MOTFM	0.060 ± 0.003	0.378 ± 0.011	0.175 ± 0.005
		SF2M	0.025 ± 0.004	0.272 ± 0.024	0.086 ± 0.013
		SplineFlow ($m = 3$)	$\mathbf{0.013 \pm 0.001}$	$\mathbf{0.207 \pm 0.017}$	$\mathbf{0.040 \pm 0.001}$
	Extra.	MOTFM	0.083 ± 0.009	0.448 ± 0.051	0.239 ± 0.007
		SF2M	$\mathbf{0.058 \pm 0.011}$	$\mathbf{0.439 \pm 0.044}$	$\mathbf{0.194 \pm 0.016}$
		SplineFlow ($m = 2$)	0.069 ± 0.002	0.448 ± 0.044	0.207 ± 0.030

6 Further Analysis

Selection of Spline Degree. We analyze the role of spline degree m under varying levels of sampling sparsity. Figure 2c shows that higher spline degrees become increasingly beneficial as p increases for oscillatory and nonlinear ODE systems. This observation is expected, as higher-degree B-splines better capture curvature and higher-order dynamics when observations are limited. In contrast, for approximately linear dynamics (e.g., HopperPhysics), lower degrees ($m = 1$) suffice and perform best, consistent with the fact that linear interpolants are a special case of B-splines (Theorem 3.2). Full ablation results are deferred to Appendix J. These results highlight the importance of adapting the supervision signal in flow matching to the underlying data complexity. Figure 5 in Appendix F shows that a simple cross-validation procedure produces estimates strongly correlated with the optimal degree (Spearman correlation $\rho = 0.85$), suggesting a practical mechanism for selecting the spline degree parameter m in SplineFlow without extensive hyperparameter tuning. We further report the SplineFlow performance when using the cross-validation selection rule (m_{CV}) in Appendix F.

Cellular Dynamics. We evaluate SplineFlow on a Stereo-seq spatial transcriptomics dataset [Wei et al., 2022] from a temporal study of post-traumatic brain regeneration in an Axolotl Telencephalon (Salamander) tissue, as well as on an unpaired transcriptomic Embryoid developmental dataset [Moon et al., 2019]. Both datasets contain unpaired transcriptomic expression profiles across 5 distinct stages, with replicates collected from different individual organisms at each stage, and we consider both interpolation ($t = 3$) and extrapolation ($t = 5$) settings. For SplineFlow, we use static pairwise entropic OT couplings to generate trajectories to define the conditional probability paths. Since SDE modeling of cellular processes has been shown to outperform ODE modeling, as used in MOTFM [Tong et al., 2024], we adopt the stochastic version of SplineFlow.

Table 3 shows that for both datasets and across interpolation and extrapolation settings, SplineFlow and SF2M outperform MOTFM, highlighting the importance of stochastic modeling of cellular processes. Moreover, SplineFlow outperforms SF2M quite comfortably in the interpolation setting, while being best overall in the extrapolation setting for the Stereo-seq dataset. PHATE embedding captures geometric nonlinear structures in transcriptomic data, thereby making SplineFlow better

equipped than SF2M to model temporal processes more accurately. We observe similar trends for experiments with PCA embedding, which contains linear projections unlike PHATE embedding (see Tables 15 and 16 in Appendix H for detailed comparison results). For the Embryoid Evolution dataset, Table 3 shows that SplineFlow outperforms MOTFM and SF2M under the interpolation setting, while remaining competitive under the extrapolation setup.

7 Conclusion

We introduced SplineFlow, a simulation-free flow matching algorithm that constructs conditional paths using B-spline interpolants. SplineFlow provides a scalable and flexible framework for modeling complex dynamical systems, with potential applications ranging from faster data-driven simulations to modeling temporal biological processes, such as longitudinal drug treatment response. Future work may further improve upon this by learning optimal knot placement during training and by extending the framework to model stochastic dynamics with state-dependent diffusion terms.

Availability

Our code is available at: `https://github.com/santanurathod/SplineFlow`.

References

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

George B Arfken, Hans J Weber, and Frank E Harris. *Mathematical methods for physicists: a comprehensive guide*. Academic press, 2011.

Ludwig Arnold. Random dynamical systems. In *Dynamical Systems: Lectures Given at the 2nd Session of the Centro Internazionale Matematico Estivo (CIME) held in Montecatini Terme, Italy, June 13–22, 1994*, pages 1–43. Springer, 2006.

Sambhunath Biswas and Brian C Lovell. B-splines and its applications. In *Bézier and Splines in Image Processing and Machine Vision*, pages 109–131. Springer, 2008.

Robert L Boylestad and Louis Nashelsky. *Electronic devices and circuit theory*. Pearson, 2011.

Charlotte Bunne, Ya-Ping Hsieh, Marco Cuturi, and Andreas Krause. The schrödinger bridge between gaussian measures has a closed form. In *International Conference on Artificial Intelligence and Statistics*, pages 5802–5833. PMLR, 2023.

Kenneth R Butterfield. The computation of all the derivatives of a b-spline basis. *IMA Journal of Applied Mathematics*, 17(1):15–25, 1976.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018a.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018b.

Carl De Boor. A bound on the l_∞-norm of l_2-approximation by splines in terms of a global mesh ratio. *Mathematics of computation*, 30(136):765–771, 1976.

Carl De Boor. Good approximation by splines with variable knots. ii. In *Conference on the Numerical Solution of Differential Equations: Dundee 1973*, pages 12–20. Springer, 2006.

Carl De Boor and Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in neural information processing systems*, 34:17695–17709, 2021.

James F Epperson. On the runge example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

Md Shahid Hasan, Md Nur Alam, Md Fayz-Al-Asad, Noor Muhammad, and Cemil Tunç. B-spline curve theory: An overview and applications in real life. *Nonlinear Engineering*, 13(1):20240054, 2024.

Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*. springer, 2014.

Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

Christian Léonard. A survey of the schrödinger problem and some of its connections with optimal transport. *arXiv preprint arXiv:1308.0215*, 2013.

Liang Li. Application of cubic b-spline curve in computer-aided animation design. *Computer-Aided Design and Applications*, 18(S1):43–52, 2020.

Zihao Li, Zhichen Zeng, Xiao Lin, Feihao Fang, Yanru Qu, Zhe Xu, Zhining Liu, Xuying Ning, Tianxin Wei, Ge Liu, et al. Flow matching meets biology and life science: A survey. *arXiv preprint arXiv:2507.17731*, 2025.

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.

Edward N Lorenz. Deterministic nonperiodic flow 1. In *Universality in Chaos, 2nd edition*, pages 367–378. Routledge, 2017.

Xiaoshan Luo, Zhenyu Wang, Qingchang Wang, Xuechen Shao, Jian Lv, Lei Wang, Yanchao Wang, and Yanming Ma. Crystalflow: a flow-based generative model for crystalline materials. *Nature Communications*, 16(1):9267, 2025.

Tom Lyche and Larry L Schumaker. Local spline approximation methods. *Journal of Approximation Theory*, 15(4):294–325, 1975.

Kevin R Moon, David Van Dijk, Zheng Wang, Scott Gigante, Daniel B Burkhardt, William S Chen, Kristina Yim, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, et al. Visualizing structure and transitions in high-dimensional biological data. *Nature biotechnology*, 37(12): 1482–1492, 2019.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Johan Paulsson. Models of stochastic gene expression. *Physics of life reviews*, 2(2):157–175, 2005.

Eckhard Platen. An introduction to numerical methods for stochastic differential equations. *Acta numerica*, 8:197–246, 1999.

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.

Santanu Subhash Rathod, Francesco Ceccarelli, Sean B Holden, Pietro Liò, Xiao Zhang, and Jovan Tanevski. Contextflow: Context-aware flow matching for trajectory inference from spatial omics data. *arXiv preprint arXiv:2510.02952*, 2025.

Martin Rohbeck, Edward De Brouwer, Charlotte Bunne, Jan-Christian Huetter, Anne Biton, Kelvin Y Chen, Aviv Regev, and Romain Lopez. Modeling complex system dynamics with flow matching across time and conditions. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Erwin Schrödinger. Sur la théorie relativiste de l'électron et l'interprétation de la mécanique quantique. In *Annales de l'institut Henri Poincaré*, volume 2, pages 269–310, 1932.

Larry Schumaker. *Spline functions: basic theory*. Cambridge university press, 2007.

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*, 2020.

Helle Sørensen. Parametric inference for diffusion processes observed at discrete points in time: a survey. *International statistical review*, 72(3):337–354, 2004.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching. *arXiv preprint arXiv:2307.03672*, 2023.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

Lloyd N Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.

Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

Vito Volterra. Variations and fluctuations of the number of individuals in animal species living together. *ICES Journal of marine science*, 3(1):3–51, 1928.

Xiaoyu Wei, Sulei Fu, Hanbo Li, Yang Liu, Shuai Wang, Weimin Feng, Yunzhi Yang, Xiawei Liu, Yan-Yun Zeng, Mengnan Cheng, et al. Single-cell stereo-seq reveals induced progenitor cells involved in axolotl brain regeneration. *Science*, 377(6610):eabp9444, 2022.

Xi Nicole Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modelling. *Advances in Neural Information Processing Systems*, 37:107198–107224, 2024.

A Preliminaries

A.1 Flow Matching

Flow matching (FM) [Lipman et al., 2022] is a technique that characterizes continuous normalizing flows for a system of variables. Let $\psi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ be the underlying continuous flow that acts on random variables at each time $t \in [0, 1]$, which transforms samples x_0 from the source distribution $q_0 = q(x_0)$ into samples x_1 from the target distribution $q_1 = q(x_1)$, so that $\psi(1, x_0) = x_1$. In continuous normalizing flows, ψ can be defined by an ordinary differential equation (ODE):

$$\frac{d\psi(t, x_0)}{dt} = \frac{dx_t}{dt} = u_t(x_t), \tag{10}$$

where $u_t(x) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the velocity field describing the rate of change of the random variables. Transforming the random variable x_t also transforms the associated probability distribution $p_t(x) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}_+$, which satisfies the boundary conditions $p_{t=0} = q(x_0)$ and $p_{t=1} = q(x_1)$. The relationship between the velocity field and its induced probability distribution at time t is given by the well-known *continuity equation* [Villani et al., 2008]:

$$\frac{\partial p_t(x)}{\partial t} = -\nabla_x \cdot \big(u_t(x)p_t(x)\big), \tag{11}$$

where ∇_x denotes the divergence operator. Flow matching approximates the velocity field by a neural network $u_\theta : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ by minimizing a regression loss $\mathcal{L}_{\text{FM}}$:

$$\min_\theta \ \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim p_t(x)} \big\| u_\theta(t, x) - u_t(x) \big\|^2, \tag{12}$$

where $\mathcal{U}(0, 1)$ denotes the uniform distribution over $[0, 1]$, and p_t is the time-varying probability distribution induced by the velocity field u_t. Once trained, the velocity field u_θ can be integrated from 0 to 1 to transform a source sample $x_0 \sim q_0$ into a target sample $x_1 \sim q_1$. In practice, however, the velocity field $u_t(x)$ is unknown, and several velocity fields can exist such that the boundary conditions for the induced probability paths are satisfied.

A.2 Conditional Flow Matching

To tackle the intractability of the flow matching objective $\mathcal{L}_{\text{FM}}$ in Equation 12, conditional flow matching (CFM) was introduced [Lipman et al., 2022] , where the velocity field $u_t(x)$ and the induced probability path $p_t(x)$ are defined as:

$$p_t(x) = \int p_t(x|z)q(z)dz, \ \ u_t(x) = \int u_t(x|z)\frac{p_t(x|z)q(z)}{p_t(x)}dz. \tag{13}$$

Here, z is the conditional variable, and $u_t(x|z)$ denotes the conditional velocity field such that the induced conditional probability paths $p_t(x|z)$ satisfy the boundary conditions: $\int p_{t=0}(x|z)q(z)dz = q_0$ and $\int p_{t=1}(x|z)q(z)dz = q_1$. The CFM regression loss $\mathcal{L}_{\text{CFM}}$ is defined as:

$$\min_\theta \ \mathbb{E}_{t\sim\mathcal{U}(0,1),z\sim q(z),x\sim p_t(x|z)} \big\| u_\theta(t, x) - u_t(x|z) \big\|^2. \tag{14}$$

Lipman et al. [2022] proved that $\nabla_\theta\mathcal{L}_{\text{FM}} = \nabla_\theta\mathcal{L}_{\text{CFM}}$, suggesting that we can regress the velocity field using the more tractable conditional velocities. Therefore, the remaining task is to define an appropriate conditional velocity field $u_t(x|z)$ and the associated $p_t(x|z)$ to sample from.

In the existing literature, the conditional path $p_t(x|z)$ is typically modeled using a Gaussian distribution with $\mu_t(z)$ and $\sigma_t(z)$ being the constructed, latent-dependent, time-varying mean and standard deviation, such that the distribution at a sampled time point matches the observations. For such a particular choice, the associated conditional velocity $u_t(x|z)$ can be expressed in an analytical form:

$$p_t(x|z) = \mathcal{N}(x \mid \mu_t(z), \sigma_t(z)^2), \ \ u_t(x|z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z). \tag{15}$$

A.3 Flow and Score Matching

In certain scenarios, the underlying dynamics of the state random variables x_t can evolve stochastically, corresponding to a stochastic differential equation (SDE):

$$dx_t = u_t(x_t)dt + g(t)dw_t, \tag{16}$$

where $u_t(x) : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the velocity field, $g(t) : [0,1] \to \mathbb{R}_{>0}$ is a positive diffusion function, and w_t stands for the standard Wiener process. The relationship between $u_t(x)$ and the SDE induced probability path $p_t(x) : [0,1] \times \mathbb{R} \to \mathbb{R}_+$ is given by a *Fokker-Planck equation* (FPE):

$$\frac{\partial p_t(x)}{\partial t} = -\nabla_x \cdot (u_t(x)p_t(x)) + \frac{g^2(t)}{2}\nabla_x^2(p_t(x)). \tag{17}$$

Song et al. [2020] showed that starting from some initial distribution $p_0(x)$ and evolving the random variable according to the probability flow ODE with drift $u_t^o(x_t)$:

$$dx_t = \Big[u_t(x_t) - \frac{g^2(t)}{2}\nabla_x \log(p_t(x_t))\Big]dt = u_t^o(x_t)dt, \tag{18}$$

induces the same time varying marginal distribution $p_t(x)$ as the SDE defined by Equation 16. Therefore, given a diffusion schedule $g(t)$, the score function $\nabla_x \log(p_t(x_t))$, and the drift $u_t^o(x_t)$ defined by Equation 18, we can recover the drift for the original SDE in Equation 16 as:

$$u_t(x_t) = u_t^o(x_t) + \frac{g^2(t)}{2}\nabla_x \log(p_t(x_t)). \tag{19}$$

Analogous to conditional flow matching, the drift $u_t^o(x)$ and the score $\nabla_x \log(p_t(x))$ can be approximated by neural networks optimizing the following regression loss over their conditional equivalents (Theorem 3.2 in Tong et al. [2023]):

$$\mathcal{L}_{\text{SF2M}}(\theta,\phi) = \mathbb{E}_{t,z,x}\Big[\|u_\theta(t,x) - u_t^o(x|z)\|^2 + \lambda(t)^2\|s_\phi(t,x) - \nabla \log p_t(x|z)\|^2\Big]. \tag{20}$$

Here, $u_t^o(x|z)$ induces the conditional probability $p_t(x|z)$ and the expectation is taken over uniformly distributed time $t \sim \mathcal{U}(0,1)$, the latent variables $z \sim q(z)$ and the conditional probability samples $x \sim p_t(x|z)$, and $\lambda(t)$ denotes the time conditional weights. When the conditional probability path is constructed as a linear Brownian bridge:

$$p_t(x|z) = \mathcal{N}(x; (1-t)x_0 + tx_1,\ \sigma^2 t(1-t)), \tag{21}$$

where $z = (x_0, x_1)$ is the conditional variable. From Equation 15, we can compute:

$$u_t^o(x|z) = \frac{1-2t}{t(1-t)}\big(x - tx_1 - (1-t)x_0\big) + x_1 - x_0,\ \ \nabla_x \log p_t(x) = \frac{tx_1 + (1-t)x_0 - x}{\sigma^2 t(1-t)}.$$

B Polynomial Interpolants

Let $f : [a,b] \to \mathbb{R}^d$ be an unknown function observed at samples $\{(t_i, x_i)\}_{i=0}^N$, where $x_i = f(t_i)$ and $a \le t_0 < \cdots < t_N \le b$. An interpolation operator $\mathcal{I}$ constructs a continuous-time function $\mathcal{I}f$ such that $(\mathcal{I}f)(t_i) = x_i$ for all i. Below, give formal definitions for different kinds of polynomial interpolants, and discuss their structural properties, as well as limitations (see Table 4 for a summary).

Linear Interpolation. A linear interpolant is a piecewise linear function defined as:

$$(\mathcal{I}_{\text{lin}}f)(t) = x_i\frac{t_{i+1} - t}{t_{i+1} - t_i} + x_{i+1}\frac{t - t_i}{t_{i+1} - t_i},\ \ \text{for } t \in [t_i, t_{i+1}]. \tag{22}$$

It is numerically stable since it depends on local estimations. However, while the resulting interpolant is continuous, its derivatives are discontinuous at sampling points. This makes linear interpolation a suboptimal choice for modeling smooth or higher-order dynamics [Hairer et al., 1993].

Cubic Interpolation. A cubic interpolant $s(t)$ is a polynomial of degree three satisfying

$$s(t_i) = x_i, \qquad s \in C^2([a,b]). \tag{23}$$

In addition to being a continuous function, the first two derivatives ($s'(t)$ and $s''(t)$) are also continuous, making it a popular choice for approximating smooth functions. However, the construction of $s(t)$ depends on all the sampled points $(x_0, \cdots, x_N)$ to ensure the continuity of its first two derivatives or $\mathcal{C}^2[a,b]$, making it extremely sensitive to boundary conditions and reducing locality.

Table 4: Comparison of different interpolation operators and their structural properties.

Interpolant	Mathematical Form	Locality	Smoothness	Derivatives
Linear	Piecewise linear	Local	C^0	Discontinuous
Cubic	Piecewise cubic	Global	C^2	Continuous
Lagrangian	Global polynomial	Global	Smooth	Unstable
Chebyshev	Polynomial in T_k basis	Global	Smooth	Stable
Hermite	Value+Derivative interpolation	Local	C^{1+}	Noise-Sensitive
B-spline	$\sum_k c_k B_{k,m}$	Local	Degree-Controlled	Stable

B.1 Lagrangian Interpolation

A Lagrangian interpolant is a global polynomial defined as:

$$(\mathcal{I}_{\text{Lag}} f)(t) = \sum_{i=0}^{N} x_i \ell_i(t), \qquad \ell_i(t) = \prod_{j \neq i} \frac{t - t_j}{t_i - t_j}. \tag{24}$$

While it is the most natural polynomial satisfying all the boundary conditions, it suffers from Runge's Phenomenon [Epperson, 1987], characterized by large oscillations and instability, making it a poor choice to interpolate between a large number of points N. While piecewise polynomial construction can be done, it comes at the cost of discontinuous derivatives at observational time points.

B.2 Chebyshev Interpolation

Chebyshev interpolation is equivalent to polynomial interpolation at Chebyshev nodes, defined as:

$$t_k = \cos\left(\frac{(2k+1)\pi}{2(N+1)}\right), \tag{25}$$

which gives us Chebyshev interpolants:

$$(\mathcal{I}_{\text{Cheb}} f)(t) = \sum_{k=0}^{N} c_k T_k(t), \tag{26}$$

where T_k are Chebyshev polynomials of the first kind, defined by $T_0(x) = 1$, $T_1(x) = x$, and $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$. Chebyshev Interpolation is known to improve numerical stability compared to other naïve polynomial interpolants, such as the Lagrangian interpolant above [Trefethen, 2019]. However, since the interpolant is constructed globally, it's highly unsuitable for irregularly sampled time points as is common [Trefethen, 2019].

B.3 Hermite Interpolation

A Hermitian interpolant is a generalization of a Lagrangian interpolant, such that apart from satisfying observational values, the derivatives at the observational points are also satisfied:

$$(\mathcal{I}_{\text{Herm}} f)(t_i) = x_i, \qquad (\mathcal{I}_{\text{Herm}} f)'(t_i) = v_i. \tag{27}$$

When accurate derivative information is available, Hermitian interpolants are known to produce smooth and structurally consistent interpolations. However, the quality of the interpolant depends heavily on the availability of derivatives, which are often unavailable. Errors in derivatives lead to poorly fitted interpolants, making them a suboptimal choice [Schumaker, 2007].

B.4 B-Spline Interpolation

A B-spline interpolant of degree m is defined as:

$$(\mathcal{I}_{\text{B}} f)(t) = \sum_{k} c_k B_{k,m}(t), \tag{28}$$

where $\{B_{k,m}\}$ are B-spline basis functions defined over observations via the Cox–de Boor recursion formula (Equation 5). Each basis function is locally defined, ensuring its stability, while the smoothness is controlled by the degree m. This combination of locality and smoothness control makes B-splines a particularly optimal choice for modeling dynamical functions.

Table 5: Summary of the properties of different dynamical systems considered in our experiments.

System	Linear	Non-linear	Oscillatory	Chaotic
Exponential Decay	✓	✗	✗	✗
Harmonic Oscillator	✓	✗	✓	✗
Damped Harmonic Oscillator	✓	✗	✓	✗
Lotka–Volterra	✗	✓	✓	✗
Lorenz System	✗	✓	✗	✓

C Synthetic Dynamical Systems

We provide the definitions of all the synthetic dynamical systems evaluated in our experiments. Tables 5 and 6 summarize the properties, the corresponding ODE and SDE dynamics, and their use cases.

C.1 Exponential Decay

Exponential decay is a simple linear dynamical system, which can be written as the following ODE:

$$\dot{x}(t) = -\lambda x(t), \qquad \lambda > 0. \tag{29}$$

It's often used to model first-order relaxation processes such as radioactive decay or simple stable linear systems [Arfken et al., 2011]. The SDE equivalent of the exponential decay is known as an Ornstein–Uhlenbeck (OU) process given by:

$$dX_t = -\lambda X_t \, dt + \sigma \, dW_t, \tag{30}$$

where W_t denotes the standard Wiener process, λ is the hyperparameter controlling the rate of decay, and σ is the constant exogenous noise rate. OU processes are used to model systems such as the velocity of a Brownian particle under friction, gene-expression fluctuations around the steady state [Paulsson, 2005], among others.

C.2 Harmonic Oscillator

A harmonic oscillator is bistate dynamical system $z(t) = (x(t), v(t))$ with the following ODE:

$$\dot{x}(t) = v(t), \qquad \dot{v}(t) = -\omega^2 x(t), \tag{31}$$

where W_t is the standard Wiener process, and ω is the natural undamped frequency of the system, and σ is the constant exogenous noise rate. It's often used to model oscillations near stable equilibria in systems, such as LC circuits [Boylestad and Nashelsky, 2011] and linearized mechanical vibrations.

C.3 Damped Harmonic Oscillator

A damped harmonic oscillator is a bistate dynamical system with a damping coefficient (γ) compared to a simple harmonic oscillator above, given by the following ODE:

$$\dot{x}(t) = v(t), \qquad \dot{v}(t) = -\omega^2 x(t) - 2\gamma v(t), \tag{32}$$

often used to model damped vibrations in mechanical systems with friction or RLC electronic circuits [Boylestad and Nashelsky, 2011] with resistance, among others. The stochastic equivalent is also known as a Langevin System, written as:

$$dX_t = V_t \, dt, \qquad dV_t = \left(-\omega^2 X_t - 2\gamma V_t\right) dt + \sigma \, dW_t, \tag{33}$$

W_t is the standard Wiener process, and ω is the natural undamped frequency of the system, γ is the damping coefficient, and σ is the constant exogenous noise rate. It is used to model the thermal fluctuations in mechanical systems, or the motion of a Brownian particle in a quadratic potential.

Table 6: Full mathematical definition and descriptive summary of the considered dynamical systems.

Dynamical System	ODE	SDE (additive noise)	Model Use Case
Exponential Decay	$\dot{x}(t) = -\lambda x(t)$	$dX_t = -\lambda X_t\,dt + \sigma\,dW_t$	First-order relaxation processes radioactive decay, simple stable linear systems mean-reverting stochastic dynamics (OU)
Harmonic	$\dot{x}(t) = v(t)$ $\dot{v}(t) = -\omega^2 x(t)$	$dX_t = V_t\,dt$ $dV_t = -\omega^2 X_t\,dt + \sigma\,dW_t$	Oscillations near stable equilibria LC circuits, linearized mechanical vibrations
Damped Harmonic	$\dot{x}(t) = v(t)$ $\dot{v}(t) = -\omega^2 x(t) - 2\gamma v(t)$	$dX_t = V_t\,dt$ $dV_t = (-\omega^2 X_t - 2\gamma V_t)\,dt + \sigma\,dW_t$	Damped mechanical vibrations RLC circuits, underdamped Langevin dynamics
Lotka–Volterra	$\dot{x}(t) = \alpha x(t) - \beta x(t) y(t)$ $\dot{y}(t) = \delta x(t) y(t) - \gamma y(t)$	$dX_t = (\alpha X_t - \beta X_t Y_t)\,dt + \sigma\,dW_t^{(1)}$ $dY_t = (\delta X_t Y_t - \gamma Y_t)\,dt + \sigma\,dW_t^{(2)}$	Predator–prey population dynamics, ecological systems chemical reaction kinetics, resource–consumer models
Lorenz	$\dot{x}(t) = \sigma\big(y(t) - x(t)\big)$ $\dot{y}(t) = x(t)\big(\rho - z(t)\big) - y(t)$ $\dot{z}(t) = x(t)y(t) - \beta z(t)$	$dX_t = \sigma\big(Y_t - X_t\big)\,dt + \eta\,dW_t^{(1)}$ $dY_t = \big(X_t(\rho - Z_t) - Y_t\big)\,dt + \eta\,dW_t^{(2)}$ $dZ_t = \big(X_t Y_t - \beta Z_t\big)\,dt + \eta\,dW_t^{(3)}$	Canonical nonlinear and chaotic system sensitivity to initial conditions benchmark for learning chaotic dynamics

C.4 Lotka–Volterra System

A Lotka–Volterra System is also known as a predator-prey dynamical system with prey populations $x(t)$ and predator populations $y(t)$, which can be written as the following ODE:

$$\dot{x}(t) = \alpha x(t) - \beta x(t) y(t), \qquad \dot{y}(t) = \delta x(t) y(t) - \gamma y(t). \tag{34}$$

It is often used to model simple predator-prey population dynamics [Volterra, 1928] in ecological environments, chemical reaction kinetics, and resource-consumer models. The stochastic version is used to model the influence of the outside environment in the form of perturbations, given by:

$$dX_t = \big(\alpha X_t - \beta X_t Y_t\big)\,dt + \sigma\,dW_t^{(1)}, \qquad dY_t = \big(\delta X_t Y_t - \gamma Y_t\big)\,dt + \sigma\,dW_t^{(2)}, \tag{35}$$

where W_t is the standard Wiener process, α is the prey growth rate in the absence of predators, β is the rate at which predators comsume prey, γ is the death rate of predators in the absence of prey to feed over, and δ is the rate at which the consumed prey converted into predator population, and σ is the constant exogenous noise rate.

C.5 Lorenz System

The Lorenz system was originally used to model simplified thermal convection dynamics in the atmosphere [Lorenz, 2017] and was shown to exhibit interesting properties, including sensitivity to initial conditions, deterministic chaotic behavior, and strange attractor patterns. It's now considered a canonical example [Arnold, 2006] in non-linear and chaotic dynamical system studies. The state evolution can be cast into an ODE as follows:

$$\dot{x}(t) = \sigma\big(y(t) - x(t)\big), \qquad \dot{y}(t) = x(t)\big(\rho - z(t)\big) - y(t), \qquad \dot{z}(t) = x(t)y(t) - \beta z(t). \tag{36}$$

A standard additive-noise variant is used to incorporate modeling error and exogenous influences. The stochastic Lorenz system can be written as:

$$\begin{aligned} dX_t &= \sigma\big(Y_t - X_t\big)\,dt + \eta\,dW_t^{(1)}, \\ dY_t &= \big(X_t(\rho - Z_t) - Y_t\big)\,dt + \eta\,dW_t^{(2)}, \\ dZ_t &= \big(X_t Y_t - \beta Z_t\big)\,dt + \eta\,dW_t^{(3)}. \end{aligned} \tag{37}$$

where $W_t^{(i)}$ are the standard Wiener processes, and (σ, ρ, β) are system-dependent model parameters controlling coupling strengths and dissipation rates, and η is the constant exogenous noise rate.

C.6 Hopper Physics (MuJoCo)

Hopper Physics is a dataset generated using the dm-control MuJoCo physics simulation library, which models the locomotion of a rigid-legged system governed by gravity and rigid-body mechanics. Let $q(t)$ be the generalized coordinates of the joints and $\dot{q}(t)$ be their generalized velocities. A standard form can be written as:

$$M(q)\ddot{q} + C(q, \dot{q}) + g(q) = \tau + J(q)^\top \lambda, \tag{38}$$

where M is the mass matrix, C collects Coriolis/centrifugal terms, g gravity, τ actuator torques, and $J^\top \lambda$ enforces constraints/contacts. Equivalently in first-order state $s(t) = (q(t), \dot{q}(t))$:

$$\dot{s}(t) = \begin{bmatrix} \dot{q}(t) \\ M(q)^{-1}\big(\tau + J(q)^\top \lambda - C(q, \dot{q}) - g(q)\big) \end{bmatrix}. \tag{39}$$

D Proofs of Main Theoretical Results

D.1 Proof of Theorem 3.1

Theorem 3.1 (Formal). *Let f be a function in $\mathcal{C}^m[a, b]$ with bounded m-th derivative $\|f^{(m)}\|_\infty \ll \infty$, and let $t_i = a + (i-1)h$ be $n+1$ equidistant points to be interpolated, with the distant between consecutive points being $h = \frac{b-a}{n}$. Then, assuming a well defined B-Spline interpolant with knots $\{t_i\}_{i=0}^n$, $\mu(t) = \sum_{i=0}^n c_{i,m}\mathcal{B}_{i,m}(t)$, where $\mathcal{B}_{i,m}$ is the B-Spline polynomial of degree m as defined in Equation 5, has the asymptotic function approximation error $\mathcal{O}(n^{-m})$ compared to the linear interpolant $\mathcal{O}(n^{-2})$, defined as $p_1(t) = f(t_i) + \frac{f(t_{i+1})-f(t_i)}{t_{i+1}-t_i}(t - t_i)$ for $t \in [t_i, t_{i+1}]$.*

Similarly, the approximation error between the first derivatives of the interpolants and the underlying function is $\mathcal{O}(n^{-m+1})$ for the B-Spline interpolant and $\mathcal{O}(n^{-1})$ for the linear interpolant. Formally,

- $\|f(t) - \mu(t)\|_\infty \leq (1 + \|I\|)\, C^1_{m,a,b}\, n^{-m}\, \|f^{(m)}\|_\infty$,
- $\|f(t) - p_1(t)\|_\infty \leq \frac{(b-a)^2}{8}\|f''\|_\infty \cdot n^{-2}$,
- $\|f'(t) - \mu'(t)\|_\infty \leq C^2_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty$,
- $\|f'(t) - p_1'(t)\|_\infty \leq (b-a)\, n^{-1}\, \|f''\|_\infty$,

where $\|I\| = \sup\left\{\frac{\|If\|}{\|f\|} : v \in \mathcal{C}[a, b] \setminus \{0\}\right\}$ and $C_{m,a,b}$ is a constant depending only on degree m and extremums a, b and C_m is a constant depending only on degree m.

Proof. For the first two function approximation bounds, the proof essentially boils down to combining the results from the first part of Corollary D.2 and Lemma D.4 containing the function approximation error rates for the linear interpolant $p_1(t)$ and B-Spline iterpolant $\mu(t)$.

From Corollary D.2 we can write,

$$\|f(t) - p_1(t)\|_\infty \leq \frac{1}{8}\Delta^2_{\max}|f''(t)|_\infty. \tag{40}$$

Here, $\Delta_i = t_{i+1} - t_i$ and $\Delta_{\max} = \max_i \Delta_i$, and for equidistant points, $\Delta_{\max} = \frac{b-a}{n}$ which after substituting in the above Equation 40 gives us the required result for linear interpolation:

$$\|f(t) - p_1(t)\|_\infty \leq \frac{(b-a)^2}{8}\|f''\|_\infty \cdot (n)^{-2}, \tag{41}$$

Similarly, knowing that $\Delta_{\max} = \frac{b-a}{n}$ for equidistant points and utilizing result from Lemma D.1 gives us,

$$\|f(t) - \mu(t)\|_\infty \leq (1 + \|I\|)\, C^1_{m,a,b}\, n^{-m}\, \|f^{(m)}\|_\infty. \tag{42}$$

Using similar arguments for results from Lemma D.5 and second part of Corollary D.2, and substituting $\Delta_{\max} = \frac{b-a}{n}$ in the upper bounds we have that,

$$\|f'(t) - \mu'(t)\|_\infty \leq C^2_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty, \tag{43}$$

and

$$\|f'(t) - p_1'(t)\|_\infty \leq (b-a)\, n^{-1}\, \|f''\|_\infty. \tag{44}$$

From Equations 41-44, we get the required result. □

D.2 Proof of Theorem 3.2

Theorem 3.2 (Formal). *Let $\mu(t)$ be a well defined B-Spline interpolant, interpolanting function $f(t) \in \mathcal{C}^2[a,b]$ with observation sites at $\{t_i\}_{i=0}^n$, $\mu(t) = \sum_{i=0}^n c_{i,m}\mathcal{B}_{i,m}(t)$, where $\mathcal{B}_{i,m}$ is the B-Spline polynomial of degree m as defined in Equation 5. Then, degree 1 interpolant or μ_t with $m = 1$ is equivalent to the linear interpolant defined as:*

$$p_1(t) = f(t_i) + \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i}(t - t_i) \text{ for any } t \in [t_i, t_{i+1}].$$

Proof. We clamp the observation sites to get $n+3$ knots: $\tau_0 = \tau_1 < \tau_2 \cdots \tau_n < \tau_{n+1} = \tau_{n+2}$ with $\tau_0 = \tau_1 = t_0$, $\tau_{n+1} = \tau_{n+2} = t_n$, and $\tau_i = t_{i-1}$ for $i \in [1, n+1]$.

From the definition of B-Splines in Section 3.3, the 0-th degree B-Spline basis is defined as:

$$\mathcal{B}_{i,0} = \begin{cases} 1 & \text{if} \quad t \in [\tau_i, \tau_{i+1}), \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

With the higher degree B-Spline basis defined recursively as:

$$\mathcal{B}_{i,m}(t) = \frac{t - \tau_i}{\tau_{i+m} - \tau_i}\mathcal{B}_{i,m-1}(t), + \frac{\tau_{i+m+1} - t}{\tau_{i+m+1} - \tau_{i+1}}\mathcal{B}_{i+1,m-1}(t). \tag{46}$$

The interpolant function $\mu(t)$ of degree m is then a linear combination of B-Splines around $n+3$ knots, written as:

$$\mu(t) = \sum_{i=0}^{n+2} c_{i,m}\mathcal{B}_{i,m}(t) \quad \text{s.t.} \quad \mu(t_0) = f(t_0),\ \mu(t_1) = f(t_1),\ \cdots,\ \mu(t_n) = f(t_n). \tag{47}$$

Since our degree $m = 1$, we thus focus on getting expressions for $\mathcal{B}_{i,1}$, starting with the left and right endpoints. From Cox-De Boor recursion and Equation 46, we know that

$$\mathcal{B}_{0,1}(t) = \frac{t - \tau_0}{\tau_1 - \tau_0}\mathcal{B}_{0,0}(t) + \frac{\tau_2 - t}{\tau_2 - \tau_1}\mathcal{B}_{1,0}(t).$$

Since $\tau_0 = \tau_1 = t_0$ we have that $\mathcal{B}_{0,0} \equiv 0$, so we get that $\mathcal{B}_{0,1}(t) = \frac{t_1 - t}{t_1 - t_0}\mathcal{B}_{1,0}$, which gives us,

$$\mathcal{B}_{0,1}(t) = \begin{cases} \frac{t_1 - t}{t_1 - t_0}, & t \in [t_0, t_1), \\ 0, & \text{otherwise.} \end{cases} \tag{48}$$

Similarly for the right endpoint, we have that $\tau_{n+1} = \tau_{n+2} = t_{n+1}$ which implies $\mathcal{B}_{n+2,0} \equiv 0$. And using this in the Cox-De Boor recursion formula (Equation 46) gives

$$\mathcal{B}_{n+2,1}(t) = \begin{cases} \frac{t - t_n}{t_{n+1} - t_n}, & t \in [t_{n-1}, t_n], \\ 0, & \text{otherwise.} \end{cases} \tag{49}$$

For $i \in [1, n+1]$ we have that

$$\mathcal{B}_{i,1}(t) = \frac{t - \tau_i}{\tau_{i+1} - \tau_i}\mathcal{B}_{i,0}(t) + \frac{\tau_{i+2} - t}{\tau_{i+2} - \tau_{i+1}}\mathcal{B}_{i+1,0}(t), \tag{50}$$

and from Equation 45 we know that $\mathcal{B}_{i,0}(t) = 1$ in $t \in [\tau_i, \tau_{i+1})$, $\mathcal{B}_{i+1,0}(t) = 1$ in $t \in [\tau_{i+1}, \tau_{i+2})$, and utilizing the fact that $\tau_i = t_{i-1}$ for $i \in [1, n+1]$, gives us

$$\mathcal{B}_{i,1}(t) = \begin{cases} \frac{t - t_{i-1}}{t_i - t_{i-1}}, & t \in [t_{i-1}, t_i) \\ \frac{t_{i+1} - t}{t_{i+1} - t_i}, & t \in [t_i, t_{i+1}) \\ 0 \end{cases} \tag{51}$$

We can now use the constraints from Equation 47 and Equations 48, 51, 49 to solve and get,

$$c_{0,1} = c_{1,1} = f(t_0), c_{2,1} = f(t_1), \cdots, c_{n,1} = f(t_{n-1}), c_{n+1,1} = c_{n+2,1} = f(t_n). \tag{52}$$

Finally, to prove the equivalence with the linear interpolant, consider the B-Spline interpolant $\mu(t)$ for $t \in [t_i, t_{i+1}]$. We know that

$$\mu(t) = \sum_{i=0}^{n+2} c_{i,m} \mathcal{B}_{i.m}(t).$$

Utilizing Equations 51 and 52, we get that

$$\begin{aligned}
\mu(t) &= f(t_i)\frac{t_{i+1} - t}{t_{i+1} - t_i} + f(t_{i+1})\frac{t - t_i}{t_{i+1} - t_i} \\
&= \frac{f(t_i)(t_{i+1} - t_i) + f(t_i)t_i - f(t_{i+1})t_i + t(f(t_{i+1}) - f(t_i))}{t_{i+1} - t_i} \\
&= f(t_i) + \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i} \times (t - t_i),
\end{aligned}$$

which gives

$$\mu(t) = p_1(t) = f(t_i) + \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i} \times (t - t_i).$$

Thus, we have proven the equivalence between the B-Spline interpolant with $m = 1$ and $p_1(t)$. □

D.3 Proof of Theorem 4.1

Theorem 4.1 (Formal). *Let $f \in \mathcal{C}^m[a, b]$ denote the true underlying dynamics with bounded m-th derivative $\|f^{(m)}\|_\infty < \infty$, and let $\mu(t) = \sum_{i=0}^{n} c_{i,m}\mathcal{B}_{i,m}(t)$ be the degree-m B-Spline interpolant with equidistant knots $\{t_i\}_{i=0}^n$ and mesh spacing $h = \frac{b-a}{n}$. Let $u_\theta(t, x)$ be the velocity field learned by minimizing the flow matching regression objective (Equation D.6), and let $p_1(t)$ be the linear interpolant as defined in Theorem 3.2. Then, choosing the conditional probability path as $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma^2)$ with constant σ, the mean Euclidean error between the learned velocity field and the true dynamics decomposes as:*

$$\mathbb{E}_{t,x}[\|u_\theta(t, x) - f'(t)\|] \leq \underbrace{\sqrt{\mathbb{E}_{t,x}[\|u_\theta(t, x) - \mu'(t)\|^2]}}_{\textit{regression error}} + \underbrace{C_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty}_{\textit{approximation error}},$$

where $C_{m,a,b}$ is a constant depending only on degree m and the interval endpoints a, b. In particular, the approximation error is $\mathcal{O}(n^{-m+1})$ for the degree-m B-Spline interpolant compared to $\mathcal{O}(n^{-1})$ for the linear interpolant.

Proof. We proceed in two steps: we first establish the derivative-approximation error, then connect it to the flow-matching objective via the conditional velocity field, thereby yielding the decomposition.

Step 1: Derivative approximation error. From Theorem 3.1 and by setting substituting $\Delta_{\max} = h = \frac{b-a}{n}$ for the equidistant knot sequence, we know that,

$$\|f'(t) - \mu'(t)\|_\infty \leq C^2_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty. \tag{53}$$

Similarly, from Theorem 3.1, we also have the linear interpolant derivative error

$$\|f'(t) - p_1'(t)\|_\infty \leq (b - a)\, n^{-1}\, \|f''\|_\infty. \tag{54}$$

Step 2: Error Decomposition. From Equation 15 or Theorem 3 from Lipman et al. [2022], we know that for the conditional probability path $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma^2)$ with constant σ, the conditional velocity field reduces to

$$u_t(x \mid z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z) = \mu_t'(z), \tag{55}$$

where the last equality follows from $\sigma_t'(z) = 0$. Thus, the flow matching regression objective trains u_θ to approximate $\mu'(t)$, which is itself an approximation to the true dynamics $f'(t)$.

Applying the triangle inequality, we can write

$$\|u_\theta(t,x) - f'(t)\| \le \|u_\theta(t,x) - u_t(x \mid z)\| + \|u_t(x \mid z) - f'(t)\|_\infty,$$
$$\|u_\theta(t,x) - f'(t)\| \le \|u_\theta(t,x) - \mu'(t)\| + \|\mu'(t) - f'(t)\|_\infty,$$

where the second equation follows from Equation 55. Substituting the derivative approximation bound from Equation 53 into the second term yields

$$\|u_\theta(t,x) - f'(t)\| \le \|u_\theta(t,x) - \mu'(t)\| + C^2_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty.$$

Finally, taking expectation on both sides with respect to x, t and utilizing Jensen's inequality ($\mathbb{E}[\|X\|] \le \mathbb{E}([\|X\|^2])^{1/2}$) gives us the required result:

$$\mathbb{E}_{t,x}[\|u_\theta(t,x) - f'(t)\|] \le \sqrt{\mathbb{E}_{t,x}[\|u_\theta(t,x) - \mu'(t)\|^2]} + C^2_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty.$$

The first term is the regression error, controlled by the flow matching training objective, and the second term is the approximation error, determined by the spline degree and the number of observation sites. Since the analogous decomposition for the linear interpolant gives an approximation error of $\mathcal{O}(n^{-1})$ from Equation 54, increasing the spline degree m strictly reduces the approximation component of the bound, thereby pushing the learned velocity field u_θ closer to the true dynamics $f'(t)$. □

D.4 Proof of Corollary 4.2

Corollary 4.2 (Formal). *Assume similar assumptions as in Theorem 4.1. Let the ground truth and generated samples be defined as:*

$$x_t = x_0 + \int_0^t f'(s)\, ds, \qquad \hat{x}_t = x_0 + \int_0^t u_\theta(s, \hat{x}_s)\, ds.$$

Then, for any $t \in [a,b]$ (including times $t \notin \{t_i\}_{i=0}^n$ not in the training data), the sampling error satisfies

$$\mathbb{E}[\|x_t - \hat{x}_t\|] \le (b-a)\left(\mathbb{E}[\|u_\theta - \mu'\|] + C_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty\right).$$

In particular, the approximation component is $\mathcal{O}(n^{-m+1})$ for the degree-m B-Spline interpolant compared to $\mathcal{O}(n^{-1})$ for the linear interpolant.

Proof. Since both trajectories share the same initial condition x_0, we can write

$$x_t - \hat{x}_t = \int_0^t \left[f'(s) - u_\theta(s, \hat{x}_s)\right] ds. \tag{56}$$

Taking absolute values and bounding $|t| \le b - a$ gives

$$\|x_t - \hat{x}_t\| \le (b-a)\, \|f' - u_\theta\|. \tag{57}$$

The result now follows directly from Theorem 4.1, which gives

$$\mathbb{E}[\|f' - u_\theta\|] \le \mathbb{E}[\|u_\theta - \mu'\|] + C_{m,a,b}\, n^{-m+1}\, \|f^{(m)}\|_\infty.$$

Substituting into Equation 57 yields the stated bound. The comparison with the linear interpolant follows from Corollary D.2, which gives an approximation error of $\mathcal{O}(n^{-1})$. □

D.5 Proof of Theorem 4.3

Theorem 4.3 (Formal). *Let $\mu(t)$ be the degree m B-Spline interpolant for trajectories $X = [x_{t_0}, x_{t_1}, \cdots, x_{t_n}]$ as defined in Equation 5, and let the latent be $z = (x_{t_0}, x_{t_1}, \cdots, x_{t_n})$ with the distribution $q(z)$ over observed trajectories. Then, choosing the conditional probability path as $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t^2(z))$ with $\sigma_t(z) = \sigma \to 0$, enables the following:*

- *The marginal probability distribution written as, $p_t(x) = \int p_t(x|z)q(z)dz$ satifies the observation distribution with $p_{t=t_i}(x) = q(x_{t_i} = x)$.*

- *The marginal velocity field written as* $u_t(x) = \int u_t(x \mid z) \frac{p_t(x|z)\, q(z)}{p_t(x)}\, dz$ *and the marginal probability* $p_t(.)$ *obey the continuity equation* $\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (u_t(x) p_t(x))$.
- *The conditional probability path defined above is generated by the conditional velocity field given by* $u_t(x \mid z) = \sum_{i=1}^{n} m \times c_{i,m} \Big\{ \frac{\mathcal{B}_{i,m-1}(t)}{t_{i+m} - t_i} - \frac{\mathcal{B}_{i+1,m-1}(t)}{t_{i+m+1} - t_{i+1}} \Big\}$.

Proof. (A). We know that $\lim_{\sigma \to 0} \mathcal{N}(x; \mu(t \mid z), \sigma^2 I) = \delta(x - \mu(t \mid z))$, where δ is the delta distribution. For $t = t_i$ we can thus write

$$p_{t=t_i}(x) = \int \delta(x - \mu(t_i \mid z) q(z) dz.$$

Since $\mu(.)$ is a B-Spline interpolant, we have that $\mu(t_i \mid z) = x_{t_i}$. Thus, we have

$$\begin{aligned}
p_{t=t_i}(x) &= \int \delta(x - x_{t_i}) q(z) dz \\
&= \int \cdots \int \delta(x - x_{t_i}) q(x_{t_0}, x_{t_1}, \cdots, x_{t_n}) dx_{t_0} \cdots dx_{t_n} \\
&= \int \delta(x - x_{t_i}) \Big(\int \cdots \int q(x_{t_0}, x_{t_1}, \cdots, x_{t_n}) dx_{t_0} \cdots dx_{t_n} \Big) dx_{t_i} \\
&= \int \delta(x - x_{t_i}) q(x_{t_i}) dx_{t_i} = q(x = x_{t_i}),
\end{aligned}$$

which gives us the required result.

(B). This result follows from a simple extension of Theorem 1 from Lipman et al. [2022]. We can write the marginal p_t as $p_t(x) = \int p_t(x|z)\, q(z)\, dz$, which gives us

$$\frac{\partial p_t(x)}{\partial t} = \frac{\partial}{\partial t} \int p_t(x|z)\, q(z)\, dz.$$

Invoking the Continuity Equation (from Equation 11) for the conditional probability, we get

$$\begin{aligned}
\frac{\partial p_t(x)}{\partial t} &= \int -\nabla \cdot (u_t(x \mid z)\, p_t(x \mid z))\ q(z)\, dz \\
&= \int -\nabla \cdot (u_t(x \mid z)\, p_t(x \mid z)\, q(z))\ dz \\
&= -\nabla \cdot \int u_t(x \mid z)\, \frac{p_t(x \mid z)\, q(z)}{p_t(x)}\, p_t(x)\, dz \\
&= -\nabla \cdot \left(\int u_t(x \mid z)\, \frac{p_t(x \mid z)\, p(z)}{p_t(x)}\, dz \right) p_t(x) \\
&= -\nabla \cdot \left(\int u_t(x \mid z)\, \frac{p_t(x \mid z)\, q(z)}{p_t(x)}\, dz \right) p_t(x).
\end{aligned}$$

Utilizing the fact that the marginal velocity can be written as $u_t(x) = \int u_t(x \mid z) \frac{p_t(x|z)\, q(z)}{p_t(x)}\, dz$, we get the required result:

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot (u_t(x) p_t(x)).$$

(C). The random variable $x \sim \mathcal{N}(x; \mu_t(z), \sigma^2 I)$ can be written as

$$x_t = \mu_t(z) + \sigma \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1).$$

Then, from Equation 15 or Theorem 3 from Lipman et al. [2022] we get that

$$u_t(x|z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z) = 0 + \frac{d\mu_t(z)}{dt},$$

where the last equality follows from the fact that σ is constant. According to Lemma D.6, we obtain

$$u_t(x|z) = \sum_{i=1}^{n} m \times c_{i,m} \Big\{ \frac{\mathcal{B}_{i,m-1}(t)}{t_{i+m} - t_i} - \frac{\mathcal{B}_{i+1,m-1}(t)}{t_{i+m+1} - t_{i+1}} \Big\},$$

which completes the proof. □

D.6 Proof of Proposition 4.4

Proposition 4.4 (Formal). *Let the trajectories $X = [x_{t_0}, x_{t_1}, \cdots, x_{t_n}]$ be observations from the SDE dynamics as defined in Equation 16, with constant diffusion term $g(t) = \sigma$, and let the latent be $z = (x_{t_0}, x_{t_1}, \cdots, x_{t_n})$ with the distribution $q(z)$ over observed trajectories. Then, choosing the conditional probability path as a Spline-Bridge $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t^2(z))$ with $\lim_{\sigma(t_i;z)\to 0}$, where $\mu_t(z)$ is the B-spline interpolant and setting the time conditional weights $\lambda(t) = \sigma_t(z)$ we can write the regression loss $\mathcal{L}_{\text{SplineFlow}}$ as:*

$$\mathcal{L}_{\text{SplineFlow}}(\theta, \phi) = \mathbb{E}_{t,z,x}\Big[\|u_\theta(t,x) - (\epsilon\sigma_t'(z) + \mu_t'(z))\|^2 + \|\lambda(t)s_\phi(t,x) + \epsilon\|^2\Big],$$

where $\epsilon \sim \mathcal{N}(0,1)$ and the expectation is defined over time $t \sim \mathcal{U}(0,1)$, the latent variables $z \sim q(z)$ and the conditional probability samples $x \sim p_t(x \mid z)$.

Proof. We know from the preliminaries that for a given diffusion $g(t)$ the SDE dynamics can be recovered by approximating the drift $(u_t(.))$ by utilizing the probability flow velocity field $u_t^o(.)$ and the probability score $(\nabla_t p(.))$. The regression loss from SF2M objective [Tong et al., 2023] can thus be written as:

$$\mathcal{L}_{[\text{SF}]^2\text{M}}(\theta, \phi) = \mathbb{E}_{t,z,x}\Big[\|u_\theta(t,x) - u_t^o(x|z)\|^2 + \lambda(t)^2\|s_\phi(t,x) - \nabla \log p_t(x \mid z)\|^2\Big]. \tag{58}$$

And as shown in Lipman et al. [2022] for conditional gaussian paths $p_t(x|z) = \mathcal{N}(x \mid \mu_t(z), \sigma_t(z)^2)$, the velocity field inducing the distribution can be written as

$$u_t^o(x|z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z). \tag{59}$$

The samples from the conditional probability path $p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t^2(z))$ can be written as:

$$x_t = \mu_t(z) + \epsilon\sigma_t(z).$$

After substitution in Equation 59, we have

$$u_t^o(x|z) = \sigma_t'(z)\epsilon + \mu_t'(z). \tag{60}$$

And since x_t is a gaussian random variable, we get

$$\nabla_x p(x \mid z) = -\frac{(x - \mu_t(z))}{\sigma_t^2(z)},$$

which after using the fact that $x_t = \mu_t(z) + \epsilon\sigma_t(z)$ gives us

$$\nabla_x p(x \mid z) = -\frac{\epsilon}{\sigma_t(z)}. \tag{61}$$

Substituting Equation 61, Equation 60, and the given $\lambda(t) = \sigma_t(z)$ into the regression loss in Equation 58 gives us the required result,

$$\mathcal{L}_{\text{SplineFlow}}(\theta, \phi) = \mathbb{E}_{t,z,x}\Big[\|u_\theta(t,x) - (\epsilon\sigma_t'(z) + \mu_t'(z))\|^2 + \|\lambda(t)s_\phi(t,x) + \epsilon\|^2\Big].$$

□

D.7 Technical Lemmas used for Proving Main Theorems

Lemma D.1. *Let $f(t)$ be a continuous and twice differentiable function in $\mathcal{C}^2[t_i, t_{i+1}]$, with $t_i, t_{i+1} \in \mathbb{R}$, then $\exists \xi \in [t_i, t_{i+1}]$ s.t.*

$$f(t) = f(t_i) + \Big(\frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i}\Big)(t - t_i) + \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}).$$

Proof. Define the linear interpolant as

$$p_1(t) := f(t_i) + \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i}(t - t_i). \tag{62}$$

Consider some intermediary functions as

$$L(t) := f(t) - p_1(t), \qquad \phi(t) := (t - t_i)(t - t_{i+1}). \tag{63}$$

For a fixed $x \in (t_i, t_{i+1})$, define the auxiliary function as

$$g(y) := L(y) - \frac{L(t)}{\phi(t)}\,\phi(y), \qquad y \in [t_i, t_{i+1}].$$

We observe that

$$g(t_i) = g(t_{i+1}) = g(t) = 0.$$

Thus by Mean Value Theorem we know that, $\exists \xi_1 \in (t_i, x)$ and $\exists \xi_2 \in (t, t_{i+1})$ such that

$$g'(\xi_1) = g'(\xi_2) = 0.$$

Applying Mean Value Theorem theorem again, we get that $\exists \xi \in (t_i, t_{i+1})$ such that

$$g''(\xi) = 0.$$

Since $p_1(t)$ is linear, $p_1''(t) = 0$, and $\phi''(y) = 2$. We thus get

$$\begin{aligned}
g''(y) &= f''(y) - 2\frac{L(t)}{\phi(t)}, \\
&\implies f''(\xi) - \frac{2L(t)}{\phi(t)} = 0 \\
&\implies L(t) = \frac{f''(\xi)\phi(t)}{2} \\
&\implies L(t) = \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}).
\end{aligned}$$

Using this and Equations 62 and 63, we get

$$f(t) = p_1(t) + \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}), \qquad \xi \in (t_i, t_{i+1}),$$

which completes the proof. □

Corollary D.2. *Let $f(t)$ be a continuous and twice differentiable function in $\mathcal{C}^2[a, b]$, sampled at timepoints $\{t_0, t_1, \cdots, t_n\}$ such that $a < t_0 < t_1 \cdots < t_n < b$ with $p_1(t) = f(t_i) + \frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i}(t - t_i)$ be the linear interpolant in the interval $[t_i, t_{i+1}]$. Then, the approximation error is given by the following inequality:*

$$\|f(t) - p_1(t)\|_\infty \leq \frac{1}{8}\Delta_{\max}^2 |f''(t)|_\infty,$$

Similarly, the approximation error of the first derivatives is given by the following inequality:

$$\|f'(t) - p_1'(t)\|_\infty \leq \Delta_{\max} |f''(t)|_\infty,$$

where $\Delta_i = t_{i+1} - t_i$ and $\Delta_{\max} = \max_i \Delta_i$.

Proof. From Lemma D.1, we know that for $t \in [t_i, t_{i+1}]$, $\exists \xi \in [t_i, t_{i+1}]$ such that $f(t)$ could be written as

$$f(t) = f(t_i) + \Big(\frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i}\Big)(t - t_i) + \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}),$$

which could be rewritten as

$$f(t) = p_1(t) + \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}). \tag{64}$$

The approximation error then becomes

$$f(t) - p_1(t) = \frac{f''(\xi)}{2}(t - t_i)(t - t_{i+1}), \quad \|f(t) - p_1(t)\|_\infty \leq \frac{f''(\xi)}{8}(t_{i+1} - t_i)^2.$$

Taking a supremum over all the intervals gives us the approximation error bound

$$\|f(t) - p_1(t)\|_\infty \leq \Delta_{\max}^2 \frac{\|f''(t)\|_\infty}{8}.$$

Similarly, differentiating Equation 64 gives

$$f'(t) - p_1'(t) = \frac{f''(\xi)}{2}((t - t_i) + (t - t_{i+1})) \leq \frac{f''(\xi)}{2}(\|t - t_i\| + \|t - t_{i+1}\|) \leq f''(\xi)\Delta_{\max},$$

which further gives us the required bound on the first derivatives

$$\|f'(t) - p_1'(t)\|_\infty \leq \Delta_{\max}|f''(t)|_\infty.$$

□

Lemma D.3. *Let $t = (t_i)_{i=1}^{n+k}$ be a knot sequence with $t_1 = \cdots = t_k = a < \cdots < b = t_{n+1} = \cdots = t_{n+k}$ and mesh size $\Delta_{\max} := \max_i(t_{i+1} - t_i)$. Let $S_{k,t}$ denote the spline space of degree k with knots t. Assume that the interpolation operator $I : C([a, b]) \to S_{k,t}$ is well-defined and bounded, with interpolation sites at $\tau_0 < \tau_1 < \cdots < \tau_n$ for $g \in C^k([a, b])$, such that $Ig(\tau_i) = g(\tau_i)$. Then there exists a constant $C_k > 0$, depending only on k, such that*

$$\|g - Ig\|_\infty \leq (1 + \|I\|)\, C_k\, \Delta_{\max}^k\, \|g^{(k)}\|_\infty.$$

Proof. We know that the interpolant operator I is linear, and for any $s \in S_{k,t}$ since the interpolant is well defined by assumption we get that $Is = s$. We can thus write

$$g - Ig = (g - s) - I(g - s).$$

Hence, we get

$$\|g - Ig\|_\infty \leq \|g - s\|_\infty + \|I(g - s)\|_\infty \leq (1 + \|I\|)\|g - s\|_\infty.$$

Taking the infimum over $s \in S_{k,t}$ yields

$$\|g - Ig\|_\infty \leq (1 + \|I\|)\, \mathrm{dist}(g, S_{k,t}).$$

From Lemma D.4, we know that, if $g \in C^k([a, b])$ then $\mathrm{dist}(g, S_{k,t}) \leq C_k \Delta_{\max}^k \|g^{(k)}\|_\infty$. Combining the two inequalities proves the result. □

Lemma D.4. *Let $t = (t_i)_{i=1}^{n+k}$ satisfy $t_1 = \cdots = t_k = a < \cdots < b = t_{n+1} = \cdots = t_{n+k}$ and let $\Delta_{\max} := \max_i(t_{i+1} - t_i)$. Let $S_{k,t}$ be the spline space of degree k with knots t, and define*

$$\mathrm{dist}(g, S_{k,t}) := \inf_{s \in S_{k,t}} \|g - s\|_\infty.$$

If $g \in C^k([a, b])$, then there exists a constant C_k (depending only on k) such that

$$\mathrm{dist}(g, S_{k,t}) \leq C_k\, \Delta_{\max}^k\, \|g^{(k)}\|_\infty.$$

Proof. We refer readers to Chapter 12 from De Boor and De Boor [1978] or De Boor [2006]. □

Lemma D.5. *Let $t = (t_i)_{i=1}^{n+k}$ satisfy $t_1 = \cdots = t_k = a < \cdots < b = t_{n+1} = \cdots = t_{n+k}$ and let $\Delta_{\max} := \max_i(t_{i+1} - t_i)$. Let $S_{k,t}$ be the spline space of order k (degree $\leq k - 1$) with knots t. If $g \in C^k([a, b])$, then there exist B-Spline interpolation operators $Q \colon C([a, b]) \to S_{k,t}$ and constants $C_{k,j}$ (depending only on k and j) such that*

$$\|g^{(j)} - (Qg)^{(j)}\|_\infty \leq C_{k,j}\, \Delta_{\max}^{k-j}\, \|g^{(k)}\|_\infty, \qquad j = 0, 1, \ldots, k - 1,$$

provided $j \leq \lfloor k/2 \rfloor$. For $j > \lfloor k/2 \rfloor$, the same bound holds with $C_{k,j}$ replaced by a constant that depends additionally on the local mesh ratio $\max_i(\Delta t_i / \Delta t_{i-1})$.

Proof. The derivative bounds follow from Theorem (26) in Chapter XII of De Boor and De Boor [1978]; see also Theorems 5.7 and 5.8 of Lyche and Schumaker [1975].

□

Lemma D.6. *Let $\mu(t)$ be a well defined B-Spline interpolant that interpolates $f(t) \in \mathcal{C}^m[a,b]$ with observation sites at $\{t_i\}_{i=0}^n$, $\mu(t) = \sum_{i=0}^n c_{i,m}\mathcal{B}_{i,m}(t)$, where $\mathcal{B}_{i,m}$ is the B-Spline polynomial of degree m as defined in Equation 5. Then, the derivative of the interpolant function can be written as:*

$$\frac{d\mu(t)}{dt} = \sum_{i=1}^{n} m \times c_{i,m}\Big\{\frac{\mathcal{B}_{i,m-1}(t)}{t_{i+m} - t_i} - \frac{\mathcal{B}_{i+1,m-1}(t)}{t_{i+m+1} - t_{i+1}}\Big\}.$$

Proof. The result follows from the following classical result in spline theory, which says that let

$$\mathcal{B}_{i,0} = \begin{cases} 1 & \text{if} \quad t \in [t_i, t_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

With a higher degree B-Spline basis defined as:

$$\mathcal{B}_{i,m}(t) = \frac{t - t_i}{t_{i+m} - t_i}\,\mathcal{B}_{i,m-1}(t) + \frac{t_{i+m+1} - t}{t_{i+m+1} - t_{i+1}}\,\mathcal{B}_{i+1,m-1}(t).$$

Then, the derivative of $\mathcal{B}_{i,m}$ is given by

$$\frac{\mathcal{B}_{i,m}(t)}{dt} = m \times \Big\{\frac{\mathcal{B}_{i,m-1}(t)}{t_{i+m} - t_i} - \frac{\mathcal{B}_{i+1,m-1}(t)}{t_{i+m+1} - t_{i+1}}\Big\}. \tag{65}$$

This is a classical result from Butterfield [1976], also presented in De Boor and De Boor [1978], and can be derived using the method of divided differences. We can thus write

$$\frac{d\mu(t)}{dt} = \sum_i c_{i,m}\frac{d\mathcal{B}_{i,m}(t)}{dt}.$$

Utilizing Equation 65, the result follows. □

E Details of SplineFlow

E.1 Algorithm Pseudocode

Algorithm 1 SplineFlow: Flow Matching for Dynamical Systems with B-Spline Interpolants

Input: N training trajectories $\{X_i\}_{i=0}^{N-1}$, spline degree m, learning rate η, base variance σ^2, `variance`, `dynamics`
Output: velocity network u_θ and score network s_ϕ
Initialize parameters θ and ϕ
while not converged **do**
  Sample mini-batch $\mathcal{B} \subset \{X_i\}_{i=0}^{N-1}$
  for each $X \in \mathcal{B}$ **do**
    Fit a B-spline interpolant of degree m to X: $\mu(t) = \sum_j c_{j,m}\,\mathcal{B}_{j,m}(t)$
    Sample $t_j \sim \mathcal{U}(t_0, t_n),\ \epsilon \sim \mathcal{N}(0, I)$
    Sample $x_{t_j} \sim \mathcal{N}(\mu(t_j), \sigma^2(t_j)I)$
    Compute $u(t_j) \leftarrow \mu'(t_j)$ ▷ via Theorem 4.3
    if `variance` = `quadratic` **then**
      Compute $\sigma'(t_j)$ ▷ via Equation 9
    else
      Set $\sigma'(t_j) \leftarrow 0$
    end if
  end for
  if `dynamics` = `ODE` **then**
    $\mathcal{L}(\theta) \leftarrow \frac{1}{|\mathcal{B}|}\sum_{X\in\mathcal{B}} \|u_\theta(t_j, x_{t_j}) - u(t_j)\|_2^2$
    $\theta \leftarrow \theta - \eta\,\nabla_\theta\mathcal{L}(\theta)$
  else ▷ `dynamics` = `SDE`
    $\mathcal{L}(\theta,\phi) \leftarrow \frac{1}{|\mathcal{B}|}\sum_{X\in\mathcal{B}} \left(\|u_\theta(t_j, x_{t_j}) - (u(t_j) + \epsilon\,\sigma'(t_j))\|_2^2 + \|\lambda(t_j)\,s_\phi(t_j, x_{t_j}) + \epsilon\|_2^2\right)$
    $\theta \leftarrow \theta - \eta\,\nabla_\theta\mathcal{L}(\theta,\phi),\ \ \phi \leftarrow \phi - \eta\,\nabla_\phi\mathcal{L}(\theta,\phi)$
  end if
end while

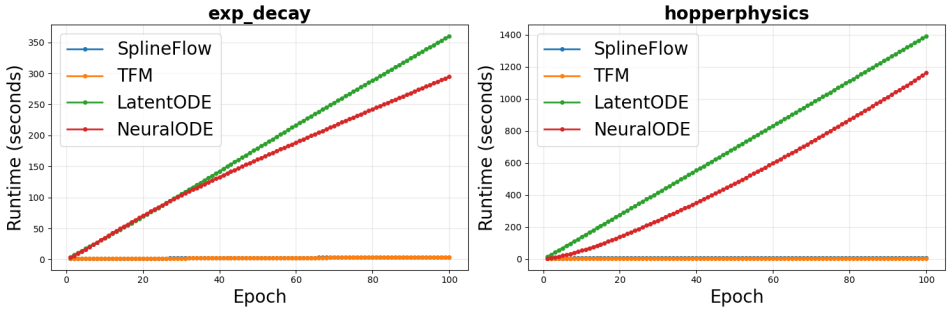


Figure 3: Runtime of adjoint ODE-based and simulation-free flow matching methods.

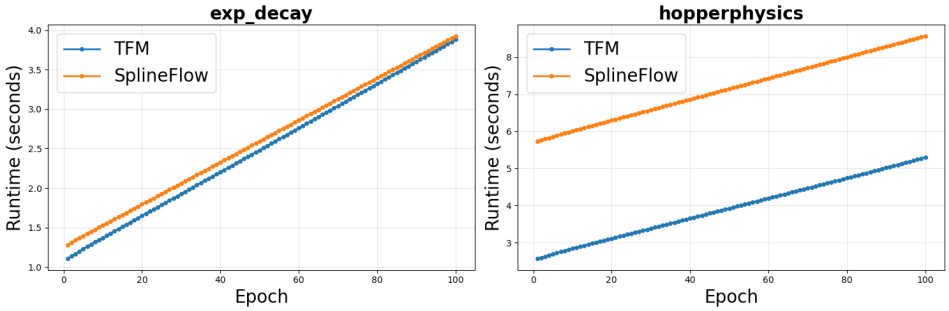


Figure 4: Runtime comparison between flow matching methods.

E.2 Runtime Complexity

While using SplineFlow, the total runtime depends on (a) creating B-splines, (b) sampling values and derivatives from B-splines, and (c) forward-backward passes for each epoch. Consider a total of N trajectories of dimensionality D, each with n observations in total, interpolated with B-splines of order m, and let Q be the number of sampling queries made to calculate conditional velocities and values. Let total epochs be T and let F be the average time taken for each forward-backward pass while training on the regression objective for a batch size B.

For every trajectory, for each observation i, we need to recursively create m spline bases, and for each interval $t \in [t_i, t_{i+1})$, we have m bases from neighboring observations to add. And since we do this separately for each dimension D, the cost comes out as $\mathcal{O}(NDnm^2)$. For every Q queries we evaluate from the B-spline construction (for conditional entities), for every trajectory, we again sum over m bases for each query, thus bringing the cost to $\mathcal{O}(2 \times NQDm)$, together for x_t and v_t. The complexity for calculating precomputed conditional paths and velocities is thus $\mathcal{O}(NDm(nm + Q))$. These precomputed entities are then passed along the training algorithm, with a total time of $\mathcal{O}(TF)$ per batch B and N/B dataloader passes per epoch. Thus, the total complexity of SplineFlow comes out to be $\mathcal{O}(NDm(nm + Q) + TF \times (N/B))$.

We also run experiments to compare SplineFlow ($m = 3$) with baselines on the smallest Exponential Decay dataset ($d = 1$) and the largest HopperPhysics dataset ($d = 14$). Figure 3 demonstrates significant advantages of simulation-free methods in terms of total runtime. Owing to its encoder-decoder architecture, LatentODE has a higher runtime than NeuralODE. Whereas from Figure 4, we can see that for Exponential Decay $d = 1$, both TFM and SplineFlow have almost identical runtime.

Table 7: Time taken for processing of conditional paths and velocities before training epochs begin

Dataset	Method	Time (s)
Exponential Decay	SplineFlow	1.116
	TFM	0.934
HopperPhysics	SplineFlow	5.558
	TFM	2.407

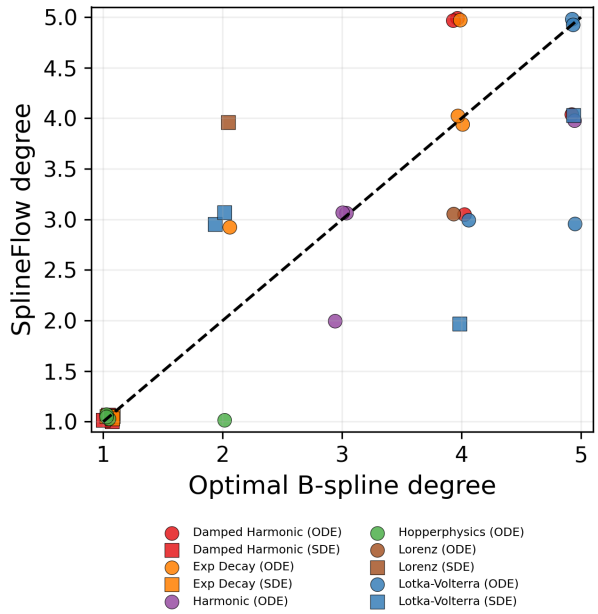


Figure 5: Visualizations of the B-spline degree selected via cross-validation vs. the optimal degree, which corresponds to a Spearman correlation coefficient of $\rho = 0.85$.

Table 8: Comparisons between methods, with SplineFlow degree m_{CV} selected using cross validation.

p	Model	Exp.	Harm.	Damp.	LV	Hopper	Lorenz
0	NeuralODE	9.99e−1	6.54	3.47	1.98e4	2.95	1.49
	LatentODE	9.80e−5	8.20e−4	1.21e−4	**1.56e−3**	1.89	1.16
	MMFM	1.40e−3	1.40e−2	2.00e−3	4.87e−1	1.82e1	8.02e−1
	TFM	6.56e−4	1.81e−1	1.52e−2	1.84e−1	1.55	2.44
	SplineFlow (CV)	**8.90e−5**	**5.28e−4**	**7.20e−5**	5.53e−2	**1.41**	**7.77e−1**
0.25	LatentODE	9.80e−5	1.22e−3	3.42e−4	**9.42e−4**	1.60	–
	MMFM	2.00e−3	1.40e−2	1.80e−3	3.72e−1	3.13e1	–
	TFM	6.86e−4	1.74e−1	1.55e−2	2.59e−1	1.51	–
	SplineFlow (CV)	**8.20e−5**	**6.53e−4**	**5.10e−5**	1.00e−1	**1.43**	–
0.5	LatentODE	1.56e−4	7.64e−4	7.25e−4	**1.55e−3**	**1.65**	–
	MMFM	1.70e−3	1.20e−2	3.00e−3	4.41	4.78e1	–
	TFM	9.44e−4	1.77e−1	1.61e−2	4.26	1.72	–
	SplineFlow (CV)	**1.06e−4**	**5.46e−4**	**4.70e−5**	7.05e−1	1.83	–
0.75	LatentODE	9.60e−5	**7.94e−4**	9.71e−4	**2.90e−3**	3.56	–
	MMFM	2.40e−3	1.00e−2	4.00e−3	8.62	6.11e1	–
	TFM	1.35e−3	2.05e−1	3.12e−2	1.98	**2.81**	–
	SplineFlow (CV)	**8.60e−5**	1.31e−3	**4.80e−5**	1.78	3.34	–
	Avg. Training Runtime (s)						
	LatentODE	2.80e4	4.47e4	4.38e4	4.44e4	1.21e5	5.30e4
	SplineFlow (CV)	**1.99e2**	**2.88e2**	**2.80e2**	**2.83e2**	**2.32e2**	**2.95e2**
	Speedup (×)	**141**	**155**	**156**	**157**	**520**	**180**

For HopperPhysics, SplineFlow incurs about 3 seconds of B-spline calculation overhead compared to the linear-interpolant calculation, as shown in Table 7 below. However, it's important to note that this overhead is a one-time cost incurred before training and thus does not affect the overall training time.

Table 9: $\mathcal{W}_2$ comparison of different methods, with SplineFlow degree m_{CV} selected using the CV rule above for SDE systems.

Setting	p	SplineFlow (CV)	SplineFlow	SF2M
Exp.	0	3.31e−1	3.31e−1	**3.05e−1**
	0.5	**1.95e−1**	**1.95e−1**	2.04e−1
Damp.	0	**4.10e−1**	**4.10e−1**	4.30e−1
	0.5	3.21e−1	3.21e−1	**3.19e−1**
LV	0	3.21e−1	**2.94e−1**	4.43e−1
	0.5	3.74e−1	**3.10e−1**	3.75e−1
Lorenz	0	4.30e−1	**1.80e−1**	6.32e−1

F Hyperparameter Selection

Degree of B-Spline Interpolant m. The degree m could be selected by performing a simple cross-validation (CV) on training data. Let $\{X^i\}_{i\in[1\cdots N]}$ be observed trajectories as defined in Section 3. For each trajectory X^i create a fit X^i_{fit} and test X^i_{test} subsets such that $X^i_{\text{fit}} \cup X^i_{\text{test}} = X^i$. Fit a B-spline interpolant as defined in Equation 6 ($\mathcal{I}_m$) of degree m such that the interpolant satisfies observations at fit subset $\mathcal{I}_m(X^i_{\text{fit}}) = X^i_{\text{fit}}$, and calculate the error on X^i_{test} as $\sum_{i\in\text{training}} \|\mathcal{I}_m(X^i_{\text{test}}) - X^i_{\text{test}}\|^2$. Then, we can select the degree that minimizes the error.

Importantly, the above degree selection rule uses only training data, requires no training of a flow model, and evaluates B-spline interpolation errors for the candidate degrees. Thus, its computational overhead is negligible relative to the subsequent model training cost. Below we present the CV selected degree (m_{CV}) performance as opposed to the optimal degree (m_{oracle}) performance presented in Tables 1 and 2. Comparing Tables 1 and 8 for the ODE case, and Tables 9 and 2 for SDE case we can see that SplineFlow with CV selected degree still retains its performance gains compared to existing baselines, with the CV selected degree (m_{CV}) often lying close to the optimal degree (m_{oracle}) as seen in Table 10 below.

Table 10: m_{CV} vs. m_{oracle} for ODE and SDE systems. Each entry reports $m_{\text{CV}}/m_{\text{oracle}}$.

(a) ODE

Setting	$p=0$	$p=0.25$	$p=0.5$	$p=0.75$
Exp.	4/4	4/4	4/4	4/5
Harm.	3/2	3/3	3/3	5/4
Damp.	4/3	4/5	4/5	5/4
LV	4/3	5/5	5/3	5/5
Hopper	1/1	1/1	1/1	1/1
Lorenz	4/3	–	–	–

(b) SDE

Setting	$p=0$	$p=0.5$
Exp.	1/1	1/1
Damp.	1/1	1/1
LV	4/2	2/3
Lorenz	2/4	–

Constant Diffusion Schedule σ. Consider trajectory X^i, $i \in \{1, 2, \cdots, N\}$, sampled from the SDE evolution with constant diffusion schedule mentioned in Equation 16: $dx_t = u_t(x_t)dt + \sigma dw_t$, with $x \in \mathbb{R}^d$. Let trajectory i contain n_i observations $\{x_i(t^i_0), \cdots, x_i(t^i_{n_i})\}$, then the constant diffusion term σ can then be estimated using a discrete estimator below:

$$\hat{\sigma}^2 = \frac{1}{d\sum_{i=1}^N T_i} \sum_{i=1}^{N} \sum_{k=0}^{n_i-1} \left\| x_i(t^i_{k+1}) - x_i(t^i_k) \right\|_2^2, \tag{66}$$

where $T_i = \sum_{k=1}^{n_i}(t_{k+1} - t_k)$ is the sum of time differences. This estimator follows from classical results on quadratic variation of Ito diffusions, which imply that the sum of squared increments converges to the integrated diffusion coefficient [Sørensen, 2004, Karatzas and Shreve, 2014]. Note that the estimator is meant to provide a guide for selecting the hyperparameter σ; however, the input (σ) to Algorithm 1 should still be at the discretion of the user. In addition, we conduct an ablation over the diffusion scale σ for SDE under $\mathcal{W}_2$ with SplineFlow using the degree selected by training-only CV (m_{CV}). As shown in the Table 11 below, the best performance is generally obtained at moderate

Table 11: Ablation over σ for SDE systems

p	Dataset	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}
0	Exp.	0.496	0.477	0.447	**0.253**	6.124	116.150
0	Damp.	0.473	0.513	**0.377**	0.402	5.478	85.405
0	LV	2.954	7.641	1.163	**0.346**	0.476	0.898
0	Lorenz	0.201	0.210	**0.185**	0.219	0.587	3.379
0.5	Exp.	0.426	0.382	0.392	**0.142**	7.031	114.271
0.5	Damp.	0.499	0.490	0.502	**0.278**	6.406	89.495
0.5	LV	16.251	7.649	4.577	**0.342**	0.428	0.863

diffusion scales within $[0.01, 0.1]$. Compared to Table 2, the results obtained using our estimated values are close to the best values in the ablation, thus supporting the efficacy of the estimation rule.

G Training Details

G.1 Hyperparameters

We use similar neural network architectures and training configurations across SplineFlow and all baselines. Specifically, we set:

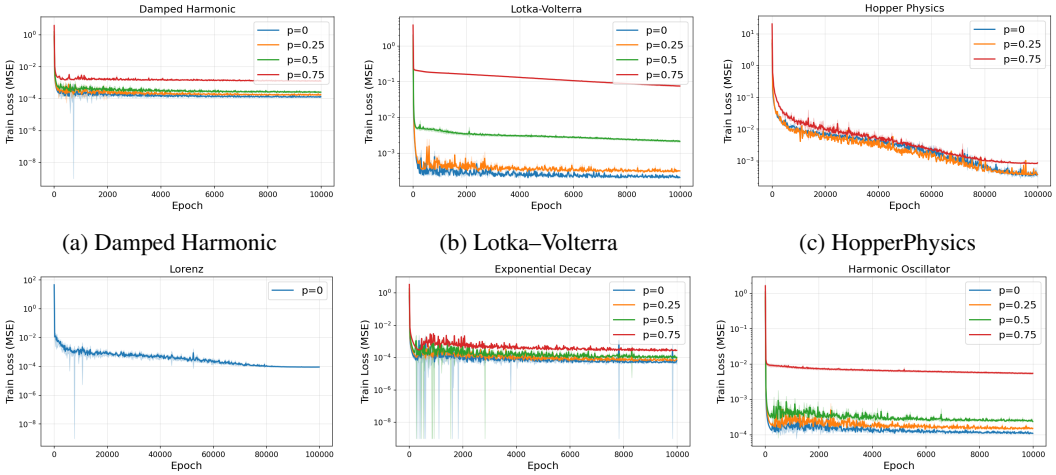
- Learning Rate (LR): 0.0005
- LR scheduler: Cosine
- Epochs: $10,000$
- NN: MLP
- Depth: 4
- Width for ODE experiments: 256 (2048 for Lorenz and HopperPhysics owing to dimensionality and complexity)
- Width for SDE experiments: 2×256 (2×2048 for Lorenz owing to complexity) for both velocity field (u_θ) and score network (s_ϕ)
- Width for cellular trajectory experiments: 64

G.2 Loss Curves

Figures 6 show training loss curves for training SplineFlow on ODE dynamics. SplineFlow consistently converges to a near-zero training loss validating that more sophisticated supervision signals do not incur additional training overhead.

H Full Experimental Results

This section presents the full experimental results and summarized heatmap visualizations for SplineFlow with the best-selected spline degree parameter m, benchmarked against baseline methods across all evaluated datasets and configurations.



(a) Damped Harmonic (b) Lotka–Volterra (c) HopperPhysics

(d) Lorenz (e) Exponential Decay (f) Harmonic Oscillator

Figure 6: Training loss (log scale) vs. epoch for SplineFlow on ODE systems ($p = 0$, regular sampling). Shaded bands show the standard deviation across five runs.

Table 12: Comparisons of different methods for modeling ODE dynamics in MSE metric across 5 deterministic dynamical systems and varying sampling irregularity $p \in \{0, 0.25, 0.5, 0.75\}$.

p	Model	Exp-Decay	Harmonic	Damped Harmonic	Lotka–Volterra	HopperPhysics	Lorenz
0	NeuralODE	0.999 ± 0.126	6.542 ± 5.965	3.474 ± 0.366	$1.98e4 \pm 1.24e4$	2.947 ± 0.183	1.491 ± 0.359
	LatentODE	$9.8e{-5} \pm 4.7e{-5}$	$8.2e{-4} \pm 4.2e{-4}$	$1.2e{-4} \pm 6.3e{-4}$	$\mathbf{1.6e{-3} \pm 1.2e{-3}}$	1.894 ± 0.523	1.157 ± 0.204
	MMFM	$1.4e{-3} \pm 1.0e{-4}$	0.014 ± 0.009	0.002 ± 0.002	0.487 ± 0.384	18.179 ± 4.049	0.802 ± 0.008
	TFM	$6.6e{-4} \pm 8.3e{-5}$	$0.181 \pm 3.8e{-4}$	$0.015 \pm 2.7e{-4}$	0.184 ± 0.104	1.554 ± 0.297	2.442 ± 0.076
	SplineFlow	$\mathbf{8.9e{-5} \pm 3.3e{-5}}$	$\mathbf{3.7e{-4} \pm 3.1e{-4}}$	$\mathbf{9.5e{-5} \pm 1.0e{-4}}$	0.086 ± 0.043	$\mathbf{1.410 \pm 0.172}$	$\mathbf{0.639 \pm 0.004}$
0.25	LatentODE	$9.8e{-5} \pm 3.7e{-5}$	0.001 ± 0.001	$3.4e{-4} \pm 5.8e{-4}$	$\mathbf{9e-4 \pm 3.3e{-4}}$	1.605 ± 0.019	–
	MMFM	$2.0e{-3} \pm 8.0e{-4}$	0.014 ± 0.006	–	0.372 ± 0.213	31.294 ± 8.263	–
	TFM	$6.9e{-4} \pm 9.6e{-5}$	0.174 ± 0.008	$0.015 \pm 2.6e{-4}$	0.259 ± 0.172	1.512 ± 0.030	–
	SplineFlow	$\mathbf{8.2e{-5} \pm 2.9e{-5}}$	$\mathbf{6.5e{-4} \pm 2.1e{-4}}$	$\mathbf{3.0e{-5} \pm 4.8e{-6}}$	0.100 ± 0.057	$\mathbf{1.433 \pm 0.030}$	–
0.5	LatentODE	$1.6e{-4} \pm 4.5e{-5}$	$7.6e{-4} \pm 6.7e{-5}$	$7.3e{-4} \pm 3.6e{-4}$	$\mathbf{1.5e-3 \pm 0.001}$	$\mathbf{1.649 \pm 0.073}$	–
	MMFM	$1.7e{-3} \pm 3.0e{-4}$	0.012 ± 0.002	0.003 ± 0.001	4.41 ± 0.223	47.836 ± 4.026	–
	TFM	$9.4e{-4} \pm 1.7e{-4}$	0.177 ± 0.018	0.016 ± 0.001	4.259 ± 5.690	1.718 ± 0.092	–
	SplineFlow	$\mathbf{1.1e{-4} \pm 6.8e{-5}}$	$\mathbf{5.5e{-4} \pm 2.2e{-4}}$	$\mathbf{4.7e{-5} \pm 2.1e{-5}}$	0.705 ± 1.210	1.825 ± 0.602	–
0.75	LatentODE	$9.6e{-5} \pm 5.0e{-6}$	$\mathbf{7.9e{-4} \pm 2.5e{-4}}$	$9.7e{-4} \pm 3.7e{-4}$	$\mathbf{3e-3 \pm 0.001}$	3.563 ± 0.148	–
	MMFM	$2.4e{-3} \pm 4.0e{-4}$	0.010 ± 0.002	0.004 ± 0.0005	8.62 ± 0.038	61.110 ± 19.287	–
	TFM	$1.4e{-3} \pm 2.0e{-4}$	0.205	0.031 ± 0.002	1.976 ± 1.810	$\mathbf{2.810 \pm 0.165}$	–
	SplineFlow	$\mathbf{8.6e{-5} \pm 6.4e{-6}}$	0.001 ± 0.001	$\mathbf{4.8e{-5} \pm 2.3e{-5}}$	1.783 ± 1.430	3.336 ± 0.430	–

Table 13: Performance across SDE dynamics datasets for different metrics under regular sampling

Dataset	Model	MSE (P-ODE)	MSE (SDE)	Wasserstein	MMD	Energy
Exp.	SF2M	$3.91e{-01} \pm 2.26e{-03}$	$4.77e{-01} \pm 9.90e{-03}$	$\mathbf{3.05e{-01} \pm 4.85e{-03}}$	$\mathbf{2.40e{-02} \pm 1.00e{-03}}$	$\mathbf{6.10e{-02} \pm 2.62e{-03}}$
	SplineFlow	$\mathbf{3.90e{-01} \pm 4.23e{-03}}$	$\mathbf{4.55e{-01} \pm 8.27e{-03}}$	$3.30e{-01} \pm 2.83e{-03}$	$2.70e{-02} \pm 2.48e{-03}$	$6.94e{-02} \pm 1.08e{-03}$
Damped.	SF2M	$\mathbf{8.44e{-01} \pm 1.54e{-02}}$	$\mathbf{9.66e{-01} \pm 3.29e{-02}}$	$4.29e{-01} \pm 7.40e{-03}$	$4.84e{-02} \pm 2.96e{-03}$	$1.25e{-01} \pm 3.74e{-03}$
	SplineFlow	$8.94e{-01} \pm 5.12e{-03}$	$9.69e{-01} \pm 9.41e{-03}$	$\mathbf{4.10e{-01} \pm 4.41e{-03}}$	$\mathbf{4.19e{-02} \pm 1.54e{-03}}$	$\mathbf{1.12e{-01} \pm 7.12e{-03}}$
LV	SF2M	$4.58e{-01} \pm 9.74e{-02}$	$4.60e{-01} \pm 9.67e{-02}$	$4.43e{-01} \pm 4.99e{-02}$	$1.25e{-01} \pm 2.35e{-02}$	$4.65e{-01} \pm 9.44e{-02}$
	SplineFlow	$\mathbf{2.68e{-01} \pm 1.44e{-02}}$	$\mathbf{2.73e{-01} \pm 1.37e{-02}}$	$\mathbf{2.93e{-01} \pm 3.16e{-03}}$	$\mathbf{4.21e{-02} \pm 3.45e{-03}}$	$\mathbf{1.56e{-01} \pm 1.33e{-02}}$
Lorenz	SF2M	$2.17e{+00} \pm 7.94e{-03}$	$2.15e{+00} \pm 8.35e{-03}$	$6.32e{-01} \pm 7.61e{-03}$	$1.29e{-01} \pm 2.56e{-03}$	$4.04e{-01} \pm 1.08e{-02}$
	SplineFlow	$\mathbf{1.23e{+00} \pm 4.01e{-02}}$	$\mathbf{1.22e{+00} \pm 5.62e{-02}}$	$\mathbf{1.79e{-01} \pm 1.47e{-02}}$	$\mathbf{5.00e{-03} \pm 1.79e{-03}}$	$\mathbf{3.66e{-02} \pm 7.65e{-03}}$

Table 14: Performance across SDE dynamics datasets for different metrics under irregular sampling.

Dataset	p	Model	MSE (P-ODE)	MSE (SDE)	Wasserstein	MMD	Energy
Exp.	0.25	SF2M	$\mathbf{3.64e{-}01 \pm 1.27e{-}03}$	$\mathbf{4.65e{-}01 \pm 6.30e{-}03}$	$\mathbf{2.45e{-}01 \pm 7.37e{-}03}$	$\mathbf{1.46e{-}02 \pm 1.15e{-}03}$	$\mathbf{4.01e{-}02 \pm 2.76e{-}03}$
		SplineFlow	$3.66e{-}01 \pm 6.08e{-}03$	$4.69e{-}01 \pm 5.90e{-}03$	$2.50e{-}01 \pm 5.53e{-}03$	$1.51e{-}02 \pm 1.03e{-}03$	$4.12e{-}02 \pm 3.14e{-}03$
	0.5	SF2M	$3.77e{-}01 \pm 5.88e{-}03$	$\mathbf{5.23e{-}01 \pm 1.07e{-}02}$	$2.04e{-}01 \pm 1.16e{-}02$	$9.09e{-}03 \pm 1.93e{-}03$	$2.83e{-}02 \pm 4.68e{-}03$
		SplineFlow	$\mathbf{3.76e{-}01 \pm 3.30e{-}03}$	$5.30e{-}01 \pm 6.59e{-}03$	$\mathbf{1.94e{-}01 \pm 1.42e{-}03}$	$\mathbf{7.42e{-}03 \pm 3.31e{-}04}$	$\mathbf{2.39e{-}02 \pm 1.36e{-}03}$
	0.75	SF2M	$\mathbf{4.05e{-}01 \pm 6.73e{-}03}$	$\mathbf{6.46e{-}01 \pm 1.07e{-}02}$	$\mathbf{1.36e{-}01 \pm 1.20e{-}02}$	$\mathbf{1.47e{-}03 \pm 9.50e{-}04}$	$\mathbf{1.27e{-}02 \pm 3.13e{-}03}$
		SplineFlow	$4.18e{-}01 \pm 5.87e{-}03$	$6.76e{-}01 \pm 8.23e{-}03$	$1.42e{-}01 \pm 1.72e{-}02$	$2.05e{-}03 \pm 1.55e{-}03$	$1.51e{-}02 \pm 5.15e{-}03$
Damped.	0.25	SF2M	$8.35e{-}01 \pm 3.23e{-}02$	$9.50e{-}01 \pm 2.01e{-}02$	$3.71e{-}01 \pm 3.46e{-}02$	$\mathbf{3.29e{-}02 \pm 7.30e{-}03}$	$9.31e{-}02 \pm 1.77e{-}02$
		SplineFlow	$\mathbf{7.98e{-}01 \pm 1.57e{-}02}$	$\mathbf{9.13e{-}01 \pm 2.28e{-}02}$	$\mathbf{3.64e{-}01 \pm 2.11e{-}02}$	$3.42e{-}02 \pm 4.34e{-}03$	$\mathbf{9.16e{-}02 \pm 1.07e{-}02}$
	0.5	SF2M	$8.28e{-}01 \pm 3.29e{-}02$	$\mathbf{9.72e{-}01 \pm 2.54e{-}02}$	$\mathbf{3.19e{-}01 \pm 7.43e{-}03}$	$2.20e{-}02 \pm 2.68e{-}03$	$\mathbf{6.56e{-}02 \pm 5.04e{-}03}$
		SplineFlow	$\mathbf{8.05e{-}01 \pm 8.73e{-}03}$	$9.81e{-}01 \pm 3.99e{-}02$	$3.21e{-}01 \pm 2.84e{-}02$	$\mathbf{2.17e{-}02 \pm 5.59e{-}03}$	$6.87e{-}02 \pm 1.21e{-}02$
	0.75	SF2M	$\mathbf{9.12e{-}01 \pm 2.70e{-}02}$	$\mathbf{1.15e{+}00 \pm 3.93e{-}02}$	$\mathbf{2.75e{-}01 \pm 2.17e{-}02}$	$\mathbf{1.09e{-}02 \pm 2.14e{-}03}$	$\mathbf{4.65e{-}02 \pm 6.44e{-}03}$
		SplineFlow	$9.37e{-}01 \pm 2.89e{-}02$	$1.17e{+}00 \pm 1.17e{-}02$	$2.85e{-}01 \pm 1.10e{-}02$	$1.24e{-}02 \pm 7.65e{-}04$	$5.27e{-}02 \pm 4.89e{-}03$
LV	0.25	SF2M	$4.07e{-}01 \pm 6.92e{-}02$	$4.04e{-}01 \pm 7.33e{-}02$	$4.00e{-}01 \pm 2.56e{-}02$	$9.24e{-}02 \pm 8.00e{-}03$	$3.45e{-}01 \pm 3.23e{-}02$
		SplineFlow	$\mathbf{3.18e{-}01 \pm 3.58e{-}02}$	$\mathbf{3.21e{-}01 \pm 3.21e{-}02}$	$\mathbf{3.18e{-}01 \pm 1.21e{-}02}$	$\mathbf{4.44e{-}02 \pm 5.03e{-}03}$	$\mathbf{1.74e{-}01 \pm 2.60e{-}02}$
	0.5	SF2M	$3.50e{-}01 \pm 2.88e{-}02$	$3.54e{-}01 \pm 3.12e{-}02$	$3.74e{-}01 \pm 3.69e{-}02$	$8.03e{-}02 \pm 2.88e{-}02$	$2.93e{-}01 \pm 9.23e{-}02$
		SplineFlow	$\mathbf{2.87e{-}01 \pm 2.32e{-}02}$	$\mathbf{2.85e{-}01 \pm 2.54e{-}02}$	$\mathbf{3.10e{-}01 \pm 1.16e{-}02}$	$\mathbf{4.41e{-}02 \pm 3.80e{-}03}$	$\mathbf{1.63e{-}01 \pm 2.32e{-}02}$
	0.75	SF2M	$5.24e{-}01 \pm 1.24e{-}01$	$5.26e{-}01 \pm 1.21e{-}01$	$4.46e{-}01 \pm 9.53e{-}02$	$1.06e{-}01 \pm 3.98e{-}02$	$4.15e{-}01 \pm 1.58e{-}01$
		SplineFlow	$\mathbf{3.77e{-}01 \pm 5.73e{-}02}$	$\mathbf{3.78e{-}01 \pm 5.48e{-}02}$	$\mathbf{3.60e{-}01 \pm 4.04e{-}02}$	$\mathbf{5.57e{-}02 \pm 9.43e{-}03}$	$\mathbf{2.22e{-}01 \pm 4.77e{-}02}$

Table 15: Performance of models on holdout interpolation and extrapolation settings for the Brain Regeneration dataset.

Embedding	Task	Model	Wasserstein	MMD	Energy
PHATE	Interpolation	MOTFM	$7.94e{-}1 \pm 2.64e{-}2$	$5.03e{-}1 \pm 1.60e{-}2$	$1.44e{+}0 \pm 6.21e{-}2$
		SF2M	$7.53e{-}1 \pm 2.58e{-}2$	$4.57e{-}1 \pm 8.30e{-}3$	$1.31e{+}0 \pm 3.55e{-}2$
		SplineFlow	$\mathbf{6.07e{-}1 \pm 2.71e{-}2}$	$\mathbf{2.85e{-}1 \pm 6.60e{-}3}$	$\mathbf{8.15e{-}1 \pm 2.14e{-}2}$
	Extrapolation	MOTFM	$1.57e{+}0 \pm 4.50e{-}3$	$1.01e{+}0 \pm 1.15e{-}2$	$3.89e{+}0 \pm 5.07e{-}2$
		SF2M	$\mathbf{1.53e{+}0 \pm 4.72e{-}2}$	$9.72e{-}1 \pm 9.20e{-}3$	$3.77e{+}0 \pm 7.24e{-}2$
		SplineFlow	$1.59e{+}0 \pm 1.92e{-}2$	$\mathbf{9.55e{-}1 \pm 8.70e{-}3}$	$\mathbf{3.74e{+}0 \pm 4.24e{-}2}$
PCA	Interpolation	MOTFM	$9.01e{-}1 \pm 2.16e{-}2$	$7.12e{-}2 \pm 4.40e{-}3$	$1.16e{+}0 \pm 5.24e{-}2$
		SF2M	$8.88e{-}1 \pm 7.40e{-}3$	$6.68e{-}2 \pm 7.00e{-}4$	$1.14e{+}0 \pm 1.24e{-}2$
		SplineFlow	$\mathbf{8.38e{-}1 \pm 1.37e{-}2}$	$\mathbf{5.82e{-}2 \pm 2.50e{-}3}$	$\mathbf{1.02e{+}0 \pm 3.93e{-}2}$
	Extrapolation	MOTFM	$\mathbf{7.99e{-}1 \pm 3.10e{-}3}$	$1.35e{-}1 \pm 6.90e{-}3$	$1.65e{+}0 \pm 4.28e{-}2$
		SF2M	$8.65e{-}1 \pm 1.54e{-}2$	$1.77e{-}1 \pm 2.50e{-}3$	$1.92e{+}0 \pm 1.27e{-}2$
		SplineFlow	$8.04e{-}1 \pm 3.72e{-}2$	$\mathbf{1.29e{-}1 \pm 6.80e{-}3}$	$\mathbf{1.65e{+}0 \pm 5.66e{-}2}$

Table 16: Performance of models across holdout interpolation and extrapolation settings for the Embryoid Evolution dataset.

Embedding	Task	Model	Wasserstein	MMD	Energy
PHATE	Interpolation	MOTFM	$3.78e{-}1 \pm 1.11e{-}2$	$6.01e{-}2 \pm 2.52e{-}3$	$1.75e{-}1 \pm 4.98e{-}3$
		SF2M	$2.72e{-}1 \pm 2.36e{-}2$	$2.46e{-}2 \pm 4.03e{-}3$	$8.63e{-}2 \pm 1.33e{-}2$
		SplineFlow	$\mathbf{2.07e{-}1 \pm 1.69e{-}2}$	$\mathbf{1.26e{-}2 \pm 5.87e{-}4}$	$\mathbf{3.96e{-}2 \pm 1.40e{-}3}$
	Extrapolation	MOTFM	$4.48e{-}1 \pm 5.05e{-}2$	$8.25e{-}2 \pm 9.05e{-}3$	$2.39e{-}1 \pm 7.18e{-}3$
		SF2M	$\mathbf{4.39e{-}1 \pm 4.40e{-}2}$	$\mathbf{5.79e{-}2 \pm 1.08e{-}2}$	$\mathbf{1.94e{-}1 \pm 1.57e{-}2}$
		SplineFlow	$4.48e{-}1 \pm 4.41e{-}2$	$6.90e{-}2 \pm 1.70e{-}3$	$2.07e{-}1 \pm 2.96e{-}2$
PCA	Interpolation	MOTFM	$2.86e{-}1 \pm 1.37e{-}2$	$1.14e{-}2 \pm 9.33e{-}4$	$1.65e{-}1 \pm 1.23e{-}2$
		SF2M	$\mathbf{2.74e{-}1 \pm 1.05e{-}2}$	$\mathbf{9.12e{-}3 \pm 8.42e{-}4}$	$\mathbf{1.55e{-}1 \pm 1.25e{-}2}$
		SplineFlow	$2.96e{-}1 \pm 1.46e{-}2$	$1.27e{-}2 \pm 3.11e{-}4$	$1.99e{-}1 \pm 9.34e{-}3$
	Extrapolation	MOTFM	$7.89e{-}1 \pm 1.29e{-}1$	$3.75e{-}2 \pm 2.11e{-}3$	$7.21e{-}1 \pm 5.09e{-}2$
		SF2M	$5.92e{-}1 \pm 2.02e{-}2$	$3.77e{-}2 \pm 3.98e{-}4$	$7.47e{-}1 \pm 1.18e{-}2$
		SplineFlow	$\mathbf{5.39e{-}1 \pm 3.35e{-}2}$	$\mathbf{3.58e{-}2 \pm 4.42e{-}3}$	$\mathbf{6.89e{-}1 \pm 1.01e{-}1}$

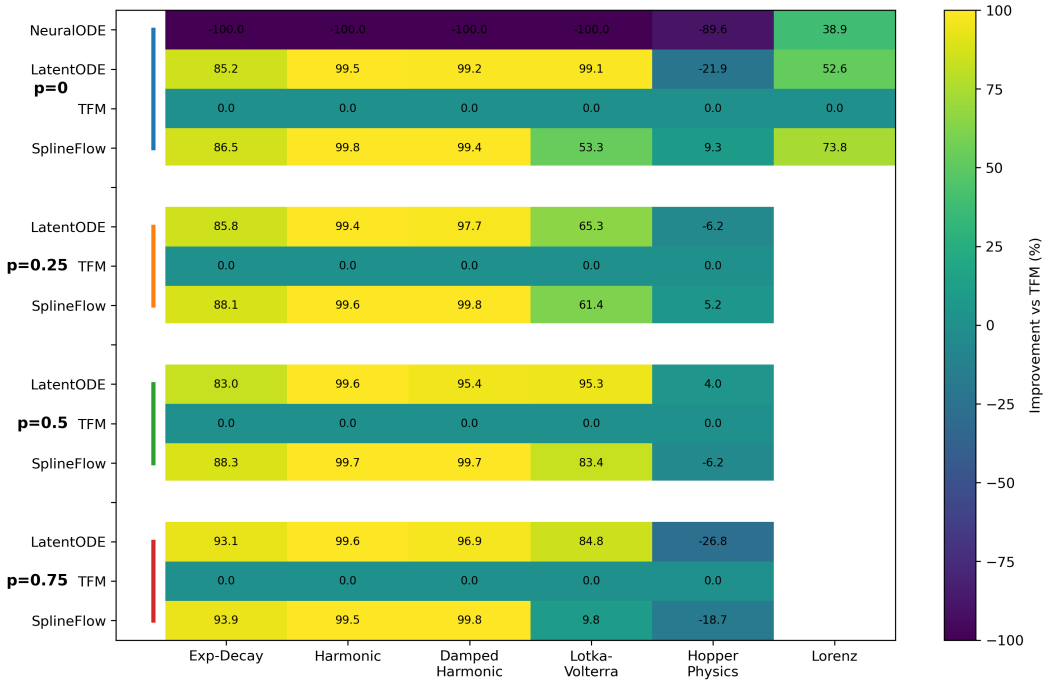


Figure 7: Heatmap visualization of relative improvements of LatentODE and SplineFlow in MSE with reference to TFM, summarized from the ODE Table 12.

I Visualizations of Predicted Dynamics

In this section, we visualize the predicted trajectories alongside the ground truth in different spaces to further examine the learning performance of different methods (apart from quantitative metric performance). For irregularly sampled settings, we choose $p = 0.75$ to reflect the most challenging scenarios. The trajectories predicted by our SplineFlow method can recover the ground-truth dynamics well, particularly outperforming other methods for the chaotic Lorenz system.

I.1 ODE Dynamics

I.1.1 Harmonic Oscillator ($p = 0.75$)

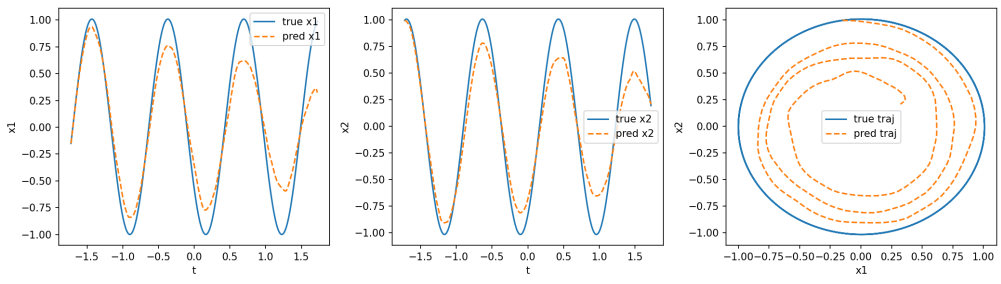


Figure 8: TFM evaluated path for Harmonic Oscillator.

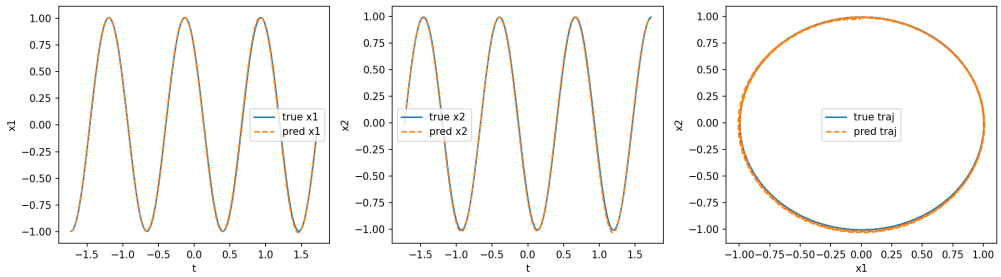


Figure 9: SplineFlow evaluated path for Harmonic Oscillator.

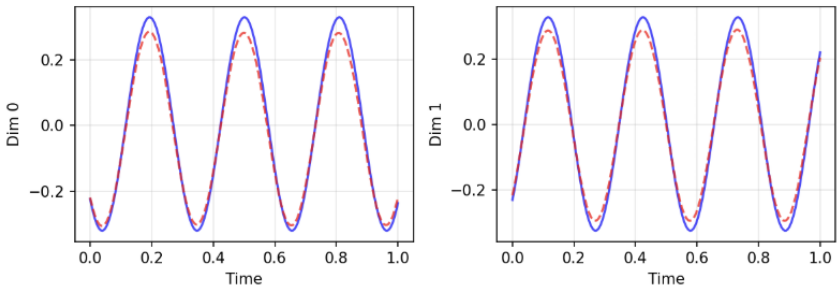


Figure 10: LatentODE evaluated path for Harmonic Oscillator.

I.1.2 Damped Harmonic Oscillator ($p = 0.75$)

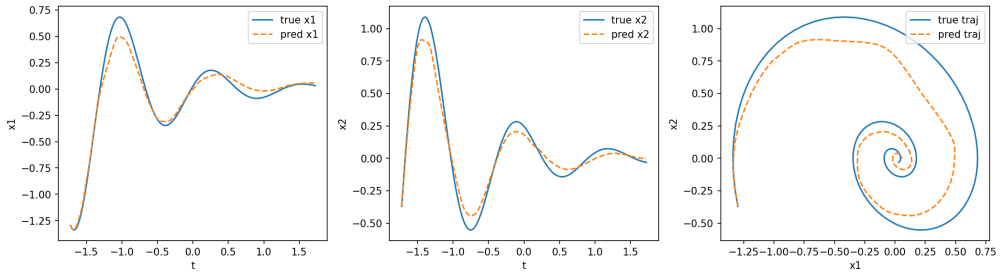


Figure 11: TFM evaluated path for Damped Harmonic Oscillator.

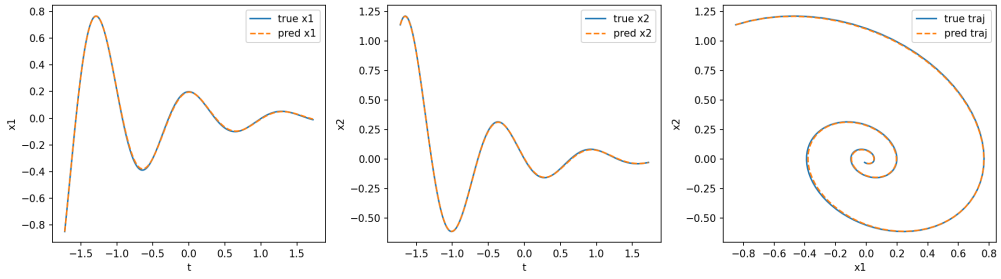


Figure 12: SplineFlow evaluated path for Damped Harmonic Oscillator.

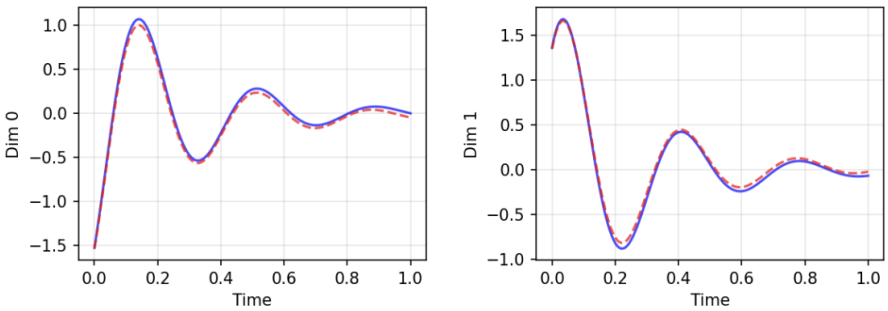


Figure 13: LatentODE evaluated path for Damped Harmonic Oscillator.

I.1.3 Lotka–Volterra ($p = 0.75$)

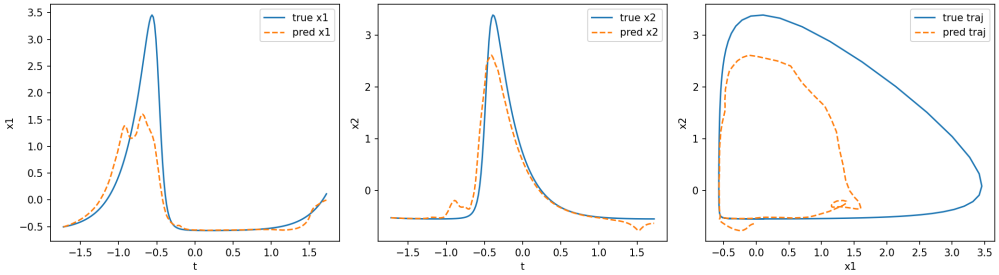


Figure 14: TFM evaluated path for Lotka–Volterra System.

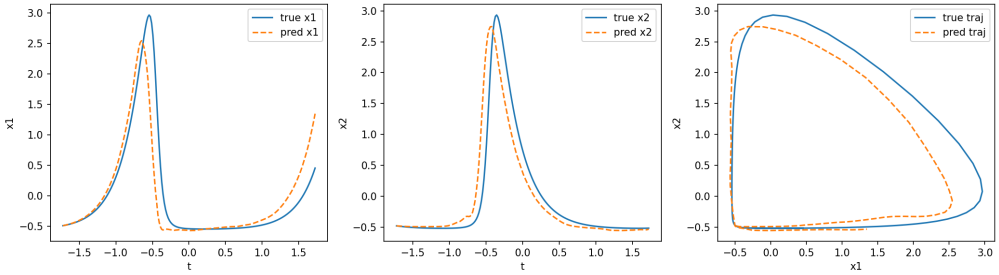


Figure 15: SplineFlow evaluated path for Lotka–Volterra System.

I.2 Chaotic Systems (Lorenz)

I.2.1 Adjoint-Methods ODE Dynamics

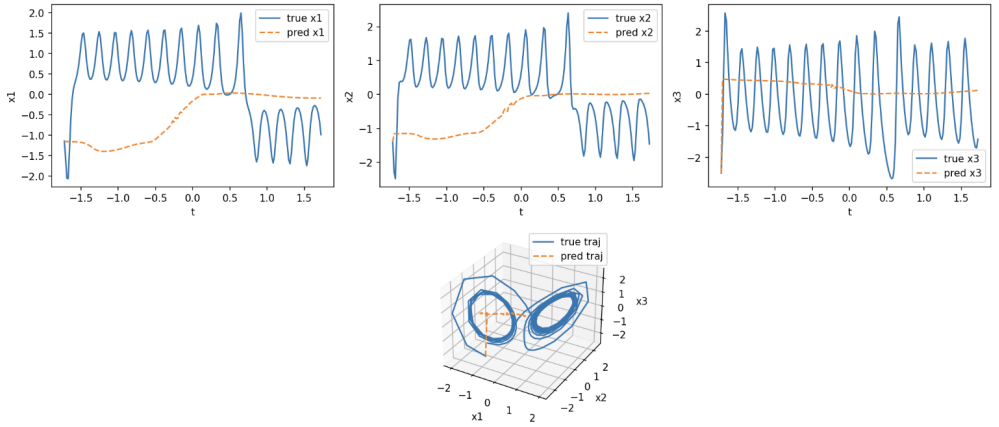


Figure 16: NeuralODE evaluated path for Lorenz System.

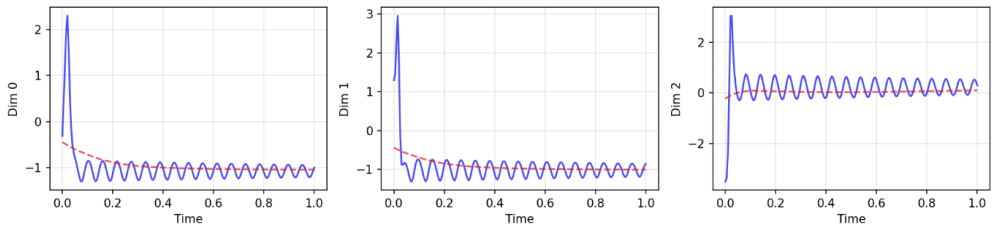


Figure 17: LatentODE evaluated path for Lorenz System.

I.2.2 Trajectory Flow Matching ODE Dynamics

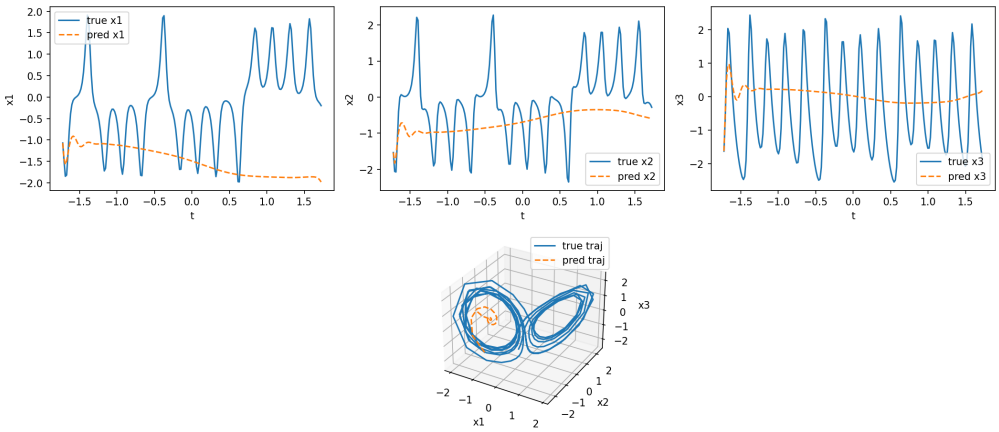


Figure 18: TFM evaluated path for Lorenz System.

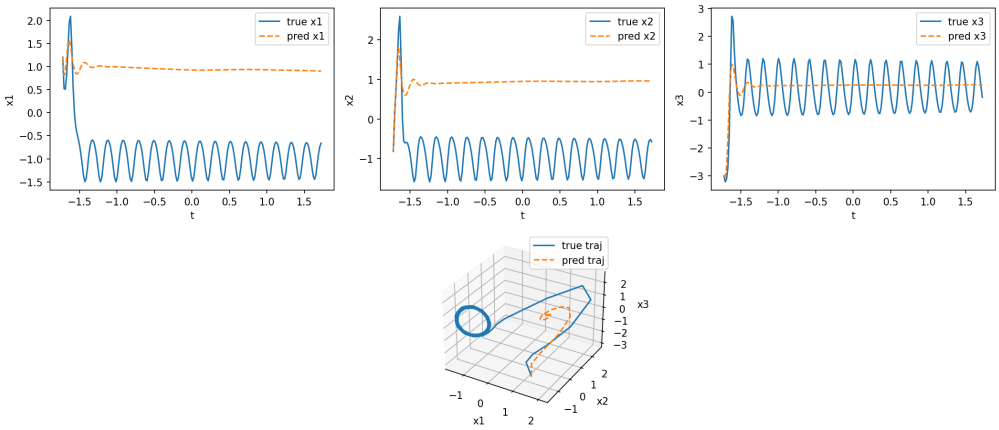


Figure 19: TFM evaluated path for Lorenz System.

I.2.3 SplineFlow ODE Dynamics

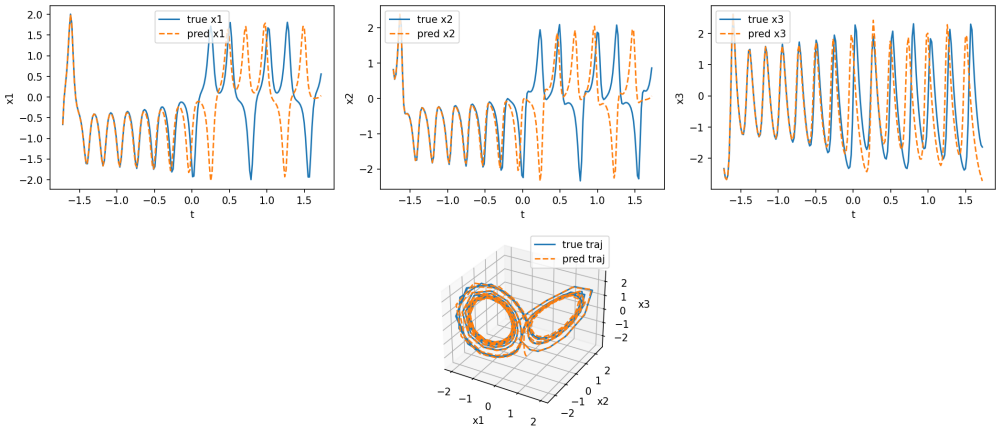


Figure 20: SplineFlow evaluated path for Lorenz System.

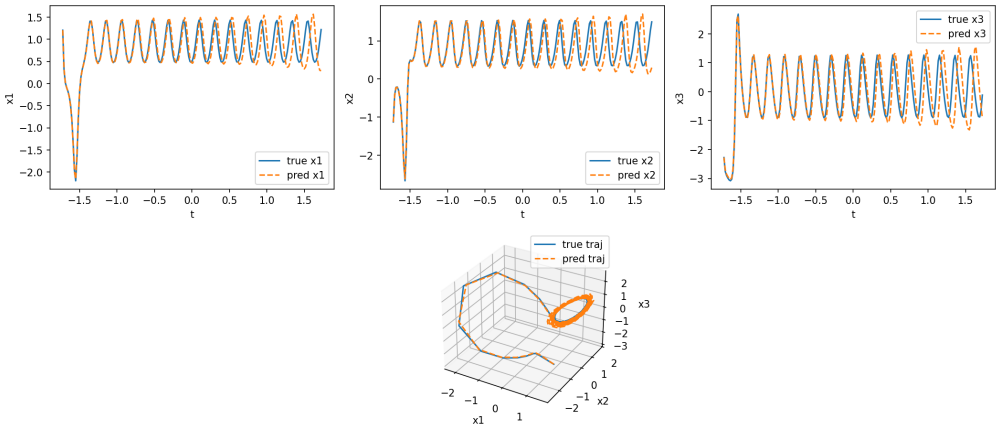


Figure 21: SplineFlow evaluated path for Lorenz System.

I.2.4 SF2M SDE Dynamics

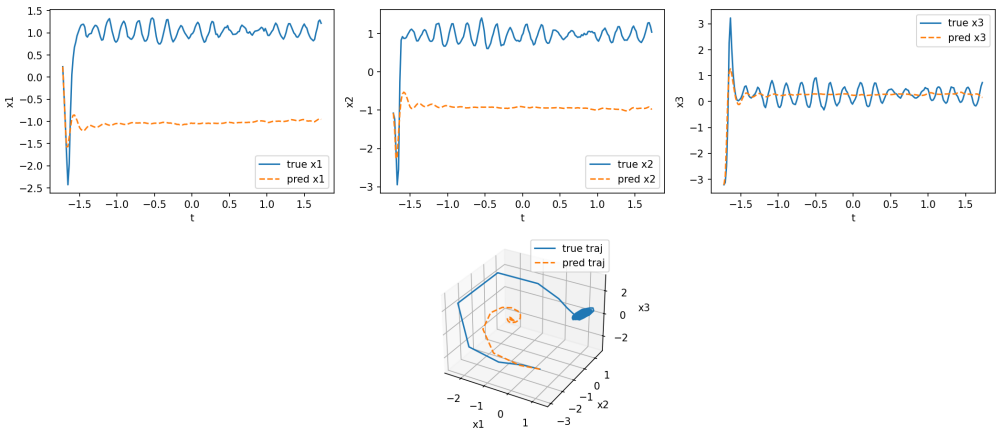


Figure 22: SF2M evaluated path for Lorenz System.

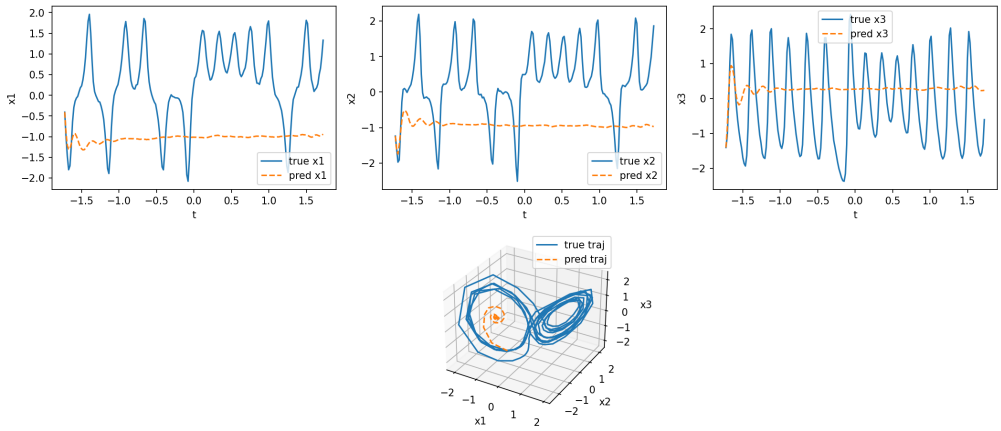


Figure 23: SF2M evaluated path for Lorenz System.

I.2.5 SplineFlow SDE Dynamics

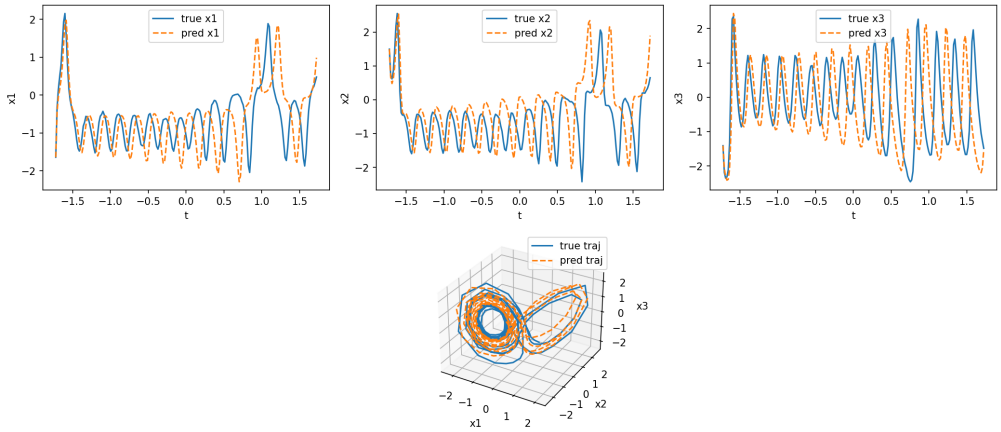


Figure 24: SplineFlow evaluated path for Lorenz System.

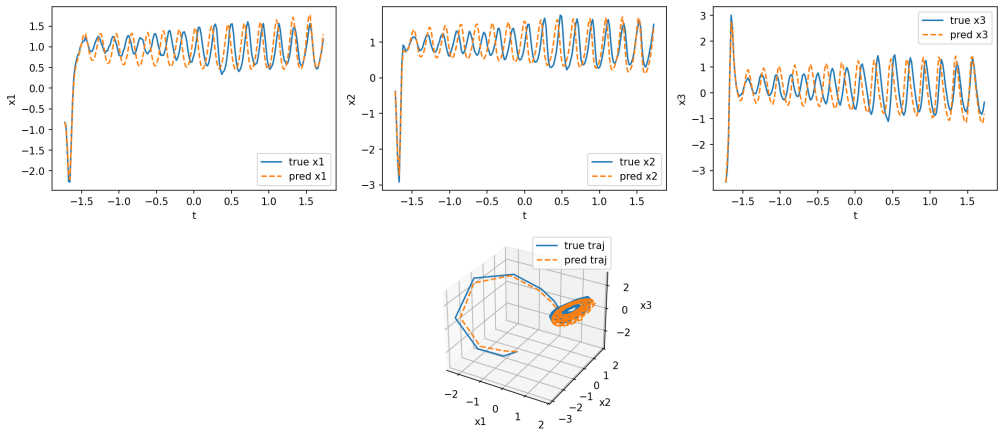


Figure 25: SplineFlow evaluated path for Lorenz System.

J Full Ablations on Irregularity and Spline Degree

J.1 ODE Dynamics (ablations with order k) for non-linear and oscillatory systems

From the ablation studies, we observe that higher degree B-splines are the most useful when modeling at high irregularity ($p > 0$) regimes, for example, Damped Harmonic and Lotka–Volterra. For systems with linear dynamics, such as HopperPhysics, linear degrees perform best.

J.1.1 Damped Harmonic Oscillator

Table 17: Effect of spline degree on MSE performance of SplineFlow on damped harmonic system across missingness probabilities p.

Degree	$p = 0$	$p = 0.25$	$p = 0.5$	$p = 0.75$
1	$1.97e{-2}\pm4.83e{-4}$	$2.05e{-2}\pm2.69e{-3}$	$3.19e{-2}\pm4.18e{-3}$	$1.71e{-1}\pm3.71e{-2}$
2	$6.5e{-5}\pm3.47e{-5}$	$6.2e{-5}\pm2.69e{-5}$	$5.4e{-5}\pm3.44e{-5}$	$4.15e{-4}\pm8.13e{-5}$
3	$3.1e{-5}\pm8.47e{-6}$	$3.4e{-5}\pm9.69e{-6}$	$1.00e{-4}\pm8.68e{-5}$	$1.84e{-4}\pm4.29e{-5}$
4	$7.2e{-5}\pm3.02e{-5}$	$5.1e{-5}\pm3.44e{-5}$	$4.7e{-5}\pm3.37e{-5}$	$3.4e{-5}\pm4.54e{-6}$
5	$9.5e{-5}\pm1.02e{-4}$	$3.0e{-5}\pm4.77e{-6}$	$4.7e{-5}\pm2.14e{-5}$	$4.8e{-5}\pm2.30e{-5}$

J.1.2 Lotka–Volterra

Table 18: Effect of spline degree on MSE performance of SplineFlow on Lotka–Volterra system across missingness probabilities p.

Degree	$p = 0$	$p = 0.25$	$p = 0.5$	$p = 0.75$
1	$4.19e{+0}\pm7.51e{+0}$	$4.29e{+0}\pm6.00e{+0}$	$1.22e{+1}\pm8.84e{+0}$	$2.15e{+2}\pm1.28e{+2}$
2	$5.83e{-2}\pm4.14e{-2}$	$2.56e{-1}\pm2.05e{-1}$	$3.56e{-1}\pm2.24e{-1}$	$6.82e{+0}\pm3.98e{+0}$
3	$4.50e{-2}\pm2.78e{-2}$	$1.87e{-1}\pm1.33e{-1}$	$2.75e{-1}\pm1.89e{-1}$	$8.88e{+0}\pm8.82e{+0}$
4	$5.53e{-2}\pm2.36e{-2}$	$1.83e{-1}\pm1.41e{-1}$	$4.20e{+0}\pm7.96e{+0}$	$4.31e{+0}\pm4.40e{+0}$
5	$8.55e{-2}\pm4.28e{-2}$	$1.00e{-1}\pm5.72e{-2}$	$7.05e{-1}\pm1.21e{+0}$	$1.78e{+0}\pm1.43e{+0}$

J.1.3 HopperPhysics

Table 19: Effect of spline degree on MSE performance of SplineFlow on hopperphysics simulations across missingness probabilities p.

Degree	$p = 0$	$p = 0.25$	$p = 0.5$	$p = 0.75$
1	$1.41e{+0}\pm1.72e{-1}$	$1.43e{+0}\pm2.98e{-2}$	$1.83e{+0}\pm6.19e{-2}$	$3.34e{+0}\pm4.30e{-1}$
2	$6.60e{+0}\pm2.35e{+0}$	$5.39e{+1}\pm2.47e{+1}$	$2.80e{+2}\pm5.93e{+1}$	$1.00e{+3}\pm6.07e{+1}$
3	$1.25e{+1}\pm8.86e{+0}$	$3.91e{+2}\pm1.80e{+2}$	$6.38e{+4}\pm1.24e{+4}$	$1.73e{+4}\pm1.56e{+3}$
4	$1.63e{+1}\pm1.29e{+1}$	$1.41e{+5}\pm2.11e{+4}$	$4.64e{+5}\pm1.50e{+5}$	$1.20e{+5}\pm6.57e{+4}$
5	$2.26e{+1}\pm6.33e{+0}$	$8.95e{+5}\pm1.36e{+5}$	$6.71e{+5}\pm1.30e{+5}$	$9.25e{+4}\pm2.44e{+4}$

J.2 SDE Dynamics (ablations with order k) for linear, non-linear and oscillatory systems

From the ablations in this section, we observe that higher-degree B-splines are useful for modeling nonlinear systems such as the Lotka–Volterra system in the stochastic case.

J.2.1 Exponential Decay ($p = 0$)

Table 20: Effect of spline order on Exp-Decay performance.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$3.91e{-1}\pm4.23e{-3}$	$4.56e{-1}\pm8.27e{-3}$	$3.31e{-1}\pm2.83e{-3}$	$2.73e{-2}\pm2.48e{-3}$	$6.94e{-2}\pm1.08e{-3}$
2	$1.25e{+0}\pm4.04e{-1}$	$2.38e{+0}\pm5.77e{-1}$	$7.94e{-1}\pm1.78e{-1}$	$3.38e{-2}\pm3.08e{-3}$	$1.12e{-1}\pm1.67e{-2}$
3	$8.22e{-1}\pm6.78e{-2}$	$2.03e{+0}\pm4.47e{-1}$	$6.87e{-1}\pm1.70e{-1}$	$3.85e{-2}\pm1.01e{-2}$	$1.25e{-1}\pm3.65e{-2}$
4	$1.20e{+0}\pm6.28e{-1}$	$4.74e{+0}\pm4.43e{+0}$	$1.17e{+0}\pm8.69e{-1}$	$3.46e{-2}\pm1.40e{-2}$	$1.30e{-1}\pm7.66e{-2}$
5	$1.27e{+0}\pm3.89e{-1}$	$9.89e{+0}\pm1.17e{+1}$	$1.49e{+0}\pm1.31e{+0}$	$6.13e{-2}\pm1.22e{-2}$	$2.26e{-1}\pm9.57e{-2}$

J.2.2 Exponential Decay ($p = 0.25$)

Table 21: Effect of B-spline degree on Exp-decay performance for $p = 0.25$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$3.66e{-1}\pm6.08e{-3}$	$4.70e{-1}\pm5.90e{-3}$	$2.51e{-1}\pm5.53e{-3}$	$1.51e{-2}\pm1.03e{-3}$	$4.12e{-2}\pm3.14e{-3}$
2	$7.30e{-1}\pm1.41e{-1}$	$1.49e{+0}\pm3.84e{-2}$	$5.52e{-1}\pm3.67e{-2}$	$4.28e{-2}\pm1.87e{-2}$	$1.43e{-1}\pm6.38e{-2}$
3	$1.37e{+0}\pm8.59e{-1}$	$2.55e{+0}\pm9.96e{-1}$	$7.75e{-1}\pm2.18e{-1}$	$4.36e{-2}\pm1.19e{-2}$	$1.47e{-1}\pm4.29e{-2}$
4	$8.48e{-1}\pm2.68e{-1}$	$2.73e{+0}\pm1.64e{+0}$	$8.58e{-1}\pm3.85e{-1}$	$4.18e{-2}\pm7.33e{-3}$	$1.38e{-1}\pm3.88e{-2}$
5	$3.12e{+0}\pm1.10e{+0}$	$4.41e{+0}\pm8.47e{-1}$	$1.17e{+0}\pm1.97e{-1}$	$6.01e{-2}\pm1.21e{-2}$	$2.60e{-1}\pm4.65e{-2}$

J.2.3 Exponential Decay ($p = 0.5$)

Table 22: Effect of B-spline degree on Exp-decay performance for $p = 0.5$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$3.76e{-1}\pm3.30e{-3}$	$5.30e{-1}\pm6.59e{-3}$	$1.95e{-1}\pm1.42e{-3}$	$7.42e{-3}\pm3.31e{-4}$	$2.40e{-2}\pm1.36e{-3}$
2	$6.44e{-1}\pm5.12e{-2}$	$1.80e{+0}\pm5.27e{-1}$	$5.92e{-1}\pm1.49e{-1}$	$2.92e{-2}\pm8.18e{-3}$	$8.90e{-2}\pm1.62e{-2}$
3	$6.52e{-1}\pm9.32e{-2}$	$1.61e{+0}\pm9.35e{-2}$	$5.57e{-1}\pm4.34e{-2}$	$2.34e{-2}\pm9.21e{-3}$	$7.47e{-2}\pm2.55e{-2}$
4	$8.61e{-1}\pm3.17e{-1}$	$2.78e{+0}\pm1.86e{+0}$	$8.49e{-1}\pm4.06e{-1}$	$3.21e{-2}\pm1.25e{-2}$	$1.34e{-1}\pm5.61e{-2}$
5	$2.75e{+0}\pm2.96e{+0}$	$3.53e{+0}\pm3.21e{+0}$	$9.04e{-1}\pm6.04e{-1}$	$7.77e{-2}\pm5.26e{-2}$	$3.93e{-1}\pm3.49e{-1}$

J.2.4 Exponential Decay ($p = 0.75$)

Table 23: Effect of B-spline degree on Exp-decay performance for $p = 0.75$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$4.19e{-1}\pm5.87e{-3}$	$6.77e{-1}\pm8.23e{-3}$	$1.42e{-1}\pm1.72e{-2}$	$2.06e{-3}\pm1.55e{-3}$	$1.51e{-2}\pm5.15e{-3}$
2	$7.83e{-1}\pm4.41e{-2}$	$2.24e{+0}\pm1.00e{+0}$	$7.53e{-1}\pm2.53e{-1}$	$3.91e{-2}\pm9.25e{-3}$	$1.43e{-1}\pm5.00e{-2}$
3	$9.82e{-1}\pm7.21e{-2}$	$5.56e{+0}\pm3.84e{+0}$	$1.39e{+0}\pm6.96e{-1}$	$4.30e{-2}\pm1.32e{-2}$	$1.69e{-1}\pm3.47e{-2}$
4	$1.12e{+0}\pm9.37e{-1}$	$2.61e{+0}\pm5.93e{-1}$	$9.23e{-1}\pm1.44e{-1}$	$5.42e{-2}\pm2.47e{-2}$	$2.26e{-1}\pm1.46e{-1}$
5	$3.18e{+5}\pm4.50e{+5}$	$3.03e{+5}\pm4.23e{+5}$	$1.99e{+2}\pm2.47e{+2}$	$2.38e{-1}\pm2.83e{-1}$	$2.61e{+2}\pm3.68e{+2}$

J.2.5 Lotka–Volterra ($p = 0$)

Table 24: Effect of spline order on Lotka–Volterra performance.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$4.70e{-1}\pm9.54e{-2}$	$4.69e{-1}\pm9.23e{-2}$	$4.27e{-1}\pm7.98e{-2}$	$1.12e{-1}\pm6.87e{-2}$	$4.26e{-1}\pm2.38e{-1}$
2	$2.68e{-1}\pm1.44e{-2}$	$2.73e{-1}\pm1.37e{-2}$	$2.94e{-1}\pm3.16e{-3}$	$4.21e{-2}\pm3.45e{-3}$	$1.56e{-1}\pm1.33e{-2}$
3	$3.03e{-1}\pm6.56e{-3}$	$3.03e{-1}\pm1.10e{-2}$	$3.21e{-1}\pm8.22e{-3}$	$4.64e{-2}\pm3.21e{-3}$	$1.74e{-1}\pm1.36e{-2}$
4	$3.47e{-1}\pm3.19e{-2}$	$3.47e{-1}\pm3.10e{-2}$	$3.60e{-1}\pm2.25e{-2}$	$6.97e{-2}\pm1.63e{-2}$	$2.60e{-1}\pm5.59e{-2}$
5	$3.28e{-1}\pm2.62e{-2}$	$3.30e{-1}\pm2.62e{-2}$	$3.50e{-1}\pm3.36e{-2}$	$6.87e{-2}\pm2.85e{-2}$	$2.52e{-1}\pm9.45e{-2}$

J.2.6 Lotka–Volterra ($p = 0.25$)

Table 25: Effect of B-spline degree on Lotka–Volterra performance for $p = 0.25$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$5.22e{-1}\pm1.72e{-1}$	$5.24e{-1}\pm1.71e{-1}$	$4.56e{-1}\pm1.33e{-1}$	$1.32e{-1}\pm1.11e{-1}$	$4.89e{-1}\pm3.77e{-1}$
2	$3.40e{-1}\pm5.04e{-2}$	$3.38e{-1}\pm4.76e{-2}$	$3.20e{-1}\pm1.01e{-2}$	$4.57e{-2}\pm2.87e{-3}$	$1.82e{-1}\pm1.18e{-2}$
3	$3.34e{-1}\pm5.05e{-2}$	$3.39e{-1}\pm4.86e{-2}$	$3.37e{-1}\pm1.39e{-2}$	$6.22e{-2}\pm1.12e{-2}$	$2.47e{-1}\pm5.83e{-2}$
4	$3.18e{-1}\pm3.58e{-2}$	$3.21e{-1}\pm3.21e{-2}$	$3.19e{-1}\pm1.21e{-2}$	$4.45e{-2}\pm5.03e{-3}$	$1.75e{-1}\pm2.60e{-2}$
5	$2.81e{-1}\pm1.70e{-2}$	$2.83e{-1}\pm1.54e{-2}$	$3.19e{-1}\pm5.33e{-3}$	$4.83e{-2}\pm3.97e{-3}$	$1.83e{-1}\pm1.57e{-2}$

J.2.7 Lotka–Volterra ($p = 0.5$)

Table 26: Effect of B-spline degree on Lotka–Volterra performance for $p = 0.5$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$3.82e{-1}\pm6.10e{-2}$	$3.83e{-1}\pm5.94e{-2}$	$3.70e{-1}\pm1.68e{-2}$	$7.09e{-2}\pm4.52e{-3}$	$2.75e{-1}\pm1.38e{-2}$
2	$4.11e{-1}\pm4.68e{-2}$	$4.15e{-1}\pm4.79e{-2}$	$3.74e{-1}\pm3.81e{-2}$	$6.24e{-2}\pm9.27e{-3}$	$2.59e{-1}\pm3.66e{-2}$
3	$2.88e{-1}\pm2.32e{-2}$	$2.85e{-1}\pm2.54e{-2}$	$3.10e{-1}\pm1.16e{-2}$	$4.42e{-2}\pm3.80e{-3}$	$1.64e{-1}\pm2.32e{-2}$
4	$3.70e{-1}\pm1.10e{-1}$	$3.72e{-1}\pm1.11e{-1}$	$3.48e{-1}\pm4.99e{-2}$	$5.39e{-2}\pm2.09e{-2}$	$2.22e{-1}\pm1.02e{-1}$
5	$3.19e{-1}\pm2.96e{-2}$	$3.23e{-1}\pm3.05e{-2}$	$3.11e{-1}\pm1.57e{-2}$	$4.50e{-2}\pm2.94e{-3}$	$1.79e{-1}\pm7.45e{-3}$

J.2.8 Lotka–Volterra ($p = 0.75$)

Table 27: Effect of B-spline degree on Lotka–Volterra performance for $p = 0.75$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$4.98e{-1}\pm1.36e{-1}$	$4.99e{-1}\pm1.35e{-1}$	$4.53e{-1}\pm4.31e{-2}$	$1.21e{-1}\pm1.03e{-2}$	$4.53e{-1}\pm7.16e{-2}$
2	$4.30e{-1}\pm7.67e{-2}$	$4.33e{-1}\pm7.35e{-2}$	$3.97e{-1}\pm6.21e{-2}$	$8.44e{-2}\pm3.38e{-2}$	$3.30e{-1}\pm1.15e{-1}$
3	$3.77e{-1}\pm5.73e{-2}$	$3.78e{-1}\pm5.48e{-2}$	$3.61e{-1}\pm4.04e{-2}$	$5.57e{-2}\pm9.43e{-3}$	$2.23e{-1}\pm4.77e{-2}$
4	$4.49e{-1}\pm1.40e{-1}$	$4.43e{-1}\pm1.29e{-1}$	$3.55e{-1}\pm4.84e{-2}$	$6.56e{-2}\pm1.72e{-2}$	$2.75e{-1}\pm9.15e{-2}$
5	$6.03e{-1}\pm2.33e{-1}$	$5.98e{-1}\pm2.31e{-1}$	$4.60e{-1}\pm7.29e{-2}$	$1.26e{-1}\pm4.30e{-2}$	$5.06e{-1}\pm1.91e{-1}$

J.2.9 Damped Harmonic Oscillator ($p = 0$)

Table 28: Effect of spline order on damped harmonic oscillator performance.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$8.95e{-1}\pm5.12e{-3}$	$9.70e{-1}\pm9.41e{-3}$	$4.10e{-1}\pm4.41e{-3}$	$4.19e{-2}\pm1.54e{-3}$	$1.12e{-1}\pm7.12e{-3}$
2	$1.78e{+0}\pm1.96e{-1}$	$8.02e{+0}\pm4.45e{+0}$	$1.58e{+0}\pm6.03e{-1}$	$7.48e{-3}\pm1.30e{-3}$	$8.23e{-2}\pm3.24e{-2}$
3	$5.51e{+0}\pm2.89e{+0}$	$1.30e{+1}\pm7.52e{+0}$	$2.07e{+0}\pm6.98e{-1}$	$2.08e{-2}\pm1.41e{-2}$	$2.06e{-1}\pm1.72e{-1}$
4	$9.23e{+0}\pm2.11e{+0}$	$1.56e{+1}\pm2.84e{+0}$	$2.43e{+0}\pm2.60e{-1}$	$1.69e{-2}\pm7.13e{-3}$	$1.90e{-1}\pm4.45e{-2}$
5	$8.53e{+0}\pm5.52e{+0}$	$2.21e{+1}\pm1.21e{+1}$	$3.04e{+0}\pm7.89e{-1}$	$2.13e{-2}\pm1.77e{-3}$	$2.67e{-1}\pm1.09e{-1}$

J.2.10 Damped Harmonic Oscillator ($p = 0.25$)

Table 29: Effect of B-spline degree on Damped-Harmonic performance for $p = 0.25$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$7.99e{-1}\pm1.57e{-2}$	$9.14e{-1}\pm2.28e{-2}$	$3.65e{-1}\pm2.11e{-2}$	$3.42e{-2}\pm4.34e{-3}$	$9.17e{-2}\pm1.07e{-2}$
2	$2.22e{+0}\pm8.18e{-1}$	$3.48e{+0}\pm7.77e{-1}$	$8.66e{-1}\pm1.83e{-1}$	$1.43e{-2}\pm2.24e{-3}$	$9.77e{-2}\pm7.13e{-3}$
3	$1.49e{+0}\pm3.66e{-2}$	$6.24e{+0}\pm2.46e{+0}$	$1.33e{+0}\pm4.36e{-1}$	$7.18e{-3}\pm3.63e{-3}$	$7.58e{-2}\pm2.36e{-2}$
4	$4.11e{+0}\pm8.82e{-1}$	$9.91e{+0}\pm3.36e{+0}$	$1.76e{+0}\pm2.72e{-1}$	$2.62e{-2}\pm2.77e{-3}$	$1.96e{-1}\pm2.59e{-2}$
5	$7.85e{+0}\pm3.86e{+0}$	$1.52e{+1}\pm7.33e{+0}$	$2.34e{+0}\pm7.83e{-1}$	$3.32e{-2}\pm2.33e{-2}$	$3.66e{-1}\pm1.77e{-1}$

J.2.11 Damped Harmonic Oscillator ($p = 0.5$)

Table 30: Effect of B-spline degree on Damped-Harmonic performance for $p = 0.5$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$8.05e{-1}\pm8.73e{-3}$	$9.81e{-1}\pm3.99e{-2}$	$3.21e{-1}\pm2.84e{-2}$	$2.18e{-2}\pm5.59e{-3}$	$6.87e{-2}\pm1.21e{-2}$
2	$1.61e{+0}\pm4.11e{-1}$	$9.25e{+0}\pm7.31e{+0}$	$1.52e{+0}\pm9.81e{-1}$	$1.23e{-2}\pm7.06e{-3}$	$1.04e{-1}\pm6.55e{-2}$
3	$1.34e{+0}\pm2.35e{-1}$	$3.23e{+0}\pm1.62e{+0}$	$7.52e{-1}\pm3.50e{-1}$	$1.05e{-2}\pm4.37e{-3}$	$7.49e{-2}\pm2.28e{-2}$
4	$1.64e{+0}\pm4.51e{-1}$	$1.38e{+1}\pm1.04e{+1}$	$2.08e{+0}\pm1.07e{+0}$	$1.02e{-2}\pm3.90e{-3}$	$9.98e{-2}\pm4.27e{-2}$
5	$4.26e{+0}\pm2.48e{+0}$	$2.05e{+1}\pm2.01e{+1}$	$2.71e{+0}\pm1.93e{+0}$	$5.87e{-3}\pm2.05e{-3}$	$6.62e{-2}\pm2.93e{-2}$

J.2.12 Damped Harmonic Oscillator ($p = 0.75$)

Table 31: Effect of B-spline degree on Damped-Harmonic performance for $p = 0.75$.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$9.38e{-1}\pm2.89e{-2}$	$1.17e{+0}\pm1.17e{-2}$	$2.86e{-1}\pm1.10e{-2}$	$1.24e{-2}\pm7.65e{-4}$	$5.28e{-2}\pm4.89e{-3}$
2	$2.24e{+0}\pm6.32e{-1}$	$7.16e{+0}\pm5.03e{-1}$	$1.56e{+0}\pm1.18e{-1}$	$1.29e{-2}\pm4.25e{-3}$	$1.02e{-1}\pm3.60e{-2}$
3	$3.34e{+0}\pm2.64e{+0}$	$8.62e{+0}\pm5.79e{+0}$	$1.43e{+0}\pm5.99e{-1}$	$1.13e{-2}\pm1.98e{-3}$	$9.04e{-2}\pm1.63e{-2}$
4	$7.48e{+0}\pm8.12e{+0}$	$3.67e{+1}\pm2.28e{+1}$	$3.83e{+0}\pm1.39e{+0}$	$3.72e{-2}\pm3.14e{-2}$	$2.98e{-1}\pm2.59e{-1}$
5	$8.79e{+1}\pm1.22e{+2}$	$4.87e{+1}\pm5.88e{+1}$	$3.11e{+0}\pm2.19e{+0}$	$3.30e{-2}\pm3.02e{-2}$	$3.06e{-1}\pm3.25e{-1}$

J.3 Chaotic Lorenz Systems

In this section, we conduct ablation studies on the Lorenz system, a highly chaotic system. We observe that higher-degree interpolants are especially effective for modeling stochastic chaotic dynamics, whereas all degree $m > 2$ variants perform better than the linear baseline in the deterministic cases.

J.3.1 Lorenz ODE ($p = 0$)

Table 32: Lorenz system results for regularly sampled observations $p = 0$.

Degree	MSE
1	$1.380 \pm 1.890\text{e}{-02}$
2	$6.800\text{e}{-01} \pm 1.520\text{e}{-02}$
3	$6.400\text{e}{-01} \pm 3.790\text{e}{-03}$
4	$7.800\text{e}{-01} \pm 1.320\text{e}{-01}$
5	$9.200\text{e}{-01} \pm 7.360\text{e}{-02}$

J.3.2 Lorenz SDE ($p = 0$)

Table 33: Effect of spline order on stochastic Lorenz system performance.

Degree	MSE_{PODE}	MSE_{SDE}	Wasserstein	MMD	Energy
1	$2.21e{+0}\pm9.71e{-3}$	$2.21e{+0}\pm9.67e{-3}$	$6.31e{-1}\pm2.03e{-3}$	$1.30e{-1}\pm1.22e{-3}$	$4.01e{-1}\pm3.50e{-3}$
2	$2.19e{+0}\pm1.38e{-2}$	$2.19e{+0}\pm1.88e{-2}$	$6.30e{-1}\pm1.90e{-3}$	$1.29e{-1}\pm2.21e{-4}$	$3.98e{-1}\pm1.52e{-3}$
3	$4.82e{+0}\pm4.96e{+0}$	$3.71e{+0}\pm3.37e{+0}$	$9.31e{-1}\pm9.85e{-1}$	$7.44e{-3}\pm6.53e{-3}$	$5.61e{-2}\pm3.44e{-2}$
4	$1.23e{+0}\pm4.01e{-2}$	$1.23e{+0}\pm5.62e{-2}$	$1.80e{-1}\pm1.47e{-2}$	$4.51e{-3}\pm1.79e{-3}$	$3.66e{-2}\pm7.65e{-3}$
5	$1.36e{+0}\pm6.41e{-2}$	$1.50e{+0}\pm2.46e{-1}$	$3.07e{-1}\pm1.49e{-1}$	$6.44e{-3}\pm1.73e{-3}$	$4.47e{-2}\pm1.08e{-2}$

J.4 Experiments on Transcriptomics Datasets

J.4.1 Brain Regeneration

From the ablation study, we observe that higher-degree variants perform especially well with PHATE embeddings. While the degree $m = 2$ is often the optimal variant for PCA embedding. This aligns

with PHATE embeddings, which are designed to capture the structural organization of datasets and thus benefit from B-splines.

J.4.2 PHATE Embedding: Interpolation

Table 34: Effect of B-spline degree on PHATE interpolation performance.

Degree	Wasserstein	MMD	Energy
1	$7.714\text{e}{-01} \pm 1.250\text{e}{-02}$	$4.813\text{e}{-01} \pm 1.740\text{e}{-02}$	$1.391\text{e}{+00} \pm 7.250\text{e}{-02}$
2	$6.421\text{e}{-01} \pm 5.280\text{e}{-02}$	$3.088\text{e}{-01} \pm 2.570\text{e}{-02}$	$9.172\text{e}{-01} \pm 9.580\text{e}{-02}$
3	$6.519\text{e}{-01} \pm 2.100\text{e}{-02}$	$3.174\text{e}{-01} \pm 5.200\text{e}{-03}$	$9.277\text{e}{-01} \pm 2.030\text{e}{-02}$
4	$6.548\text{e}{-01} \pm 2.930\text{e}{-02}$	$3.120\text{e}{-01} \pm 2.020\text{e}{-02}$	$9.189\text{e}{-01} \pm 7.080\text{e}{-02}$
5	$6.066\text{e}{-01} \pm 2.710\text{e}{-02}$	$2.846\text{e}{-01} \pm 6.600\text{e}{-03}$	$8.152\text{e}{-01} \pm 2.140\text{e}{-02}$

J.4.3 PHATE Embedding: Extrapolation

Table 35: Effect of B-spline degree on PHATE extrapolation performance.

Degree	Wasserstein	MMD	Energy
1	$1.588 \pm 2.270\text{e}{-02}$	$9.796\text{e}{-01} \pm 4.000\text{e}{-03}$	$3.843 \pm 3.700\text{e}{-02}$
2	$1.584 \pm 3.670\text{e}{-02}$	$9.654\text{e}{-01} \pm 6.300\text{e}{-03}$	$3.787 \pm 6.170\text{e}{-02}$
3	$1.587 \pm 1.700\text{e}{-02}$	$9.623\text{e}{-01} \pm 2.700\text{e}{-03}$	$3.761 \pm 2.500\text{e}{-02}$
4	$1.536 \pm 1.920\text{e}{-02}$	$9.550\text{e}{-01} \pm 8.700\text{e}{-03}$	$3.740 \pm 4.240\text{e}{-02}$
5	$1.545 \pm 2.720\text{e}{-02}$	$9.644\text{e}{-01} \pm 3.200\text{e}{-03}$	$3.751 \pm 1.290\text{e}{-02}$

J.4.4 PCA Embedding: Interpolation

Table 36: Effect of B-spline degree on PCA interpolation performance.

Degree	Wasserstein	MMD	Energy
1	$8.719\text{e}{-01} \pm 3.180\text{e}{-02}$	$6.320\text{e}{-02} \pm 3.600\text{e}{-03}$	$1.120 \pm 4.510\text{e}{-02}$
2	$8.380\text{e}{-01} \pm 1.370\text{e}{-02}$	$5.820\text{e}{-02} \pm 2.500\text{e}{-03}$	$1.019 \pm 3.930\text{e}{-02}$
3	$8.346\text{e}{-01} \pm 1.670\text{e}{-02}$	$6.030\text{e}{-02} \pm 1.200\text{e}{-03}$	$1.035 \pm 1.660\text{e}{-02}$
4	$8.502\text{e}{-01} \pm 1.110\text{e}{-02}$	$6.140\text{e}{-02} \pm 3.000\text{e}{-03}$	$1.063 \pm 3.660\text{e}{-02}$
5	$8.512\text{e}{-01} \pm 1.940\text{e}{-02}$	$6.150\text{e}{-02} \pm 2.600\text{e}{-03}$	$1.063 \pm 3.940\text{e}{-02}$

J.4.5 PCA Embedding: Extrapolation

Table 37: Effect of B-spline degree on PCA extrapolation performance.

Degree	Wasserstein	MMD	Energy
1	$8.434\text{e}{-01} \pm 2.500\text{e}{-02}$	$1.730\text{e}{-01} \pm 8.800\text{e}{-03}$	$1.869 \pm 2.100\text{e}{-02}$
2	$8.040\text{e}{-01} \pm 3.720\text{e}{-02}$	$1.292\text{e}{-01} \pm 6.800\text{e}{-03}$	$1.645 \pm 5.660\text{e}{-02}$
3	$8.622\text{e}{-01} \pm 4.030\text{e}{-02}$	$1.415\text{e}{-01} \pm 7.900\text{e}{-03}$	$1.738 \pm 7.230\text{e}{-02}$
4	$8.206\text{e}{-01} \pm 4.100\text{e}{-02}$	$1.355\text{e}{-01} \pm 1.180\text{e}{-02}$	$1.736 \pm 8.500\text{e}{-02}$
5	$8.234\text{e}{-01} \pm 1.070\text{e}{-02}$	$1.336\text{e}{-01} \pm 1.310\text{e}{-02}$	$1.690 \pm 1.068\text{e}{-01}$

J.4.6 Embryoid Evolution

From the ablation study, we observe that, in almost all cases, the better-performing variant degree is greater than linear for both PHATE and PCA embeddings. The higher-degree variants are especially suitable for extrapolation settings with PCA embeddings.

J.4.7 PHATE Embedding: Interpolation

Table 38: Effect of B-spline degree on PHATE interpolation performance.

Degree	Wasserstein	MMD	Energy
1	$2.355\text{e}{-01} \pm 3.280\text{e}{-02}$	$2.783\text{e}{-02} \pm 5.360\text{e}{-03}$	$7.954\text{e}{-02} \pm 1.660\text{e}{-02}$
2	$2.745\text{e}{-01} \pm 3.700\text{e}{-02}$	$1.768\text{e}{-02} \pm 3.220\text{e}{-03}$	$6.331\text{e}{-02} \pm 1.540\text{e}{-02}$
3	$2.071\text{e}{-01} \pm 1.690\text{e}{-02}$	$1.260\text{e}{-02} \pm 5.870\text{e}{-04}$	$3.963\text{e}{-02} \pm 1.400\text{e}{-03}$
4	$2.560\text{e}{-01} \pm 6.300\text{e}{-02}$	$1.592\text{e}{-02} \pm 4.450\text{e}{-03}$	$5.462\text{e}{-02} \pm 1.620\text{e}{-02}$
5	$2.217\text{e}{-01} \pm 3.530\text{e}{-02}$	$1.408\text{e}{-02} \pm 3.100\text{e}{-03}$	$4.467\text{e}{-02} \pm 9.410\text{e}{-03}$

J.4.8 PHATE Embedding: Extrapolation

Table 39: Effect of B-spline degree on PHATE extrapolation performance.

Degree	Wasserstein	MMD	Energy
1	$5.765\text{e}{-01} \pm 5.860\text{e}{-02}$	$7.572\text{e}{-02} \pm 1.110\text{e}{-02}$	$2.833\text{e}{-01} \pm 4.370\text{e}{-02}$
2	$4.483\text{e}{-01} \pm 4.410\text{e}{-02}$	$6.902\text{e}{-02} \pm 1.700\text{e}{-03}$	$2.073\text{e}{-01} \pm 2.960\text{e}{-02}$
3	$4.774\text{e}{-01} \pm 6.720\text{e}{-02}$	$6.540\text{e}{-02} \pm 9.270\text{e}{-03}$	$2.367\text{e}{-01} \pm 5.200\text{e}{-02}$
4	$5.502\text{e}{-01} \pm 7.380\text{e}{-02}$	$7.318\text{e}{-02} \pm 1.650\text{e}{-02}$	$2.916\text{e}{-01} \pm 8.900\text{e}{-02}$
5	$4.733\text{e}{-01} \pm 6.450\text{e}{-02}$	$7.417\text{e}{-02} \pm 3.690\text{e}{-03}$	$2.423\text{e}{-01} \pm 2.410\text{e}{-02}$

J.4.9 PCA Embedding: Interpolation

Table 40: Effect of B-spline degree on PCA interpolation performance.

Degree	Wasserstein	MMD	Energy
1	$3.008\text{e}{-01} \pm 6.130\text{e}{-03}$	$1.341\text{e}{-02} \pm 7.060\text{e}{-04}$	$1.913\text{e}{-01} \pm 8.150\text{e}{-03}$
2	$3.073\text{e}{-01} \pm 1.260\text{e}{-02}$	$1.231\text{e}{-02} \pm 5.310\text{e}{-04}$	$2.070\text{e}{-01} \pm 9.860\text{e}{-03}$
3	$3.080\text{e}{-01} \pm 1.510\text{e}{-02}$	$1.328\text{e}{-02} \pm 1.680\text{e}{-03}$	$2.134\text{e}{-01} \pm 2.560\text{e}{-02}$
4	$2.959\text{e}{-01} \pm 1.460\text{e}{-02}$	$1.267\text{e}{-02} \pm 3.110\text{e}{-04}$	$1.994\text{e}{-01} \pm 9.340\text{e}{-03}$
5	$3.248\text{e}{-01} \pm 9.690\text{e}{-03}$	$1.358\text{e}{-02} \pm 4.650\text{e}{-04}$	$2.230\text{e}{-01} \pm 1.100\text{e}{-02}$

J.4.10 PCA Embedding: Extrapolation

Table 41: Effect of B-spline degree on PCA extrapolation performance.

Degree	Wasserstein	MMD	Energy
1	$5.652\text{e}{-01} \pm 2.770\text{e}{-02}$	$3.352\text{e}{-02} \pm 1.930\text{e}{-03}$	$6.546\text{e}{-01} \pm 5.310\text{e}{-02}$
2	$5.861\text{e}{-01} \pm 2.590\text{e}{-02}$	$3.790\text{e}{-02} \pm 2.500\text{e}{-03}$	$7.244\text{e}{-01} \pm 5.470\text{e}{-02}$
3	$5.387\text{e}{-01} \pm 3.350\text{e}{-02}$	$3.583\text{e}{-02} \pm 4.420\text{e}{-03}$	$6.895\text{e}{-01} \pm 1.010\text{e}{-01}$
4	$5.389\text{e}{-01} \pm 2.420\text{e}{-02}$	$3.469\text{e}{-02} \pm 2.410\text{e}{-03}$	$6.588\text{e}{-01} \pm 5.070\text{e}{-02}$
5	$5.871\text{e}{-01} \pm 1.240\text{e}{-02}$	$3.842\text{e}{-02} \pm 3.320\text{e}{-03}$	$7.504\text{e}{-01} \pm 5.620\text{e}{-02}$

J.5 Comparison between SplineFlow with $m = 3$ and MMFM [Rohbeck et al., 2025]

Table 42: Comparison of MMFM and SplineFlow across ODE systems and sampling irregularity p.

p	Model	Exp.	Damp.	LV	Lorenz
0	MMFM	1.40e−3	2.00e−3	4.87e−1	8.02e−1
	SplineFlow ($m = 3$)	**9.23e−5**	**3.10e−5**	**4.50e−2**	**6.39e−1**
0.25	MMFM	2.00e−3	1.80e−3	3.72e−1	–
	SplineFlow ($m = 3$)	**8.42e−5**	**2.90e−5**	**1.87e−1**	–
0.5	MMFM	1.70e−3	3.00e−3	4.41	–
	SplineFlow ($m = 3$)	**1.13e−4**	**1.00e−4**	**2.75e−1**	–
0.75	MMFM	2.40e−3	4.00e−3	**8.62**	–
	SplineFlow ($m = 3$)	**9.00e−5**	**1.84e−4**	8.88	–

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our main claims in the abstract and introduction revolve around the limitations of existing flow matching techniques that use linear interpolants to model dynamical systems and the proposal of SplineFlow to mitigate those limitations. In Sections 3.3 and 4, we derive theoretical results showing that using b-splines leads to lower function approximation error, which in turn yields better approximation of the underlying dynamics by the velocity fields and better sample quality. Our exhaustive experiments in Section 5 demonstrate the benefits of using higher-order B-splines across ODE, SDE, and cellular dynamics settings.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations in terms of more clarity with respect to optimal knot placement in splines, and lack of provision for non-additive diffusion terms for modeling the SDE dynamics.

   Guidelines:

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

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Yes, the formal statements of our theorems are mentioned in Appendix D, and we discuss all the necessary assumptions and conditions under which the guarantees hold.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We've included all the implementation details, such as network architecture, hyperparameter settings and runtime, in Appendix G.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Yes, we have released the code; the link is in the Availability section.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We've included the training details, such as network architecture, hyperparameter settings, runtime, etc., in Appendix G.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We've provided mean $\pm$ std across several seeds for all the metrics, and full ablation studies in Appendix J, including the condensed results in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We've included the details (network architecture, hyperparameter settings, and runtime) in Appendix G. All the experiments were run on a single Nvidia A100 GPU.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We've read the guidelines and confirm that we're not in violation of any code of ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

Justification: The paper present a general principled methodology to extend flow matching for dynamical systems, rather than specific applications with direct societal implications.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: We built on prior codebase from baselines as well as commonly used benchmarks used for dynamical systems

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Yes, we have released the codebase for SplineFlow along with its core additions.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

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