Generative Modeling with Flux Matching

Peter Pao-Huang[1] **Xiaojie Qiu**[1,2] **Stefano Ermon**[1]
[1]Department of Computer Science, Stanford University
[2]Department of Genetics, Stanford University
{peterph,xiaojie,ermon}@stanford.edu

Abstract

We introduce *Flux Matching*, a new paradigm for generative modeling that generalizes existing score-based models to a broader family of vector fields that need not be conservative. Rather than requiring the model to equal the data score, the Flux Matching objective imposes a weaker condition that admits infinitely many vector fields whose stationary distribution is the data. This flexibility enables a class of generative models that cannot be learned under score matching, in which inductive biases, structural priors, and properties of the dynamics can be directly imposed or optimized. We show that Flux Matching performs strongly on high-dimensional image datasets and, more importantly, that our added freedom unlocks a range of applications including faster sampling, interpretable and mechanistic models, and dynamics that encode directed dependencies between variables. More broadly, Flux Matching opens a new dimension in generative modeling by turning the vector field itself into a design choice rather than a fixed target. Code is available at `https://github.com/peterpaohuang/flux_matching`.

1 Introduction

Many different vector fields produce diffusion processes with the same stationary distribution. Modern generative modeling [58, 59, 25], however, canonically targets one particular vector field called the (Stein) score function, typically fit via score matching [31, 31, 59], whose population loss is the Fisher divergence. Once learned, the score model can be used in Langevin dynamics or other gradient-based Markov chain Monte Carlo (MCMC) methods to generate samples from the target distribution p_{data}. This score-based paradigm dominants current state-of-the-art image generation models [54, 12, 51], protein generation and design models [63, 1], robotics [9, 50], and others [26, 35, 21, 11, 32, 64].

The narrow focus on the score overlooks a large space of alternative vector fields whose diffusion processes share the same target distribution. We refer to these vector fields as *generative vector fields*, or generative fields for short, with the Fokker–Planck equation (FPE) characterizing the full family [27]. Figure 1 highlights the distinction: in the space of vector fields Ω, score matching picks out a single point, $\nabla \log p_{\text{data}}$, when any other point in the rectangle of generative fields characterized by the FPE is also equally valid. These non-score generative fields provide an extra degree of freedom for encoding useful attributes, illustrated abstractly by Attributes A and B in Figure 1. Concretely, they can capture directed dependencies between variables, impose mechanistic structure, improve smoothness or mixing, and produce dynamics that are meaningful in their own right rather than merely a means of sampling.

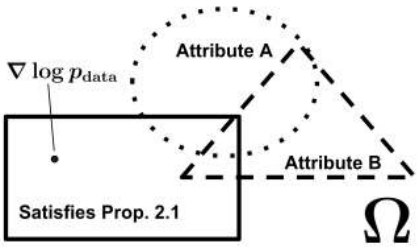


Figure 1: Ω is the space of vector fields $\in L^2(p_{\text{data}})$. Score matching learns $\nabla \log p_{\text{data}}$, a single point in this space. In contrast, *Flux Matching* can learn any vector field inside the rectangle.

Preprint.

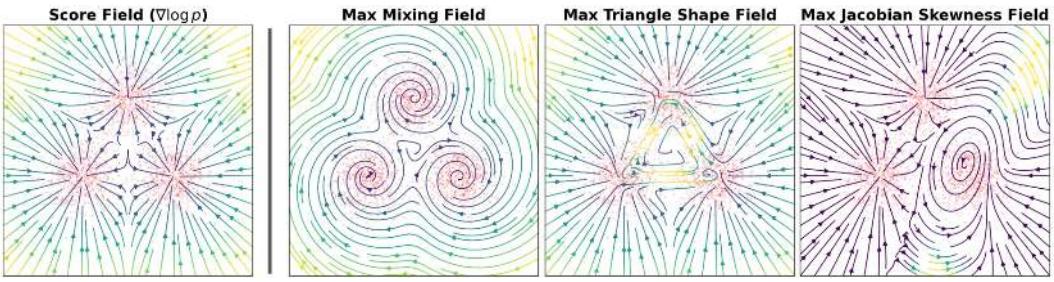


Figure 2: From Section 4.1 where we maximize different vector field attributes that generate the same stationary distribution. (**Left**) Score function (**Right**) Alternative vector fields with useful properties.

In this work, we propose *Flux Matching*, a novel paradigm for learning generative vector fields beyond the score (aka any point inside the rectangle of Figure 1). Instead of requiring the model to equal $\nabla \log p_{\text{data}}$ pointwise, Flux Matching requires a weaker condition that only the divergence of the probability flux matches. This condition guarantees that the field generates the target distribution while leaving a nullspace of infinitely many valid generative fields. In order to compare the flux divergences in the same $L^2(p_{\text{data}})$ geometry as the Fisher divergence—while preserving the non-score degrees of freedom—we define a new statistical divergence called the *projected Fisher divergence* and derive a tractable Flux Matching loss that computes it. We further extend Flux Matching to the noise annealed setting used by diffusion models [59, 58]. Rather than learning one field for the data distribution, we learn a continuum of fields for increasingly noise annealed distributions. Among the many valid vector fields that generate the target distribution, we also show how to select application-specific solutions either through architectural constraints or by adding regularizers that favor desired attributes.

Empirically, we show that Flux Matching is both scalable and useful in varying domains: (1) Flux Matching can be used as a standalone generative objective on high-dimensional image datasets such as CIFAR-10 and CelebA 64×64; (2) Flux Matching can learn faster mixing fields to accelerate sampling speed; (3) Flux Matching can fit interpretable RNA-velocity in single-cell genomics; and (4) Flux Matching can embed structural priors, such as directed temporal dependencies, directly into the generative field.

To summarize, our contributions are:

- We introduce *Flux Matching*, a generative modeling paradigm that learns vector fields beyond the score by matching the divergence of the probability flux.
- We derive an efficient Flux Matching loss that preserves the Fisher divergence geometry.
- We extend Flux Matching to noise annealed generative modeling and show that it scales to complex image distributions.
- We demonstrate new use cases enabled by non-score generative fields, including faster mixing, interpretable fields like RNA velocity, and structured generative dynamics.

2 Preliminaries

Let p_{data} denote an unknown data distribution on $\mathbb{R}^d$, observed only through samples $\{x_i\}_{i=1}^n \sim p_{\text{data}}$. A key goal of generative modeling is to learn a representation of p_{data} that allows us to generate new samples from this distribution. Existing approaches do this by either modeling the density itself [13, 47], an unnormalized density [15, 39, 22, 24], or the closely related score function [31, 59, 25].

2.1 The (Stein) score function $\nabla \log p_{\text{data}}(x)$

Learning. If the score was directly available, we could fit a vector field $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ by minimizing the Fisher divergence:

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\| f_\theta(x) - \nabla \log p_{\text{data}}(x) \|^2 \right]. \tag{1}$$

However, the score $\nabla \log p_{\text{data}}(x)$ is typically inaccessible because p_{data} itself is unknown. Score-based methods therefore rely on objectives that avoid direct access to this target, including implicit

score matching [31], denoising score matching [62], and nonparametric kernel density estimation (KDE) approximations [31, 62].

Sampling. Once a score estimator $f_\theta \approx \nabla \log p_{\text{data}}$ has been learned, we can sample from p_{data} by simulating the diffusion $dx_t = f_\theta(x_t)\,dt + \sqrt{2}\,dW_t$, where W_t is standard Brownian motion. In practice, we run gradient-based Markov chain Monte Carlo (MCMC) methods that discretize this diffusion. A standard example is unadjusted Langevin dynamics, $x_{k+1} = x_k + \eta f_\theta(x_k) + \sqrt{2\eta}\,\xi_k$ with $\xi_k \sim \mathcal{N}(0, I)$, which (under mild regularity) converges to p_{data} as $\eta \to 0$ and $k \to \infty$. For brevity, we say that vector field f_θ *generates* p_{data} (or samples from p_{data}) as shorthand for when the diffusion with drift f_θ has p_{data} as its stationary distribution.

2.2 The Fokker–Planck equation

To understand why the score is useful for sampling, it is helpful to look at how densities evolve under stochastic dynamics. Again, consider the diffusion $dx_t = f_\theta(x_t)\,dt + \sqrt{2}\,dW_t$. If p_t denotes the time t marginal density of x_t, then p_t evolves according to the Fokker–Planck equation:

$$\frac{\partial p_t(x)}{\partial t} = -\Big[\nabla \cdot \underbrace{\Big(p_t(x) f_\theta(x)\Big)}_{\text{flux of } f_\theta} - \nabla \cdot \underbrace{\Big(p_t(x) \nabla \log p_t(x)\Big)}_{\text{flux of score}}\Big]. \tag{2}$$

Intuitively, Equation (2) is a continuity equation in which the density evolves under the competing divergences of two *probability fluxes*, the drift flux $p_t(x) f_\theta(x)$ carrying mass along the vector field f_θ and the score flux $p_t(x) \nabla \log p_t(x)$ encoding the smoothing effect of Brownian motion. At stationarity, the two forces perfectly balance and the density no longer changes in time, so $\partial_t p_t = 0$, which gives the following:

Proposition 2.1. *[Classical stationary Fokker–Planck characterization, Section 2.4 of [48]]* p_{data} *is a stationary distribution of the diffusion* $dx_t = f_\theta(x_t)\,dt + \sqrt{2}\,dW_t$ *iff*

$$\nabla \cdot \big(p_{\text{data}}(x) f_\theta(x)\big) - \nabla \cdot \big(p_{\text{data}}(x) \nabla \log p_{\text{data}}(x)\big) = 0 \qquad \textit{for all } x.$$

We therefore refer to vector fields f_θ satisfying Proposition 2.1 as *generative vector fields*, since simulating the corresponding diffusion produces samples from p_{data}. The usual choice is $f_\theta(x) = \nabla \log p_{\text{data}}(x)$, for which the condition holds trivially. Importantly, however, Proposition 2.1 shows that the score is *not* the only valid drift. Any vector field of the form

$$f_\theta(x) = \nabla \log p_{\text{data}}(x) + v(x) \quad \text{with} \quad \nabla \cdot \big(p_{\text{data}}(x) v(x)\big) = 0, \tag{3}$$

has the same stationary distribution p_{data}. In other words, there is generally a whole family of vector fields that preserve the same target distribution, and the score is only one particular member of this family. Pictorially, some alternative vector fields are shown in Figure 2.

3 Flux Matching

We now shift from learning f_θ by matching the score to learning f_θ by matching the divergence of the *probability flux* the score induces, which we call *Flux Matching*. The motivation comes directly from the Fokker–Planck equation, where if our goal is to ensure that f_θ generates p_{data}, then it is only necessary to match the flux divergence $\nabla \cdot (p_{\text{data}} f_\theta)$ (from Proposition 2.1) rather than to match the vector field pointwise. Consequently, Flux Matching allows learning the family of generative vector fields that need not be the score (e.g. Figure 2).

New Capability: *Same* Distribution, *Many* Dynamics. Flux Matching enables vector fields to follow any dynamics that generate the target distribution.

3.1 Projected Fisher Divergence

Matching $\nabla \cdot (p_{\text{data}} f_\theta)$ and $\nabla \cdot (p_{\text{data}} \nabla \log p_{\text{data}})$ requires a geometry suited to learning vector fields since f_θ is the vector field we optimize. The most direct option, comparing the two flux divergences as scalar fields, is invariant to p_{data}-preserving dynamics by construction, but moves

one derivative beyond the $L^2(p_{\text{data}})$ vector field geometry of the Fisher divergence, making the objective sensitive to derivative-level artifacts [7]. We therefore seek an objective that matches flux divergences *within* the geometry of the Fisher divergence while remaining invariant to any perturbation v with $\nabla \cdot (p_{\text{data}} v) = 0$. To this end, let $\Pi_{\text{flux}} f$ denote the unique gradient field that satisfies $\nabla \cdot (p_{\text{data}}\, \Pi_{\text{flux}} f_\theta) = \nabla \cdot (p_{\text{data}} f_\theta)$, and note that since $\nabla \log p_{\text{data}}$ is already a gradient field, $\Pi_{\text{flux}}(\nabla \log p_{\text{data}}) = \nabla \log p_{\text{data}}$. We define the *projected Fisher divergence*:

$$\widetilde{\mathcal{J}}(\theta) := \mathbb{E}_{x\sim p_{\text{data}}}\left[\|\Pi_{\text{flux}} f_\theta(x) - \nabla \log p_{\text{data}}(x)\|^2\right]. \tag{4}$$

This objective is invariant to any perturbation v with $\nabla \cdot (p_{\text{data}} v) = 0$ because $\nabla \cdot (p_{\text{data}}(f_\theta + v)) = \nabla \cdot (p_{\text{data}} f_\theta)$, and hence $\Pi_{\text{flux}}(f_\theta + v) = \Pi_{\text{flux}} f_\theta$ by the definition of Π_{flux}. Directly computing Equation (4), however, is intractable in high dimensions.

3.2 Flux Matching Loss

We provide a scalable training objective with the same gradients as the projected Fisher divergence of Equation (4). Let $u_\theta := f_\theta - \nabla \log p_{\text{data}}$ and $r_\theta := p_{\text{data}}^{-1} \nabla \cdot (p_{\text{data}} u_\theta) = \nabla \cdot u_\theta + u_\theta \cdot \nabla \log p_{\text{data}}$. To bridge Equation (4) to our final loss, we show a series of key identities (proven in Section A):

$$\begin{aligned} \widetilde{\mathcal{J}}(\theta) &= \int_0^\infty \mathbb{E}_{x_0\sim p_{\text{data}},\, x_t|x_0}[r_\theta(x_0) r_\theta(x_t)]\; dt && \text{(\textbf{Step 1} via Lemma A.2)} \\ &= -\int_0^\infty \mathbb{E}_{x_0\sim p_{\text{data}},\, x_t|x_0}\left[u_\theta(x_0)^\top \frac{\partial x_t}{\partial x_0}^\top \nabla_{x_t} r_\theta(x_t)\right]\; dt && \text{(\textbf{Step 2} via Lemma A.3).} \end{aligned} \tag{5}$$

Intuitively, r_θ (aka the *Langevin-Stein operator*) is invariant to any p_{data}-preserving dynamics, but applies the local differential operator $\nabla\cdot$, which sits one derivative beyond the Fisher divergence geometry. **Step 1** closes this gap by using diffusion simulations $dx_t = \nabla \log p_{\text{data}}(x_t)\, dt + \sqrt{2}\, dW_t$ to propagate the pointwise value of r_θ across nearby regions of the data distribution. Integrating the autocorrelation $r_\theta(x_0) r_\theta(x_t)$ over time t accumulates these propagated values, effectively "undoing" the $\nabla\cdot$ operator and projecting r_θ back in the Fisher divergence geometry, as illustrated in Figure 3. **Step 2** uses integration by parts to convert the autocorrelation $r_\theta(x_0) r_\theta(x_t)$ into components $u_\theta(x_0)$ and $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t)$. As a final **Step 3**, we apply a stop-gradient on $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t)$. Lemma A.4 shows this preserves the gradient w.r.t. θ (up to a factor of 2) while eliminating the need to backpropagate through the expensive $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t)$ term. Sampling the simulation horizon $t \sim q$ with $t \in [0, \infty)$ gives the resulting *Flux Matching loss*:

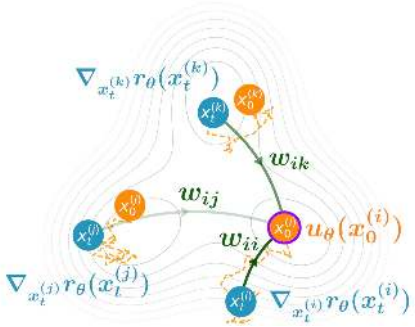


Figure 3: Geometric interpretation of Flux Matching. Colors are detailed in the accompanying Algorithm 1.

$$\mathcal{L}_{\text{flux}}(\theta) := -\mathbb{E}_{\substack{t\sim q \\ x_0\sim p_{\text{data}},\, x_t|x_0}}\left[\frac{1}{q(t)}\, u_\theta(x_0)^\top \operatorname{sg}\left(\frac{\partial x_t}{\partial x_0}^\top \nabla_{x_t} r_\theta(x_t)\right)\right] \quad \blacktriangleright\ \textbf{Flux Matching} \tag{6}$$

where sg denotes stop-gradient and $x_t \mid x_0$ denotes a simulation chain from x_0 to x_t. We formalize the end-to-end connection between the projected Fisher divergence (Equation (4)) and the final Flux Matching loss (Equation (6)) via the following:

Theorem 3.1. *Assume $p_{\text{data}} > 0$ on $\mathbb{R}^d$ and boundary terms in integration-by-parts arguments vanish. Then,*

$$\nabla_\theta \widetilde{\mathcal{J}}(\theta) = 2\, \nabla_\theta \mathcal{L}_{\text{flux}}(\theta).$$

3.3 Estimating the Loss in Practice

Approximating $\nabla \log p_{\text{data}}$. We replace the unknown score $\nabla \log p_{\text{data}}$ with a nonparametric score approximation [31] by building a KDE of the minibatch:

$$\nabla \log \widehat{p}_\sigma(x) = \left(\sum_{i=1}^B \exp\!\left(-\|x - x_i\|^2 / 2\sigma^2\right)(x_i - x)\right) \Big/ \left(\sigma^2 \sum_{i=1}^B \exp\!\left(-\|x - x_i\|^2 / 2\sigma^2\right)\right), \tag{7}$$

Algorithm 1 One Training Iteration of Flux Matching.

x_0 denotes initial samples, $x_0^{(i)}$ highlights the selected initial sample, x_t denotes simulated samples at time t, and w_{im} denotes the weight of simulated sample m on $x_0^{(i)}$. *(Note: Font colors match the visual nodes and edges in Figure 3).*

Input: minibatch $\{x_0^{(i)}\}_{i=1}^B$, bandwidth σ, learnable vector field f_θ

1: Construct $\nabla \log \widehat{p}_\sigma$ via Equation (7) and compute $u_\theta(x_0^{(i)}) = f_\theta(x_0^{(i)}) - \nabla \log \widehat{p}_\sigma(x_0^{(i)})$ for all i
2: Sample shared simulation time $t \sim q$ on $[0, T]$ with $T = 4\sigma^2$
3: MCMC $\{x_0^{(i)}\}_{i=1}^B$ to $\{x_t^{(i)}\}_{i=1}^B$ with $\nabla \log \widehat{p}_\sigma$ using Equation (8)
4: For selected initial sample $x_0^{(i)}$, calculate Equation (9): $\widehat{\partial x_t / \partial x_0^\top} \nabla r_\theta(x_0^{(i)}, t) := \sum_{m=1}^B w_{im} \nabla_{x_t^{(m)}} r_\theta(x_t^{(m)})$
5: Apply Step 4 for every initial sample $i = 1, \ldots, B$
6: Form $\mathcal{L}_{\text{flux}}$ from Equation (6) and update θ

which is asymptotically unbiased in the usual large-batch, vanishing-bandwidth regime and is a common technique used in modern generative models [65, 57, 20, 37].

Simulating x_t from x_0. We sample the simulation horizon $t \sim q$ where q is supported on $[0, \infty)$. Simulating arbitrarily large horizons is computationally infeasible, however, so we truncate the support of q to $[0, T]$. In practice, we find that defining q to be either a truncated uniform or exponential and setting $T = 4\sigma^2$ is sufficient (justification is provided in Section D.1.1). Given t, we run 4 MCMC steps with step size $h = \frac{1}{4}t$ starting from x_0. To enable stable large-step sampling, we use an exponentially integrated Langevin update:

$$x_{k+1} = \mu_k + e^{-h}(x_k - \mu_k) + \sigma\sqrt{1 - e^{-2h}}\xi_k, \quad \mu_k = x_k + \sigma^2 \nabla \log p_\sigma(x_k), \quad \xi_k \sim \mathcal{N}(0, I). \tag{8}$$

Note that the computational cost of calculating Equation (8) many times is *negligible* compared to running f_θ once since $\nabla \log p_\sigma$ is a closed-form KDE approximation.

Estimating $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t)$. One could backpropagate through each simulated chain to obtain the pathwise term $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t) = \nabla_{x_0} r_\theta(x_t)$. However, this term appears inside a conditional expectation over $x_t \mid x_0$ (aka an expectation over simulation paths from x_0). Therefore, for each initial point $x_0^{(i)}$ and time t, we can reduce variance by estimating the conditional expectation using all simulated chains $\{x_t^{(j)}\}_{j=1}^B$ from the current minibatch (particularly helpful is specific regimes of σ, which we detail in Section D), rather than only using a single chain $x_t^{(i)}$ generated from $x_0^{(i)}$.

Unlike the same chain sensitivity $\partial x_t^{(i)} / \partial x_0^{(i)}$, the cross-chain sensitivity $\partial x_t^{(j)} / \partial x_0^{(i)}$ where $i \neq j$ is unavailable, so we approximate it with Gaussian transition weights w_{ij} normalized over minibatch endpoints j (further details in Section D.1.2). Specifically, for each $x_0^{(i)}$, we replace $\partial x_t / \partial x_0^\top \nabla_{x_t} r_\theta(x_t)$ with the variance-reduced estimator:

$$\widehat{(\partial x_t / \partial x_0)^\top} \nabla r_\theta(x_0^{(i)}, t) := \sum_{j=1}^B w_{ij}(t) \nabla_{x_t^{(j)}} r_\theta\left(x_t^{(j)}\right). \tag{9}$$

Finally, the divergence term in $r_\theta = \nabla \cdot u_\theta + u_\theta \cdot \nabla \log p$ is approximated with a single-sample Hutchinson trace estimator [29].

3.4 Extension to Noise Annealed Generative Fields

Rather than learning a single vector field at the data distribution, diffusion models [59, 58, 40, 25, 56] learn score fields over a continuum of noise annealed distributions $\{p_\sigma\}_{\sigma \sim \mathcal{P}}$ where $\mathcal{P}$ denotes the sampling distribution for noise levels used during training. Here, $p_\sigma = p_{\text{data}} * \mathcal{N}(0, \sigma^2 I)$. Flux Matching extends to this setting by applying the same objective independently at each noise level. Let $f_\theta^\sigma(x) := f_\theta(x, \sigma)$, and let q_σ denote the σ-dependent importance sampler over simulation horizons (can be learned via Section D.1.1). We write

$$\mathcal{L}_{\text{flux}}^\sigma(\theta) := \mathcal{L}_{\text{flux}}\left(\theta;\, p_\sigma,\, q_\sigma,\, f_\theta^\sigma\right), \tag{10}$$

where the right-hand side denotes Equation (6) with p replaced by p_σ, f_θ replaced by f_θ^σ, q replaced by q_σ, and all noise-dependent quantities evaluated at the same σ (e.g., the truncation horizon $T = 4\sigma^2$). The noise annealed Flux Matching objective is then

$$\mathcal{L}_{\text{flux-noise}}(\theta, \eta) := \mathbb{E}_{\sigma \sim \mathcal{P}}\left[\mathcal{L}_{\text{flux}}^\sigma(\theta) / \exp(s_\eta(\sigma)) + s_\eta(\sigma)\right]. \tag{11}$$

$s_\eta(\sigma)$ is a learned normalizer (single-layer MLP) that reweighs losses from different noise levels to be on comparable scales [34] and is simultaneously trained with the main network.

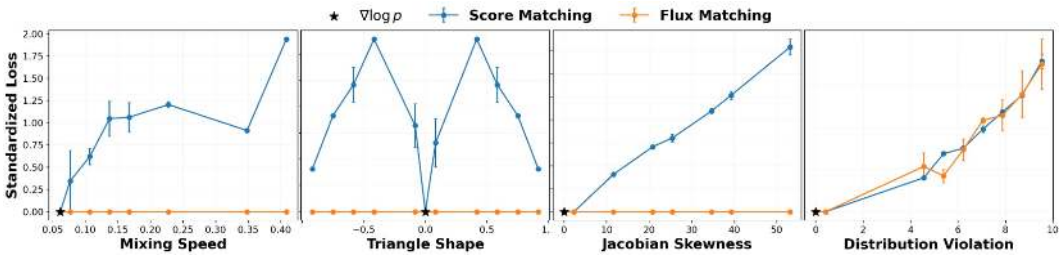


Figure 4: Normalized score matching and Flux Matching losses as we vary properties of the vector field on a Gaussian mixture. The black star denotes the attribute value of the score field $\nabla \log p_{\text{data}}$. The first three panels vary distribution preserving fields with different values of a chosen attribute; the last panel varies fields that violate the target stationary distribution.

3.5 Sampling & Likelihood Computation

Training and sampling are *fully decoupled*. Although f_θ is learned through Flux Matching, at sampling time it can replace the score term in standard score-based samplers (e.g. unadjusted Langevin dynamics) with no algorithmic changes and additional cost. Similarly, models learned via noise annealed Flux Matching can be used with reverse diffusion and probability-flow ODE samplers by simply replacing the noise conditioned score with f_θ^σ. For probability-flow ODE sampling, likelihoods can also be computed with the usual instantaneous change-of-variables formula [8]. See Proposition A.1 for a formal statement.

3.6 Learning Useful Generative Vector Fields

Flux Matching learns a family of vector fields that generate the same target distribution. The remaining task is to choose an element of this family with properties useful for the application. We provide two general strategies:

(1) Application Specific Loss. Augment the Flux Matching objective with a loss L_{app} that encourages the desired properties, e.g., $\mathcal{L}_{\text{flux}} + \sum_i \lambda_{\text{app},i}\mathcal{L}_{\text{app},i}$. For example, adding $\lambda_{\text{L2}}\|f_\theta\|^2_{L^2(p)}$ recovers the score function when minimized. In the noise annealed setting, nonnegative application losses can be normalized across noise levels using an equivalent learned normalizer as Equation (11). For signed losses, we instead can normalize within discrete σ-buckets using running statistics, as in [10].

(2) Model Parameterization. Desired attributes can also be built directly into the architecture used to represent f_θ. For example, an attention mask in a transformer can enforce directed relationships among variables [61].

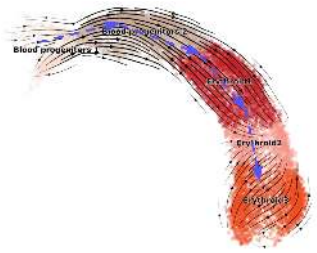
Section 4 instantiates these two strategies in different settings, showing how Flux Matching can select useful generative fields while preserving the same target distribution.

4 Applications of Flux Matching

We evaluate Flux Matching across five settings that highlight unique applications of the method. The first two experiments use the Flux Matching loss from Equation (6): Section 4.1 isolates the main controllability benefit of Flux Matching on a toy distribution, and Section 4.2 shows that Flux Matching can fit biologically interpretable vector fields. The remaining three experiments use the noise annealed objective from Equation (11): Section 4.3 tests Flux Matching as a standalone image generation objective, Section 4.4 leverages Flux Matching to optimize fields for faster sampling, and Section 4.5 uses Flux Matching to impose directed structure between variables.

4.1 *Controllable* Generative Fields

One advantage of Flux Matching is that it exposes distribution-preserving degrees of freedom for control. Many vector fields share the same stationary distribution, and their differences govern properties of the generative process such as mixing rate, circulation pattern, and reversibility. Score matching targets $\nabla \log p_{\text{data}}$ as the unique correct field and penalizes any deviation, even ones that



Dataset	Flux Matching CBC ↑ / Consist. ↑	scVelo [5] CBC ↑ / Consist. ↑
Pancreas	0.202 / **0.972**	**0.330** / 0.821
Gastrulation	**0.611** / **0.991**	-0.639 / 0.877
Dentategyrus	**0.284** / **0.981**	-0.084 / 0.791
Bone Marrow	**0.177** / **0.939**	-0.789 / 0.857
Hindbrain	**0.345** / **0.897**	0.332 / 0.874

Figure 5: **(Left)** Learned RNA velocity using Flux Matching; blue arrows indicate ground-truth biological progression between cell-types. **(Right)** CBC and consistency means across datasets.

leave the distribution unchanged. Flux Matching instead treats the entire distribution preserving family as equivalent.

Setup. On a 2D three-component Gaussian mixture, we construct three one-parameter families of distribution preserving perturbations of the score field, indexed by attributes we call *mixing speed*, *triangle shape*, and *Jacobian skewness*. For each perturbed field we compute the score matching and Flux Matching losses, alongside a distribution-violation metric that is zero exactly on the distribution preserving family. Since the two objectives have different raw scales, we report standardized losses. Full definitions and construction details appear in Section B.1.

Results. Figure 4 displays the outcomes. In the first three panels, increasing the perturbation magnitude drives the score matching loss up immediately while the Flux Matching loss stays at exactly zero. Practitioners can therefore tune these degrees of freedom to shape the dynamics, for example to increase mixing, enforce triangular circulation, or induce nonreversible structure, without changing the target density. The fourth panel perturbs the field outside the distribution preserving family, and the Flux Matching loss now rises with the degree of distribution violation. Flux Matching is not flat everywhere. It is flat precisely on the family of vector fields sharing the target stationary distribution.

4.2 *Interpretable* Generative Fields

A second practical advantage of Flux Matching is that the vector field f_θ may be of any parametric family, including ones whose parameters carry scientific meaning. This enables *interpretable* generative dynamics. Rather than learning a black-box neural field, we constrain f_θ to a structured form chosen by domain experts and fit its parameters directly from data. We illustrate this on RNA velocity [36], a problem in single-cell biology where the admissible vector fields are prescribed by a known mechanistic model.

RNA velocity. From a single static snapshot of a cell population, RNA velocity aims to infer the direction in which each cell is moving through gene expression space. For each of G genes, two quantities are measured per cell, namely an immature transcript u_g and its mature form s_g. A standard biophysical model [5] prescribes the ordinary differential equation (ODE)

$$\frac{d}{d\tau}\begin{pmatrix} u_g \\ s_g \end{pmatrix} = \begin{pmatrix} \alpha_g(\tau) - \beta_g\, u_g \\ \beta_g\, u_g - \gamma_g\, s_g \end{pmatrix}, \tag{12}$$

where the rates $\alpha_g, \beta_g, \gamma_g$ are biologically meaningful (transcription, splicing, and degradation, respectively). A dominant method, scVelo [5], fits these rates per gene using an EM-style latent-variable procedure that is known to be sensitive to initialization.

Flux Matching as a drop-in trainer. We keep the biophysical model (12) unchanged but replace the bespoke EM fit with gradient descent on $\mathcal{L}_{\text{flux}}$. Concretely, we concatenate the per-gene fields across $G = 2000$ genes into a full cell-state vector field and optimize the scalar parameters jointly. The structured ODE restricts the admissible vector fields, while Flux Matching supplies the training objective.

Results. Figure 5 reports two standard RNA velocity metrics. *Cross-boundary correctness* (CBC) measures how well predicted velocities align with known transitions between cell types, and *consistency* measures whether nearby cells receive similar velocity directions. Across five real single cell datasets, Flux Matching improves consistency on all five and CBC on four out of five, under the same parametric family as scVelo. Because the model class is unchanged, the gains are attributable to the

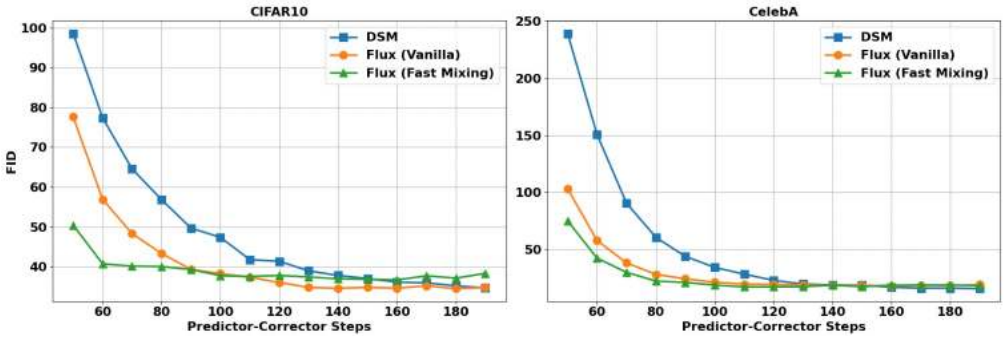


Figure 6: FID (calculated with 1K generated samples) as a function of the number of sampling steps.

fitting procedure rather than to added expressivity. We foresee that Flux Matching can be applied to other newly developed RNA velocity models with more sophisticated biological parameterizations.

4.3 *Unrestricted* Generative Fields

Flux Matching's main value is in imposing structure on the learned vector field, but the Flux Matching loss is also viable as a standalone training objective on complex high-dimensional distributions. We verify this in the *unrestricted* (vanilla) setting, where no additional field property is optimized.

Setup. We evaluate on CIFAR10 ($3 \times 32 \times 32$) and CelebA ($3 \times 64 \times 64$). We train a standard UNet architecture from [19] with the noise annealed Flux Matching objective in Equation (11), using the EDM noise level distribution [33], for $500{,}000$ steps. As a baseline, we train noise annealed denoising score matching (DSM) with the same architecture and hyperparameters.

Results. The top half of Table 1 shows that Flux Matching performs strongly on both datasets, demonstrating that the loss alone scales to realistic high-dimensional image distributions. The remaining FID gap to DSM is unsurprising since DSM has benefited from many engineering iterations specifically aimed at optimizing FID, whereas Flux Matching is evaluated here as a first-pass implementation of a new learning objective. The bottom half shows that Flux Matching is roughly 3–4$\times$ slower than DSM during training and uses about 2–3$\times$ more memory. This overhead is incurred only during training. At sampling time, the learned field can be used in the same samplers as a score model. As noted earlier, the main motivation for Flux Matching is not to replace DSM in this unrestricted setting but to enable dynamics with useful properties, as shown in the next two experiments.

Table 1: Unconditional generation performance and training-time efficiency.

Dataset	Model	Performance		
		FID (↓)	IS (↑)	NLL (bpd, ↓)
CIFAR10	DSM	4.74	8.52	3.16
	Flux	9.06	8.54	3.26
CelebA	DSM	2.41	-	2.03
	Flux	7.07	-	2.17
Dataset	**Model**	**Efficiency**		
		Speed (it/s)		**Memory/GPU (G)**
CIFAR10	DSM	11.63		2.79
	Flux	4.01		5.69
CelebA	DSM	7.20		7.79
	Flux	1.77		22.67

4.4 *Fast Mixing* Generative Fields for Accelerated Sampling

Fast mixing fields can accelerate sampling by requiring fewer sampling steps to converge to the target distribution. Score-based Langevin dynamics is reversible and known to mix slowly [30, 53, 16], so score matching cannot exploit this property while Flux Matching can. Since mixing time itself is intractable to optimize directly, we minimize a proxy defined in Section B.4.

Setup. We reuse the training and model setup of Section 4.3 but add the mixing proxy with weight $\lambda_{\text{mixing}} = 0.01$, giving $\mathcal{L}_{\text{flux}-\text{fast}} = \mathcal{L}_{\text{flux-noise}} + \lambda_{\text{mixing}} \mathcal{L}_{\text{mixing}}$. After training f_θ^σ with $\mathcal{L}_{\text{flux}-\text{fast}}$ on CIFAR10 and CelebA, we evaluate FID on 1K generated samples across different numbers of sampling steps and compare against vanilla Flux Matching and DSM (noise annealed versions).

Results. As shown in Figure 6, fast-mixing Flux Matching reaches a reasonable FID with substantially fewer sampling steps than vanilla Flux Matching and DSM on CIFAR10; on CelebA, the gain

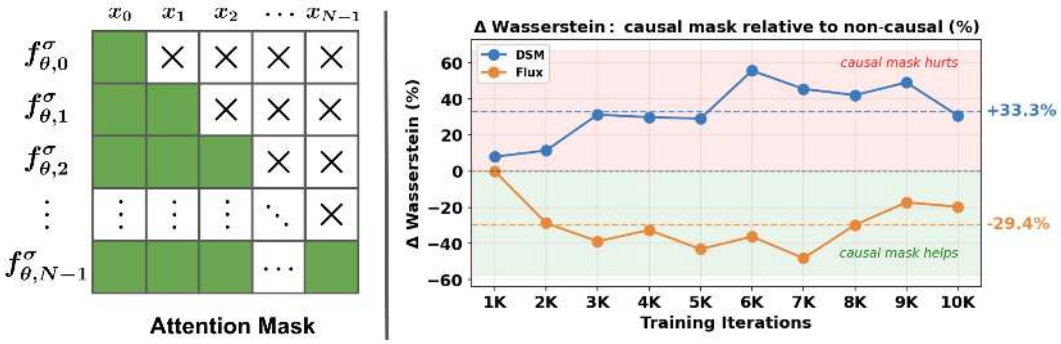


Figure 7: **(Left)** Causal attention mask used for trajectory generation. Rows index the output at each trajectory time, and columns index the input states at each trajectory time. The upper-triangular mask allows $f^{\sigma}_{\theta,n}$ to depend only on states x_m with $m \leq n$, enforcing autoregressive structure while still evaluating all outputs in parallel. **(Right)** Relative change in Wasserstein distance from adding a causal attention mask to $f^{\sigma}_{\theta,n}$ when training with DSM versus Flux Matching.

over vanilla Flux Matching is modest. Interestingly, Flux Matching without fast mixing reaches a reasonable FID with fewer sampling steps than DSM on both datasets, suggesting that Flux Matching may already learn fields whose sampling dynamics are easier to mix.

4.5 *Embedding Structure* in Generative Fields

Many generative problems have known structure among variables, and Flux Matching lets us encode this structure directly into the architecture representing the vector field. Score fields are gradient fields with symmetric Jacobians (by equality of mixed partials), so directed dependencies such as temporal autoregression are incompatible. Flux Matching has no such constraint.

Setup. We simulate trajectories of two masses connected by nonlinear springs (ODE in Section B.5), with simulator state $x(\tau) = (q_1(\tau), v_1(\tau), q_2(\tau), v_2(\tau)) \in \mathbb{R}^4$ giving the position q_i and velocity v_i of each mass at physical time τ. Each data sample is a discretized trajectory $X = (x_0, \ldots, x_{N-1}) \in \mathbb{R}^{N\times 4}$ with $x_n := x(n\Delta\tau)$ and $N = 50$. Our goal is to model the distribution over full trajectories. Temporal order provides a natural inductive bias that later states depend only on earlier states, which shrinks the hypothesis class and improves data efficiency [4]. Standard diffusion samplers generate all time points in parallel, but Flux Matching additionally lets us impose a causal mask on f^{σ}_{θ}, retaining the autoregressive inductive bias *without sequential sampling*. We train noise annealed DSM and Flux Matching, each with and without a causal mask (left of Figure 7), on 2000 simulated trajectories. All four models share the same attention architecture (Section B.5). We evaluate via the empirical Wasserstein distance $\mathcal{W}_2$ to the training distribution.

Results. The right side of Figure 7 reports the relative change in $\mathcal{W}_2$ from adding the causal mask. The mask consistently improves Flux Matching, confirming that the autoregressive inductive bias helps. The same mask worsens DSM, as expected since DSM forces the learned field to approximate a conservative score field whose symmetric Jacobian conflicts with directed temporal dependence.

5 Conclusion

In this paper, we presented *Flux Matching*, a new generative modeling paradigm that generalizes score matching to learn any vector field that generates samples from the target distribution. We proposed a scalable learning objective, the Flux Matching loss, together with a noise annealed extension whose learned models can be used out of the box with existing diffusion samplers and likelihood computations. We showed that Flux Matching performs well across a range of applications, including complex, high-dimensional image distributions. Most importantly, these applications demonstrate the flexibility Flux Matching gives practitioners to enforce and optimize attributes of the vector field itself. We showed that this flexibility enables faster samplers, more interpretable models, and generative models with prescribed relationships between variables.

Acknowledgments and Disclosure of Funding

We thank Eric Ma, Alex Belov, Meihua Dang, Gabe Guo, Jiaqi Han, and Haotian Ye for helpful feedback and discussions. This work was supported by CZ Biohub, ONR Grant N00014-23-1-2159, the Laude Institute Moonshot Seed Grant, the Pantas And Ting Sutardja Foundation, the Wu Tsai Neurosciences Institute Big Ideas in Neuroscience Program, NIH DP2 grant 1DP2OD037052-01, and NIH K99/R00 grant 4K99HG012887-02. PPH acknowledges support from the NSF Graduate Research Fellowship.

References

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[2] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

[3] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.

[4] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[5] V. Bergen, M. Lange, S. Peidli, F. A. Wolf, and F. J. Theis. Generalizing rna velocity to transient cell states through dynamical modeling. *Nature biotechnology*, 38(12):1408–1414, 2020.

[6] F. Bleile, S. Lumpp, and M. Drton. Efficient learning of stationary diffusions with stein-type discrepancies. *arXiv preprint arXiv:2601.16597*, 2026.

[7] R. Chartrand. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011(1):164564, 2011.

[8] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[10] K. Choi, C. Meng, Y. Song, and S. Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pages 2552–2573. PMLR, 2022.

[11] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*, 2022.

[12] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[13] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[14] A. Dixit, O. Parnas, B. Li, J. Chen, C. P. Fulco, L. Jerby-Arnon, N. D. Marjanovic, D. Dionne, T. Burks, R. Raychowdhury, et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *cell*, 167(7):1853–1866, 2016.

[15] Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019.

[16] A. B. Duncan, T. Lelievre, and G. A. Pavliotis. Variance reduction using nonreversible langevin samplers. *Journal of statistical physics*, 163(3):457–491, 2016.

[17] A. B. Duncan, G. A. Pavliotis, and K. Zygalakis. Nonreversible langevin samplers: Splitting schemes, analysis and implementation. *arXiv preprint arXiv:1701.04247*, 2017.

[18] N. Fishman, L. Klarner, E. Mathieu, M. Hutchinson, and V. De Bortoli. Metropolis sampling for constrained diffusion models. *Advances in Neural Information Processing Systems*, 36:62296–62331, 2023.

[19] FutureXiang. Diffusion: Minimal multi-gpu implementation of diffusion models with classifier-free guidance (cfg). `https://github.com/FutureXiang/Diffusion/tree/master`, 2023.

[20] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[21] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

[22] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[23] N. Hansen and A. Sokol. Causal interpretation of stochastic differential equations. 2014.

[24] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[25] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[26] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[27] C. Horvat and J.-P. Pfister. On gauge freedom, conservativity and intrinsic dimensionality estimation in diffusion models. *arXiv preprint arXiv:2402.03845*, 2024.

[28] Y. Huang, T. Transue, S.-H. Wang, W. Feldman, H. Zhang, and B. Wang. Improving flow matching by aligning flow divergence. *arXiv preprint arXiv:2602.00869*, 2026.

[29] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[30] C.-R. Hwang, S.-Y. Hwang-Ma, and S.-J. Sheu. Accelerating diffusions. 2005.

[31] A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[32] B. Jing, G. Corso, J. Chang, R. Barzilay, and T. Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in neural information processing systems*, 35:24240–24253, 2022.

[33] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[34] T. Karras, M. Aittala, J. Lehtinen, J. Hellsten, T. Aila, and S. Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24174–24184, 2024.

[35] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[36] G. La Manno, R. Soldatov, A. Zeisel, E. Braun, H. Hochgerner, V. Petukhov, K. Lidschreiber, M. E. Kastriti, P. Lönnerberg, A. Furlan, et al. Rna velocity of single cells. *Nature*, 560(7719):494–498, 2018.

[37] C.-H. Lai, B. Nguyen, N. Murata, Y. Takida, T. Uesaka, Y. Mitsufuji, S. Ermon, and M. Tao. A unified view of drifting and score-based models. *arXiv preprint arXiv:2603.07514*, 2026.

[38] C.-H. Lai, Y. Takida, N. Murata, T. Uesaka, Y. Mitsufuji, and S. Ermon. Fp-diffusion: Improving score-based diffusion models by enforcing the underlying score fokker-planck equation. In *International Conference on Machine Learning*, pages 18365–18398. PMLR, 2023.

[39] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[40] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[41] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[42] L. Lorch, A. Krause, and B. Schölkopf. Causal modeling with stationary diffusions. In *International Conference on Artificial Intelligence and Statistics*, pages 1927–1935. PMLR, 2024.

[43] A. Lou and S. Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pages 22675–22701. PMLR, 2023.

[44] Y.-A. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28, 2015.

[45] K. Neklyudov, R. Brekelmans, D. Severo, and A. Makhzani. Action matching: Learning stochastic dynamics from samples. In *International conference on machine learning*, pages 25858–25889. PMLR, 2023.

[46] K. Neklyudov, R. Brekelmans, A. Tong, L. Atanackovic, Q. Liu, and A. Makhzani. A computational framework for solving wasserstein lagrangian flows. *arXiv preprint arXiv:2310.10649*, 2023.

[47] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[48] G. A. Pavliotis. Stochastic processes and applications. *Texts in applied mathematics*, 60:41–43, 2014.

[49] G. A. Pavliotis and A. M. Stuart. Multiscale methods, volume 53 of texts in applied mathematics, 2008.

[50] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[51] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[52] K. Petrović, L. Atanackovic, V. Moro, K. Kapuśniak, I. I. Ceylan, M. Bronstein, A. J. Bose, and A. Tong. Curly flow matching for learning non-gradient field dynamics. *arXiv preprint arXiv:2510.26645*, 2025.

[53] L. Rey-Bellet and K. Spiliopoulos. Irreversible langevin samplers and variance reduction: a large deviations approach. *Nonlinearity*, 28(7):2081–2103, 2015.

[54] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[55] P. K. Rubenstein, S. Bongers, B. Schölkopf, and J. M. Mooij. From deterministic odes to dynamic structural causal models. *arXiv preprint arXiv:1608.08028*, 2016.

[56] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[57] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. 2023.

[58] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[59] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[60] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[62] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[63] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[64] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

[65] Y. Xu, S. Tong, and T. Jaakkola. Stable target field for reduced variance score estimation in diffusion models. *arXiv preprint arXiv:2302.00670*, 2023.

[66] Y. Zhang and M. Levin. Equilibrium flow: from snapshots to dynamics. *arXiv preprint arXiv:2509.17990*, 2025.

Appendix Overview

A Proofs

Proposition 2.1. *[Classical stationary Fokker–Planck characterization, Section 2.4 of [48]]* p_{data} *is a stationary distribution of the diffusion* $dx_t = f_\theta(x_t)\,dt + \sqrt{2}\,dW_t$ *iff*

$$\nabla \cdot \big(p_{\mathrm{data}}(x) f_\theta(x)\big) - \nabla \cdot \big(p_{\mathrm{data}}(x) \nabla \log p_{\mathrm{data}}(x)\big) = 0 \qquad \textit{for all } x.$$

Proof. Follows directly from Equation 2.37 of [48]. □

Proposition A.1. *Assume* $p_t > 0$ *on* $\mathbb{R}^d$*. If vector field* f_t *satisfies the Flux Matching condition given by Proposition 2.1 that*

$$\nabla \cdot (f_t(x) p_t(x)) = \nabla \cdot (\nabla \log p_t(x) p_t(x)) \tag{13}$$

for all t *and* x*, then replacing* $\nabla \log p_t$ *by* f_t *in the sampler leaves the continuous-time marginal density evolution unchanged.*

Proof. The Fokker–Planck contribution of replacing $\nabla \log p_t$ by f_t differs from the score-based one by

$$-\Big(\nabla \cdot (f_t(x) p_t(x)) - \nabla \cdot (\nabla \log p_t(x) p_t(x))\Big),$$

which is zero by (13). Hence the same p_t solves the same marginal evolution. □

Lemma A.2. *Let*

$$g_\theta := \Pi_{\mathrm{flux}} f_\theta - \nabla \log p_{\mathrm{data}}, \qquad u_\theta := f_\theta - \nabla \log p_{\mathrm{data}}, \qquad r_\theta := \frac{1}{p_{\mathrm{data}}} \nabla \cdot (p_{\mathrm{data}}\, u_\theta).$$

Since $\Pi_{\mathrm{flux}} f_\theta$ *and* $\nabla \log p_{\mathrm{data}}$ *are gradient fields,* $g_\theta = \nabla \phi_\theta$ *for some potential* ϕ_θ*, unique up to an additive constant. Assume* $\phi_\theta, r_\theta \in L^2_0(p_{\mathrm{data}})$ *and vanishing boundary terms. Then,*

$$\mathbb{E}_{x \sim p_{\mathrm{data}}}\big[\|g_\theta(x)\|^2\big] = \int_0^\infty \mathbb{E}_{x_0 \sim p_{\mathrm{data}}, x_t | x_0}\big[r_\theta(x_0)\, r_\theta(x_t)\big]\, dt,$$

where $(x_t)_{t \geq 0}$ *is the stationary Langevin diffusion with generator*

$$L = \Delta + \nabla \log p_{\mathrm{data}} \cdot \nabla.$$

Proof. Since both $\Pi_{\mathrm{flux}} f_\theta$ and $\nabla \log p_{\mathrm{data}}$ are gradient fields, so is

$$g_\theta = \Pi_{\mathrm{flux}} f_\theta - \nabla \log p_{\mathrm{data}}.$$

Thus $g_\theta = \nabla \phi_\theta$ for some ϕ_θ, unique up to an additive constant, which we fix by requiring $\phi_\theta \in L^2_0(p_{\mathrm{data}})$. Moreover, by definition of Π_{flux},

$$\nabla \cdot \big(p_{\mathrm{data}}\, \Pi_{\mathrm{flux}} f_\theta\big) = \nabla \cdot (p_{\mathrm{data}} f_\theta).$$

Hence

$$\begin{aligned}
r_\theta &= \frac{1}{p_{\mathrm{data}}} \nabla \cdot (p_{\mathrm{data}}\, u_\theta) \\
&= \frac{1}{p_{\mathrm{data}}} \nabla \cdot \big(p_{\mathrm{data}} (f_\theta - \nabla \log p_{\mathrm{data}})\big) \\
&= \frac{1}{p_{\mathrm{data}}} \nabla \cdot \big(p_{\mathrm{data}} (\Pi_{\mathrm{flux}} f_\theta - \nabla \log p_{\mathrm{data}})\big) \\
&= \frac{1}{p_{\mathrm{data}}} \nabla \cdot (p_{\mathrm{data}}\, g_\theta) \\
&= \nabla \cdot g_\theta + g_\theta \cdot \nabla \log p_{\mathrm{data}}.
\end{aligned}$$

Therefore r_θ is equivalently the Langevin Stein operator applied to g_θ, and since $g_\theta = \nabla\phi_\theta$,

$$r_\theta = \frac{1}{p_{\text{data}}}\nabla\cdot(p_{\text{data}}\nabla\phi_\theta) = \Delta\phi_\theta + \nabla\log p_{\text{data}}\cdot\nabla\phi_\theta = L\phi_\theta.$$

By integration by parts,

$$\begin{aligned}\mathbb{E}_{x\sim p_{\text{data}}}\big[\|g_\theta(x)\|^2\big] &= \int \|\nabla\phi_\theta(x)\|^2 p_{\text{data}}(x)\,dx \\ &= -\int \phi_\theta(x)\,L\phi_\theta(x)\,p_{\text{data}}(x)\,dx \\ &= -\int \phi_\theta(x)\,r_\theta(x)\,p_{\text{data}}(x)\,dx.\end{aligned}$$

Since $r_\theta = L\phi_\theta$, we have $\phi_\theta = -(-L)^{-1}r_\theta$, and thus

$$\mathbb{E}_{x\sim p_{\text{data}}}\big[\|g_\theta(x)\|^2\big] = \int r_\theta(x)\,(-L)^{-1}r_\theta(x)\,p_{\text{data}}(x)\,dx.$$

We use the identity $(-L)^{-1} = \int_0^\infty e^{Lt}dt$ (Ch. 11 of [49]),

$$\int r_\theta(x)\,(-L)^{-1}r_\theta(x)\,p_{\text{data}}(x)\,dx = \int_0^\infty \int r_\theta(x)\,(e^{tL}r_\theta)(x)\,p_{\text{data}}(x)\,dx\,dt.$$

Finally, for the Langevin diffusion with generator L,

$$(e^{tL}r_\theta)(x) = \mathbb{E}[r_\theta(x_t)\mid x_0 = x]\,,$$

where the expectation is over the Langevin diffusion path $(x_s)_{s\geq 0}$ started from $x_0 = x$ (see Ch. 2 of [48]). Therefore, since $x_0\sim p_{\text{data}}$,

$$\begin{aligned}\int r_\theta(x)\,(e^{tL}r_\theta)(x)\,p_{\text{data}}(x)\,dx &= \mathbb{E}_{x_0\sim p_{\text{data}}}[r_\theta(x_0)\,\mathbb{E}[r_\theta(x_t)\mid x_0]] \\ &= \mathbb{E}_{x_0\sim p_{\text{data}}}[\mathbb{E}[r_\theta(x_0)r_\theta(x_t)\mid x_0]] \\ &= \mathbb{E}_{x_0\sim p_{\text{data}},x_t|x_0}[r_\theta(x_0)r_\theta(x_t)]\,.\end{aligned}$$

Hence,

$$\int r_\theta(x)\,(-L)^{-1}r_\theta(x)\,p_{\text{data}}(x)\,dx = \int_0^\infty \mathbb{E}_{x_0\sim p_{\text{data}},x_t|x_0}[r_\theta(x_0)r_\theta(x_t)]\;dt.$$

□

Lemma A.3. *Let*

$$u_\theta := f_\theta - \nabla\log p_{\text{data}}, \qquad r_\theta := \frac{1}{p_{\text{data}}}\nabla\cdot(p_{\text{data}}\,u_\theta).$$

Assume u_θ and $e^{tL}r_\theta$ are sufficiently regular and vanishing boundary terms hold. Then, for every $t\geq 0$,

$$\mathbb{E}_{x_0\sim p_{\text{data}},x_t|x_0}[r_\theta(x_0)\,r_\theta(x_t)] = -\mathbb{E}_{x_0\sim p_{\text{data}}}\big[u_\theta(x_0)^\top\nabla(e^{tL}r_\theta)(x_0)\big]\,,$$

where $(x_t)_{t\geq 0}$ is the stationary Langevin diffusion with generator

$$L = \Delta + \nabla\log p_{\text{data}}\cdot\nabla,$$

and $\partial x_t/\partial x_0$ denotes the Jacobian of the associated Langevin path. Equivalently,

$$\mathbb{E}_{x_0\sim p_{\text{data}},x_t|x_0}[r_\theta(x_0)\,r_\theta(x_t)] = -\mathbb{E}_{x_0\sim p_{\text{data}},x_t|x_0}\big[u_\theta(x_0)^\top\partial x_t/\partial x_0^\top\nabla_{x_t}r_\theta(x_t)\big]\,.$$

Proof. For the Langevin diffusion with generator L,

$$(e^{tL}r_\theta)(x) = \mathbb{E}[r_\theta(x_t)\mid x_0 = x]\,,$$

where the expectation is over diffusion paths $(x_s)_{s\geq 0}$ started from $x_0 = x$ (Ch. 2 of [48]). Hence,

$$\begin{aligned}\mathbb{E}_{x_0\sim p_{\mathrm{data}},x_t|x_0}[r_\theta(x_0)\, r_\theta(x_t)] &= \mathbb{E}_{x_0\sim p_{\mathrm{data}}}[r_\theta(x_0)\, \mathbb{E}[r_\theta(x_t) \mid x_0]] \\ &= \mathbb{E}_{x_0\sim p_{\mathrm{data}}}\big[r_\theta(x_0)\, (e^{tL} r_\theta)(x_0)\big]\,.\end{aligned}$$

Using $r_\theta = p_{\mathrm{data}}^{-1} \nabla \cdot (p_{\mathrm{data}}\, u_\theta)$, we have

$$\begin{aligned}\mathbb{E}_{x_0\sim p_{\mathrm{data}}}\big[r_\theta(x_0)\, (e^{tL} r_\theta)(x_0)\big] &= \int \nabla\cdot\big(p_{\mathrm{data}}(x) u_\theta(x)\big)\, (e^{tL} r_\theta)(x)\, dx \\ &= -\int p_{\mathrm{data}}(x)\, u_\theta(x)^\top \nabla (e^{tL} r_\theta)(x)\, dx \qquad \text{(integration by parts)} \\ &= -\mathbb{E}_{x_0\sim p_{\mathrm{data}}}\big[u_\theta(x_0)^\top \nabla (e^{tL} r_\theta)(x_0)\big]\,.\end{aligned}$$

Finally, let $\partial x_t/\partial x_0$ denote the Jacobian of the path generated by the Langevin diffusion. Since

$$\nabla (e^{tL} r_\theta)(x_0) = \nabla_{x_0} \mathbb{E}[r_\theta(x_t) \mid x_0] = \mathbb{E}\big[\partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t) \mid x_0\big]\,,$$

we obtain

$$\mathbb{E}_{x_0\sim p_{\mathrm{data}},x_t|x_0}[r_\theta(x_0)\, r_\theta(x_t)] = -\mathbb{E}_{x_0\sim p_{\mathrm{data}},x_t|x_0}\big[u_\theta(x_0)^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big]\,.$$

□

Lemma A.4. *Let q be any strictly positive probability density on $[0,\infty)$, let $\partial x_t/\partial x_0$, and define*

$$\widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) := -\mathbb{E}_{\substack{t\sim q \\ x_0\sim p_{\mathrm{data}},\, x_t|x_0}} \big[q(t)^{-1}\, u_\theta(x_0)^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big]\,,$$

$$\mathcal{L}_{\mathrm{Flux}}(\theta) := -\mathbb{E}_{\substack{t\sim q \\ x_0\sim p_{\mathrm{data}},\, x_t|x_0}} \big[q(t)^{-1}\, u_\theta(x_0)^\top \,\mathrm{sg}\big(\partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big)\big]\,.$$

Assume differentiation and expectation may be interchanged, and that the stationary Langevin diffusion is reversible with respect to p_{data}. Then,

$$\nabla_\theta \widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) = 2\, \nabla_\theta \mathcal{L}_{\mathrm{Flux}}(\theta).$$

Proof. We differentiate componentwise in θ. For any parameter θ_j,

$$\begin{aligned}\partial_{\theta_j} \widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) = &-\mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} u_\theta(x_0))^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big] \\ &- \mathbb{E}\big[q(t)^{-1}\, u_\theta(x_0)^\top \partial x_t/\partial x_0^\top \nabla_{x_t} (\partial_{\theta_j} r_\theta(x_t))\big]\,.\end{aligned}$$

Since $r_\theta = \frac{1}{p_{\mathrm{data}}} \nabla\cdot(p_{\mathrm{data}}\, u_\theta)$ depends linearly on u_θ, we have

$$\partial_{\theta_j} r_\theta = \frac{1}{p_{\mathrm{data}}} \nabla\cdot\big(p_{\mathrm{data}}\, \partial_{\theta_j} u_\theta\big).$$

Applying Lemma 2 with u_θ replaced by $\partial_{\theta_j} u_\theta$ gives

$$-\mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} u_\theta(x_0))^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big] = \mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} r_\theta(x_0))\, r_\theta(x_t)\big]\,.$$

Similarly,

$$-\mathbb{E}\big[q(t)^{-1}\, u_\theta(x_0)^\top \partial x_t/\partial x_0^\top \nabla_{x_t} (\partial_{\theta_j} r_\theta(x_t))\big] = \mathbb{E}\big[q(t)^{-1}\, r_\theta(x_0)\, \partial_{\theta_j} r_\theta(x_t)\big]\,.$$

By reversibility of the stationary Langevin diffusion, $\mathbb{E}[\varphi(x_0,x_t)] = \mathbb{E}[\varphi(x_t,x_0)]$ for any test function φ, so the two right-hand sides are equal. Thus,

$$\partial_{\theta_j} \widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) = -2\, \mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} u_\theta(x_0))^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big]\,.$$

On the other hand, by definition of stop-gradient,

$$\begin{aligned}\partial_{\theta_j} \mathcal{L}_{\mathrm{Flux}}(\theta) &= -\mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} u_\theta(x_0))^\top \,\mathrm{sg}\big(\partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big)\big] \\ &= -\mathbb{E}\big[q(t)^{-1}\, (\partial_{\theta_j} u_\theta(x_0))^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big]\,.\end{aligned}$$

Therefore

$$\partial_{\theta_j} \widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) = 2\, \partial_{\theta_j} \mathcal{L}_{\mathrm{Flux}}(\theta) \qquad \text{for every } j,$$

and

$$\nabla_\theta \widetilde{\mathcal{L}}_{\mathrm{Flux}}(\theta) = 2\, \nabla_\theta \mathcal{L}_{\mathrm{Flux}}(\theta).$$

□

Theorem 3.1. *Assume $p_{\text{data}} > 0$ on $\mathbb{R}^d$ and boundary terms in integration-by-parts arguments vanish. Then,*

$$\nabla_\theta \widetilde{\mathcal{J}}(\theta) = 2\,\nabla_\theta \mathcal{L}_{\text{flux}}(\theta).$$

Proof. We have

$$\begin{aligned}
\mathcal{J}(\theta) &= \mathbb{E}_{x_0\sim p_{\text{data}}}\big[\|\Pi_{\text{flux}} f_\theta(x_0) - \nabla \log p_{\text{data}}(x_0)\|^2\big] \\
&= \int_0^\infty \mathbb{E}_{x_0\sim p_{\text{data}},\, x_t|x_0}[r_\theta(x_0)\, r_\theta(x_t)]\; dt && \text{(Lemma A.2)} \\
&= -\int_0^\infty \mathbb{E}_{x_0\sim p_{\text{data}},\, x_t|x_0}\big[u_\theta(x_0)^\top \partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big]\; dt && \text{(Lemma A.3).}
\end{aligned}$$

Differentiating with respect to θ and applying Lemma A.4 yields

$$\nabla_\theta \widetilde{\mathcal{J}}(\theta) = -2\,\nabla_\theta \mathbb{E}_{\substack{t\sim q \\ x_0\sim p_{\text{data}},\, x_t|x_0}}\big[q(t)^{-1}\, u_\theta(x_0)^\top \operatorname{sg}\big(\partial x_t/\partial x_0^\top \nabla_{x_t} r_\theta(x_t)\big)\big] = 2\,\nabla_\theta \mathcal{L}_{\text{Flux}}(\theta).$$

□

B Experiment Details

B.1 Application 1: *Controllable* Generative Fields

Goal. In this toy setting of a two-dimensional Gaussian mixture, we aim to show that (1) Flux Matching assign 0 loss to fields that preserve the target distribution but have very different dynamics, while penalizing fields that change the target distribution and (2) score matching assigns high loss to both fields that preserve the distribution (but have non score dynamics) and fields that change the target distribution. Ultimately, the core message of this experiment is that Flux Matching enables control over interesting attributes about the vector field (that still preserve the distribution) while score matching does not.

Target distribution. We use

$$p_\sigma(x) = \sum_{k=1}^{3} \pi_k\, \mathcal{N}(x; \mu_k, \nu_\sigma^2 I), \tag{14}$$

with

$$\mu_1 = (0,\, 2.3), \quad \mu_2 = (-1.99,\, -1.15), \quad \mu_3 = (1.99,\, -1.15), \quad \nu_\sigma^2 = \sigma_0^2 + \sigma^2 = 0.625. \tag{15}$$

The three modes lie at the vertices of an equilateral triangle. Since p_σ is known analytically, its score is also available in closed form:

$$s_\sigma(x) = \nabla \log p_\sigma(x) = \frac{m_\sigma(x) - x}{\nu_\sigma^2}, \qquad m_\sigma(x) = \sum_{k=1}^{3} \omega_k^\sigma(x)\mu_k, \tag{16}$$

where

$$\omega_k^\sigma(x) = \frac{\pi_k \exp\big(-\|x-\mu_k\|^2/(2\nu_\sigma^2)\big)}{\sum_{\ell=1}^{3} \pi_\ell \exp(-\|x-\mu_\ell\|^2/(2\nu_\sigma^2))}. \tag{17}$$

Vector field families. All fields are perturbations of the score,

$$f_\theta(x) = s_\sigma(x) + u_\theta(x), \qquad u_\theta(x) = \theta_0 c_{\text{rot}}(x) + \theta_1 c_{\text{tri}}(x) + \theta_2 c_{\text{skew}}(x). \tag{18}$$

The three controls (aka attributes) respectively induce global rotation, clockwise circulation around the triangle, and localized Jacobian asymmetry. We will then calculate specific metrics on these vector fields (specified under "Metrics"), which give us Figure 2 and Figure 4.

For the distribution-preserving family, we use

$$c_{\text{rot}}(x) \propto J s_\sigma(x), \qquad c_{\text{tri}}(x) \propto -\frac{J\nabla\psi_{\text{tri}}(x)}{p_\sigma(x)}, \qquad c_{\text{skew}}(x) \propto \frac{J\nabla\psi_{\text{skew}}(x)}{p_\sigma(x)}, \tag{19}$$

where $J = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$. The exact forms of ψ_{tri} and ψ_{skew} are not essential to this experiment; they are smooth templates chosen to produce visually interesting vector fields for Figure 2. The key structural requirement is that each distribution preserving attribute has the form $J\nabla\psi/p_\sigma$, which guarantees $\nabla \cdot (p_\sigma c) = 0$. Within this constraint, we tuned the template parameters to make the induced vector fields visually clear, high-contrast, and qualitatively distinct, where ψ_{tri} produces circulation along the triangle and ψ_{skew} produces a localized asymmetric perturbation. As a result, the somewhat elaborate formulas below should be viewed as implementation choices for constructing illustrative vector fields with interesting attributes, not as something rigorously motivated.

The scalar function ψ_{tri} concentrates near the triangle edges, while ψ_{skew} is a localized anisotropic bump. More precisely, let v_j be the triangle vertices, e_j the unit direction of edge j, n_j the outward normal to edge j, and ℓ_j the edge length. For each edge, define the along-edge coordinate and normal distance

$$a_j(x) = e_j^\top (x - v_j), \qquad d_j(x) = n_j^\top (x - v_j). \tag{20}$$

We define an edge-localized template by

$$R_j(x) = \exp\left(-\frac{d_j(x)^2}{2w^2}\right) \operatorname{sigmoid}\left(\frac{\kappa a_j(x)}{w}\right) \operatorname{sigmoid}\left(\frac{\kappa(\ell_j - a_j(x))}{w}\right), \tag{21}$$

which is large near edge j and small away from that edge segment. We also define a smooth triangle level function

$$h(x) = \frac{1}{\beta} \log \sum_{j=1}^{3} \exp\big(\beta(n_j^\top x - n_j^\top v_j)\big). \tag{22}$$

Then,

$$\psi_{\text{tri}}(x) = \exp\left(-\frac{\operatorname{ReLU}(h(x))^2}{2\eta^2}\right) \left[\frac{1}{3}\sum_{j=1}^{3} R_j(x) + \lambda_{\text{int}} \operatorname{sigmoid}(-\beta h(x))\right]. \tag{23}$$

In the experiment, we use $w = 0.24$, $\kappa = 5.0$, $\beta = 10.0$, $\lambda_{\text{int}} = 0.12$, and $\eta = 0.9$.

For the skew template, let

$$d = (\cos\alpha, \sin\alpha), \qquad z = \rho R d, \qquad d_\perp = Jd, \tag{24}$$

where $R = 2.3$, $\alpha = -20°$, and $\rho = 0.78$. For $r = x - z$, define

$$r_\parallel = d^\top r, \qquad r_\perp = d_\perp^\top r. \tag{25}$$

We set

$$\psi_{\text{skew}}(x) = \exp\left[-\frac{1}{2}\left(\frac{r_\parallel^2}{a^2} + \frac{r_\perp^2}{b^2}\right)\right] \left[1 + \lambda_{\text{bias}} \operatorname{sigmoid}\left(\frac{\gamma d^\top x}{R}\right)\right], \tag{26}$$

with $a = 0.60$, $b = 1.05$, $\lambda_{\text{bias}} = 0.45$, and $\gamma = 1.75$.

Each attribute vector field is constructed to be of 0 flux divergence (aka should make the Flux Matching loss 0):

$$\nabla \cdot \big(p_\sigma(x) c(x)\big) = 0. \tag{27}$$

Hence every linear combination satisfies

$$\nabla \cdot \big(p_\sigma(x) u_\theta(x)\big) = 0, \tag{28}$$

so these perturbations change the vector field dynamics without changing the stationary distribution p_σ. For comparison, we also construct a distribution-violating family by removing the $90°$ rotation from the same templates:

$$\tilde{c}_{\text{rot}}(x) = s_\sigma(x), \qquad \tilde{c}_{\text{tri}}(x) \propto \frac{\nabla\psi_{\text{tri}}(x)}{p_\sigma(x)}, \qquad \tilde{c}_{\text{skew}}(x) \propto \frac{\nabla\psi_{\text{skew}}(x)}{p_\sigma(x)}. \tag{29}$$

Then

$$u_\theta^{\text{viol}}(x) = \theta_0 \tilde{c}_{\text{rot}}(x) + \theta_1 \tilde{c}_{\text{tri}}(x) + \theta_2 \tilde{c}_{\text{skew}}(x), \qquad \nabla \cdot \big(p_\sigma u_\theta^{\text{viol}}\big) \neq 0 \tag{30}$$

in general, so these perturbations change the stationary distribution.

Finally, the distribution preserving attributes are rescaled to have unit RMS norm under p_σ:

$$c(x) \leftarrow \frac{c(x)}{\sqrt{\mathbb{E}_{x\sim p_\sigma}\|c(x)\|_2^2}}. \tag{31}$$

The scale factors are estimated using 1200 samples from p_σ and are reused for the corresponding distribution-violating fields, making the coefficients $\theta_0, \theta_1, \theta_2$ comparable across control directions.

Evaluation grid. All metrics are computed on a 25×25 grid over $[-5.5, 5.5]^2$ and let $\{x_i\}_{i=1}^N$ be the grid points. We use normalized density weights

$$w_i = \frac{p_\sigma(x_i)}{\sum_{j=1}^N p_\sigma(x_j)}. \tag{32}$$

Metrics. All metrics use the 25×25 grid and density weights $w_i \propto p_\sigma(x_i)$.

Mixing speed. For observables

$$\Phi = \{x_1,\ x_2,\ \|x\|_2,\ \cos(\angle x),\ \sin(\angle x)\}, \tag{33}$$

we discretize $\mathcal{L}_{f_\theta} = \Delta + f_\theta \cdot \nabla$ and, for each $\varphi \in \Phi$, solve the mean-constrained Poisson equation

$$-\mathcal{L}_{f_\theta} \psi_\varphi = \varphi - \mathbb{E}_w[\varphi]. \tag{34}$$

We estimate the integrated autocorrelation time by

$$\tau_\varphi(f_\theta) = \frac{2\langle \varphi - \mathbb{E}_w[\varphi],\ \psi_\varphi \rangle_w}{\mathrm{Var}_w(\varphi)}, \qquad M_{\mathrm{mix}}(f_\theta) = \left(\frac{1}{|\Phi|} \sum_{\varphi \in \Phi} \tau_\varphi(f_\theta) \right)^{-1}. \tag{35}$$

Larger M_{mix} means faster mixing.

Triangle shape. We measure cosine alignment between the perturbation and the triangle-shape control:

$$M_{\mathrm{tri}}(f_\theta) = \frac{\langle u_\theta, c_{\mathrm{tri}} \rangle_w}{\|u_\theta\|_w \, \|c_{\mathrm{tri}}\|_w}. \tag{36}$$

Jacobian skewness. For $u_\theta = (u_1, u_2)$, we measure the weighted squared antisymmetric Jacobian component:

$$M_{\mathrm{skew}}(f_\theta) = \sum_i w_i \, \frac{1}{2} \left[(\partial_{x_2} u_1)(x_i) - (\partial_{x_1} u_2)(x_i) \right]^2. \tag{37}$$

Distribution violation. For the non-preserving family, we measure the stationarity residual

$$r_\theta(x) = \nabla \cdot u_\theta^{\mathrm{viol}}(x) + u_\theta^{\mathrm{viol}}(x) \cdot s_\sigma(x), \qquad V(f_\theta) = \left(\sum_i w_i r_\theta(x_i)^2 \right)^{1/2}. \tag{38}$$

Here $V(f_\theta) = 0$ exactly when the perturbation preserves p_σ.

Losses. The score matching loss is

$$L_{\mathrm{SM}}(f_\theta) = \frac{1}{2} \mathbb{E}_{x \sim p_\sigma} \|f_\theta(x) - s_\sigma(x)\|_2^2 = \frac{1}{2} \mathbb{E}_{x \sim p_\sigma} \|u_\theta(x)\|_2^2. \tag{39}$$

Note again that $s_\sigma(x)$ is given to us in closed form via Equation (16). In the experiment, we approximate this expectation on the same 25×25 grid used for the metrics.

The Flux Matching loss is estimated separately by Monte Carlo, not on the 25×25 grid (since the Flux Matching loss requires short MCMC steps). For each θ, we sample $B = 256$ samples $x_0^{(i)} \sim p_\sigma$ as a minibatch, and compute the Flux Matching loss via Equation (6). We set the horizon sampler distributed to be the truncated uniform, $q = \mathcal{U}[0, T]$. We average this estimate over 64 independent minibatches.

Sweep and plotting. For Figure 4, we sweep

$$\theta_0, \theta_1, \theta_2 \in \{-2.5, 0, 2.5\}, \tag{40}$$

giving 27 fields. For each field, we record the relevant metric together with the score matching and Flux Matching losses. Since the two losses have different raw scales, each loss is divided by its mean value over the distribution-violating family before plotting. Curves are produced by binning the horizontal axis into 12 equally spaced bins and plotting the mean and standard error in each bin.

For Figure 2, we perform a separate one-dimensional scan along each control axis using 33 values in $[-8, 8]$. We display the field with the best value of each metric. For the Jacobian-skewness panel, if multiple fields are within 99.5% of the maximum skewness, we choose the one with the smallest triangle-flow alignment so that the displayed field isolates skewness rather than triangle circulation. The background empirical samples (in red dots) are 1800 draws from p_σ.

Algorithm 2 scVelo dynamical model fitting [5]

Input: Unspliced and spliced counts $\{(u_{ig}, s_{ig})\}_{i,g}$

for each gene g (independently) **do**
  Initialize Θ_g.
  repeat
    E-step. For each cell i:
      For each phase k, find the best latent time $\tau_{ig}^{(k)} = \text{argmax}_\tau \log p(u_{ig}, s_{ig} \mid k, \tau, \Theta_g)$.
      Pick the better phase: $k_{ig} = \text{argmax}_k \log p(u_{ig}, s_{ig} \mid k, \tau_{ig}^{(k)}, \Theta_g)$, and set $\tau_{ig} = \tau_{ig}^{(k_{ig})}$.
    M-step. Update $\Theta_g \leftarrow \text{argmax}_{\Theta_g} \sum_i \log p(u_{ig}, s_{ig} \mid k_{ig}, \tau_{ig}, \Theta_g)$.
  until convergence
end for

B.2 Application 2: *Interpretable* Generative Fields

Goal. The goal of this experiment is to show that Flux Matching can learn interpretable biological vector fields, such as RNA velocity. We use the scVelo dynamical model of [5], but fit its parameters with Flux Matching rather than their EM-style latent-time optimization.

Background on scVelo's Dynamical Model [5]. The Dynamical model generalizes RNA velocity beyond the original steady-state model [36] by fitting a likelihood-based dynamical model of splicing kinetics. The scVelo dynamical model describes each gene g's unspliced and spliced abundances by

$$\frac{du_g(\tau)}{d\tau} = \alpha_g(\tau) - \beta_g u_g(\tau), \qquad \frac{ds_g(\tau)}{d\tau} = \beta_g u_g(\tau) - \gamma_g s_g(\tau),$$

where the transcription rate switches between phases $k \in \{\text{INDUCTION}, \text{REPRESSION}\}$:

$$\alpha_g(\tau) = \begin{cases} \alpha_g, & k = \text{INDUCTION } (\tau \le \tau_s^g), \\ 0, & k = \text{REPRESSION } (\tau > \tau_s^g). \end{cases}$$

Induction starts from $(u_g, s_g) = (0, 0)$ at $\tau = 0$; repression starts from $(u_g(\tau_s^g), s_g(\tau_s^g))$. Closed-form solutions $\bar{u}_g(\tau, k; \Theta_g)$ and $\bar{s}_g(\tau, k; \Theta_g)$ yield a Gaussian observation model

$$\log p\big(u_{ig}, s_{ig} \mid k, \tau, \Theta_g\big) = \log \mathcal{N}\big(u_{ig};\, \bar{u}_g(\tau, k),\, \sigma_{u,g}^2\big) + \log \mathcal{N}\big(s_{ig};\, \bar{s}_g(\tau, k),\, \sigma_{s,g}^2\big),$$

with per-gene parameters $\Theta_g = (\alpha_g, \beta_g, \gamma_g, \tau_s^g, \sigma_{u,g}, \sigma_{s,g})$. The fitting procedure is summarized in Algorithm 2.

Interestingly, Flux Matching removes the need for these EM-style updates. Instead of alternating between assigning latent times under the current kinetic parameters and updating the parameters under those assignments, we directly optimize the parameters of the vector field using the Flux Matching loss.

Datasets. We use five RNA-velocity datasets: Bone marrow, Dentate Gyrus, Gastrulation, Hindbrain, and Pancreas. The datasets are loaded as follows

```
import scvelo as scv

datasets = {
    "bone_marrow": scv.datasets.bonemarrow(),
    "dentate_gyrus": scv.datasets.dentategyrus(),
    "gastrulation": scv.datasets.gastrulation(),
    "pancreas": scv.datasets.pancreas(),
    "hindbrain": scv.read("path/to/hindbrain.h5ad"),
}
```

We provide further instructions on obtaining and loading the hindbrain dataset in our code.

Table 2: Hyperparameters for the RNA-velocity experiment.

Hyperparameter	Value
Training iterations	100
Optimizer	Adam
Learning rate	10^{-3}
σ^2	10^{-3}
Horizon distribution	$q(t) = \mathcal{U}[0, T]$

Data preprocessing. We apply the same preprocessing pipeline to each dataset

```
import scanpy as sc
import scvelo as scv

sc.pp.filter_cells(adata, min_counts=1)
scv.pp.filter_and_normalize(adata, min_shared_counts=20)
sc.pp.highly_variable_genes(
    adata,
    n_top_genes=2000,
    flavor="seurat",
    subset=True,
)
sc.pp.pca(adata)
sc.pp.neighbors(adata, n_pcs=30, n_neighbors=30)
scv.pp.moments(adata, n_neighbors=None, n_pcs=None)
```

Model parameterization. The scVelo dynamical model uses gene-specific splicing and degradation dynamics that depend on a discrete latent transcriptional state (the 4 phases, namely induction initiation, induction saturation, repression initiation, and repression decay, details can be found in [5]). Because we optimize using gradient descent, we must use a continuous analogue to the discrete assignment:

$$\begin{aligned} \alpha_g(u_g, s_g) &= \mathrm{softplus}\left(a_g u_g + b_g s_g + c_g\right), \\ \frac{du_g}{d\tau} &= \alpha_g(u_g, s_g) - \beta_g u_g, \\ \frac{ds_g}{d\tau} &= \beta_g u_g - \gamma_g s_g. \end{aligned} \tag{41}$$

Here $\alpha_g(u_g, s_g) > 0$ is the learned transcription rate, approximated by a first-order approximation $\alpha_g(u_g, s_g) = \mathrm{softplus}(a_g u_g + b_g s_g + c_g)$, while $\beta, \gamma \geq 0$ are the learned splicing and degradation rates, respectively. The optimized gene-specific parameters are $\tilde{\beta}_g, \tilde{\gamma}_g, a_g, b_g, c_g$. We enforce positivity of the kinetic rates by setting

$$\beta_g = \mathrm{softplus}(\tilde{\beta}_g), \qquad \gamma_g = \mathrm{softplus}(\tilde{\gamma}_g). \tag{42}$$

The coefficients a_g, b_g, c_g provide a continuous analogue of the latent transcriptional-state assignment in the dynamical model. Rather than switching among discrete transcription rates, the model learns a smooth state-dependent transcription rate. We initialize a_g, b_g, c_g at zero, so the transcription rate is initially constant and the model begins as a simple linear kinetic ODE.

Training details. The RNA velocity experiments on all datasets use the same Flux Matching hyperparameters, summarized in Table 2. For the dynamical model, we use the package's default hyperparameters.

Visualizing Learned RNA Velocity. We provide visualizations of the inferred RNA velocity and ground truth progression of the bone marrow dataset in Figure 8, the dentate gyrus dataset in Figure 9, the gastrulation dataset in Figure 10, the hindbrain dataset in Figure 11, and the pancreas dataset in Figure 12.

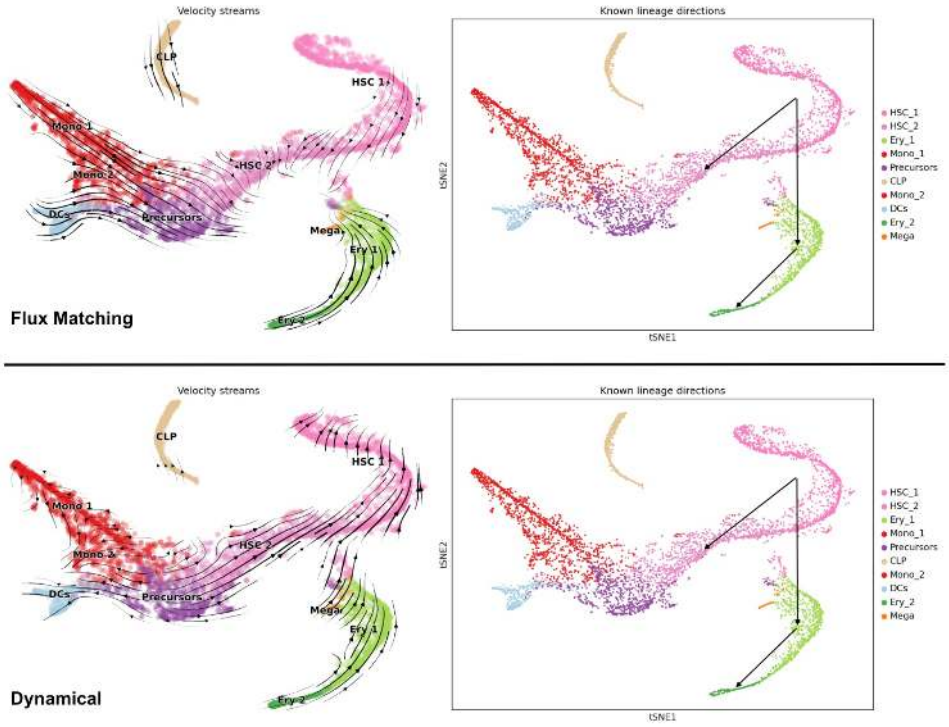


Figure 8: **Bone Marrow Dataset**. (Left half) inferred RNA velocity (Right half) ground truth biological progression given by arrows

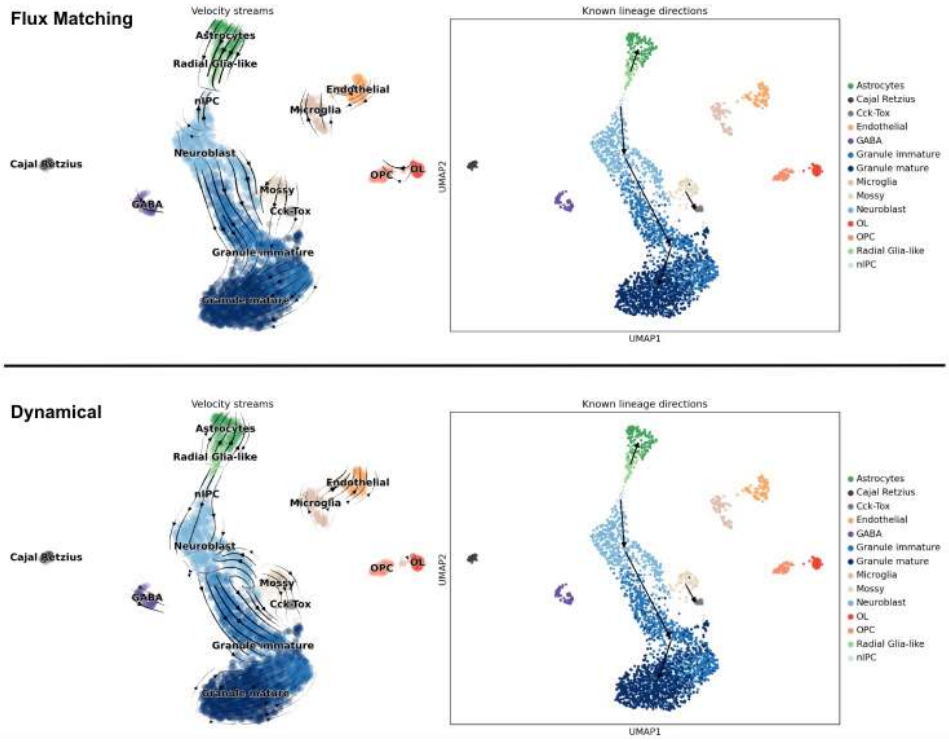


Figure 9: **Dentate Gyrus Dataset**. (Left half) inferred RNA velocity (Right half) ground truth biological progression given by arrows

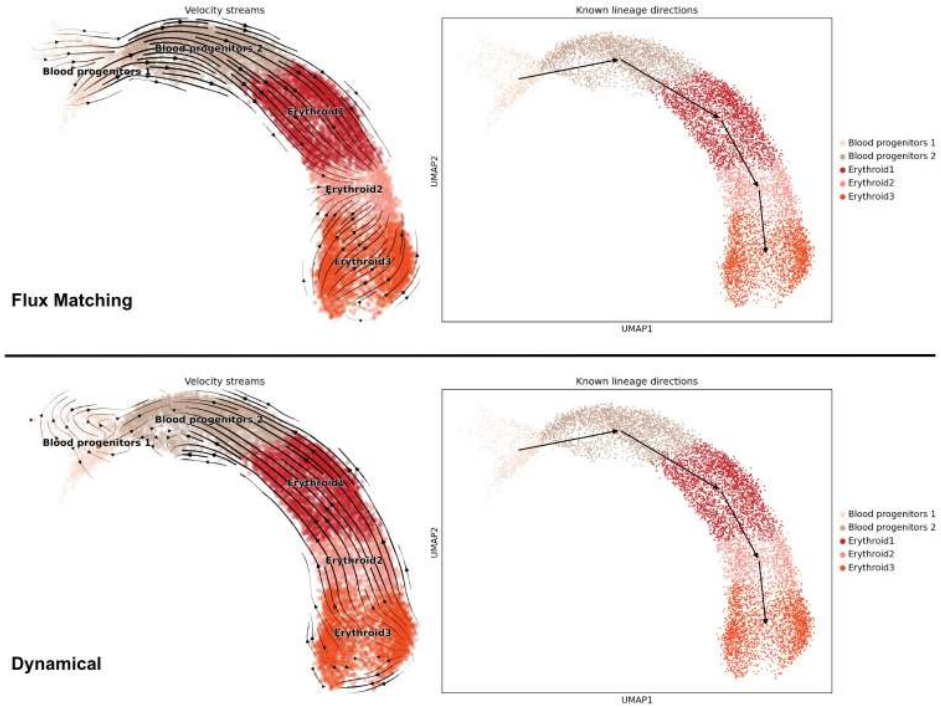


Figure 10: **Gastrulation Dataset**. (Left half) inferred RNA velocity (Right half) ground truth biological progression given by arrows

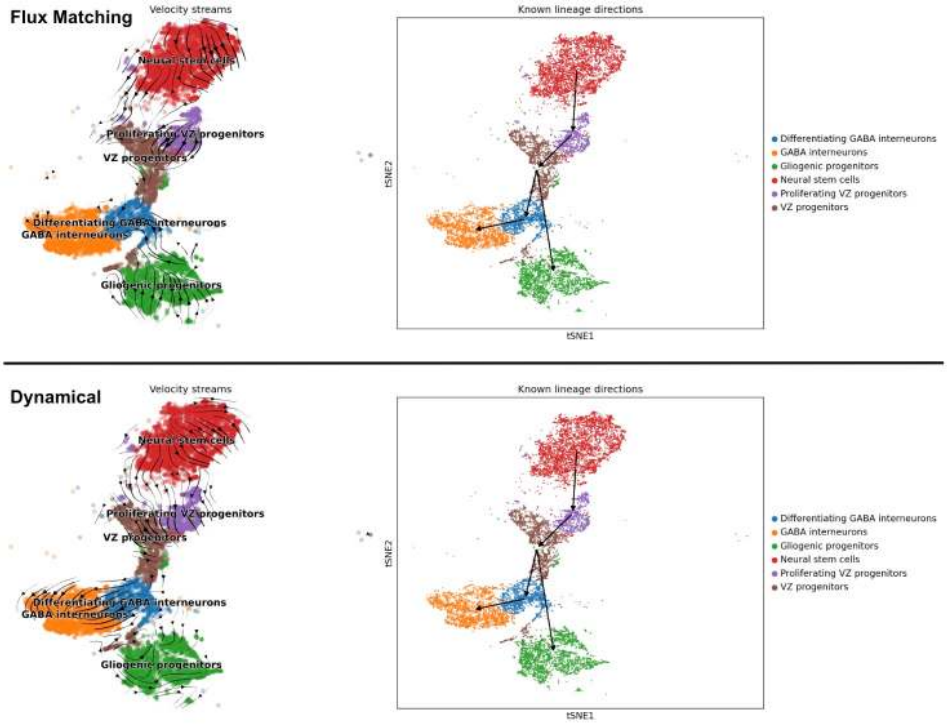


Figure 11: **Hindbrain Dataset**. (Left half) inferred RNA velocity (Right half) ground truth biological progression given by arrows

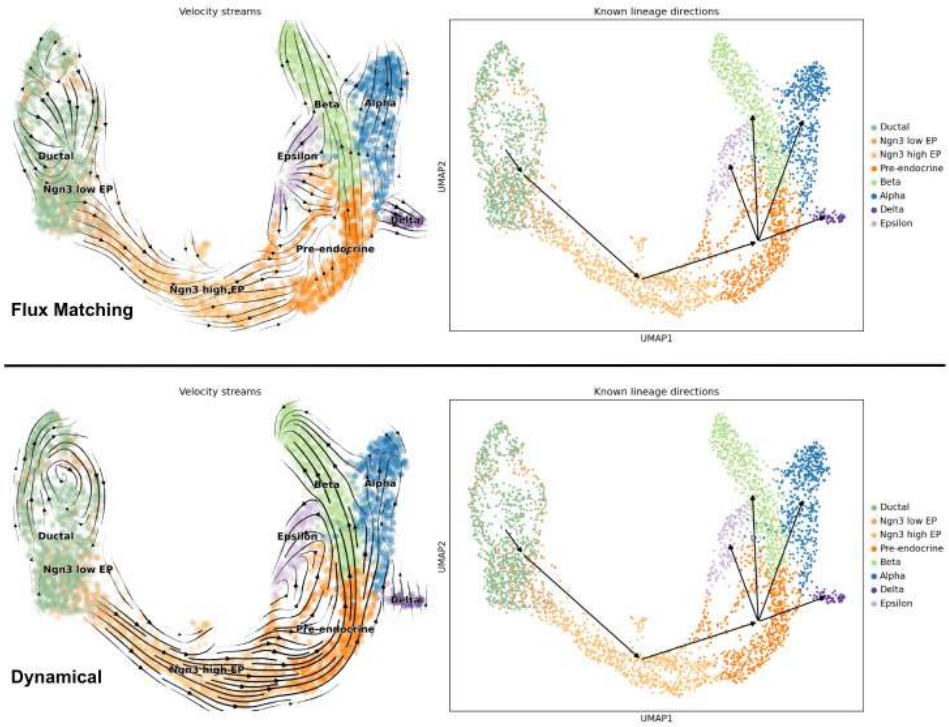


Figure 12: **Pancreas Dataset**. (Left half) inferred RNA velocity (Right half) ground truth biological progression given by arrows

Table 3: Model parameterization for Experiment 3 & 4.

Parameter	**CIFAR-10**	**CelebA**
Input shape	$3 \times 32 \times 32$	$3 \times 64 \times 64$
Base channels	128	128
Channel multipliers	$[1, 2, 2, 2]$	$[1, 2, 2, 2]$
Attention pattern	$[\texttt{False}, \texttt{True}, \texttt{False}, \texttt{False}]$	$[\texttt{False}, \texttt{True}, \texttt{False}, \texttt{False}]$
Dropout	0.1	0.1
Residual blocks per resolution	2	2
Number of parameters	35.7M	35.7M

B.3 Application 3: *Unrestricted* Generative Fields

Goal. While not the primary benefit of Flux Matching, we want to show in this experiment that Flux Matching can be used as a standalone generative model that performs well on high-dimensional, complex image distributions and scales efficiently in both runtime and memory.

Datasets. We evaluate on CIFAR-10 and CelebA. CIFAR-10 contains 50,000 training images with dimension $3 \times 32 \times 32$. CelebA contains 162,770 training images with dimension $3 \times 64 \times 64$.

Model parameterization. For both CIFAR-10 and CelebA, we use the standard U-Net architecture from [19]. The architecture details are given in Table 3.

Training objective. We train using the EDM noise parameterization and distribution with default EDM settings on bfloat16. Since Flux Matching uses a batch KDE estimate of the score in its loss, we compare against the stable target version of diffusion models [65],which uses the same batch KDE score representation. This baseline is stronger than single sample DSM since stable targets were shown to outperform the standard single sample DSM by reducing variance.

Table 4: Training and sampling hyperparameters for Experiments 3 & 4. Unless otherwise noted, the same values are used for CIFAR-10 and CelebA.

Training		**Sampling**	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
Optimizer	AdamW	Sampler	PF ODE
Learning rate	10^{-4}	Solver	Adaptive second-order Heun
EMA rate	0.9993	Implementation	`torchdiffeq.odeint`
Warmup steps	2,500	Method	`adaptive_heun`
Training iterations	500,000	Relative tolerance	10^{-5}
Number of GPUs	4	Absolute tolerance	10^{-5}
Batch size, CIFAR-10	128 per GPU	$\sigma_{\min}$	0.002
Batch size, CelebA	32 per GPU	$\sigma_{\max}$	80
Horizontal flips	`True`	Sampling interval	$[\sigma_{\min}, \sigma_{\max}]$

Training, sampling, and evaluation hyperparameters. We use the same hyperparameters for CIFAR-10 and CelebA except for the per-GPU batch size. During training, we compute 10K-image FID every 50,000 iterations. For the final reported results, we select the checkpoint with the best 10K-image FID and compute 50K-image FID, negative log-likelihood, and Inception score (standard evaluation procedure, following [59]). The negative log-likelihood is computed using the same probability-flow ODE solver and tolerances used for sampling. The main training and sampling hyperparameters are given in Table 4. Efficiency values reported in Table 1 were based on distributed training on 4 NVIDIA A100 GPUs. Furthermore, to ensure a fair comparison of efficiency values for CIFAR10 and CelebA, we standardize the batch size to 32 when computing wall-clock time and memory usage.

Extended samples. We show randomly generated CIFAR-10 samples in Figure 13 and CelebA samples in Figure 14.

B.4 Application 4: *Fast Mixing* Generative Fields for Accelerated Sampling

Goal. We want to show that Flux Matching is much more than just its unrestricted form (standalone objective), where optimizing different vector field attributes can have tangible benefits in purely generative modeling terms. Here, we show that we can reduce the number of sampling steps (aka number of function evalutions) by optimizing for vector fields with faster mixing properties.

Datasets, model parameterization, and evaluation. We use the same datasets, model parameterization, optimizer, training schedule, and noise distribution as in Section B.3. For checkpoint selection, we follow the same protocol as in Section B.3: for each method, we select the checkpoint with the best 10K FID using the adaptive Heun sampler. We then evaluate this selected checkpoint using the predictor–corrector sampler above and report 1K FID across different numbers of PC steps. These results are shown in Figure 6.

Training. Directly optimizing the mixing time of the sampler is intractable, so we introduce a tractable one-step proxy. Given a minibatch $\{x_0^{(i)}\}_{i=1}^{B} \sim p_\sigma$, we use the first half $\{x_0^{(i)}\}_{i=1}^{B/2}$ as a reference batch from the target marginal. From the second half, we construct an off-distribution batch by adding Gaussian noise $x_{\text{noise}}^{(i)} = x_0^{(i+B/2)} + \sigma z_0^{(i)}$ where $z_0^{(i)} \sim \mathcal{N}(0, I)$ for $i = 1, \ldots, B/2$. Starting from these off-distribution samples, we apply one Langevin step using the learned field f_θ^σ:

$$x_{\text{mix}}^{(i)} = x_{\text{noise}}^{(i)} + h_\sigma f_\theta^\sigma(x_{\text{noise}}^{(i)}) + \sqrt{2h_\sigma}\, z_{\text{noise}}^{(i)}, \qquad z_{\text{noise}}^{(i)} \sim \mathcal{N}(0, I), \qquad h_\sigma = 0.2\sigma^2. \tag{43}$$

Our proxy for mixing speed measures whether this single step moves x_{noise} closer to the target marginal:

$$\mathcal{L}_{\text{mixing}} = \mathbb{E}_{\sigma\sim\mathcal{P}}\left[\frac{\text{SD}\Big(\{x_{\text{mix}}^{(i)}\}_{i=1}^{B/2}, \{x_0^{(i)}\}_{i=1}^{B/2}\Big)}{\text{sg}\Big[\text{SD}\Big(\{x_{\text{noise}}^{(i)}\}_{i=1}^{B/2}, \{x_0^{(i)}\}_{i=1}^{B/2}\Big)\Big]}\, e^{-s_{\text{mixing}}(\sigma)} + s_{\text{mixing}}(\sigma)\right], \tag{44}$$

Figure 13: Random CIFAR-10 samples from Flux Matching (left) and DSM (right).

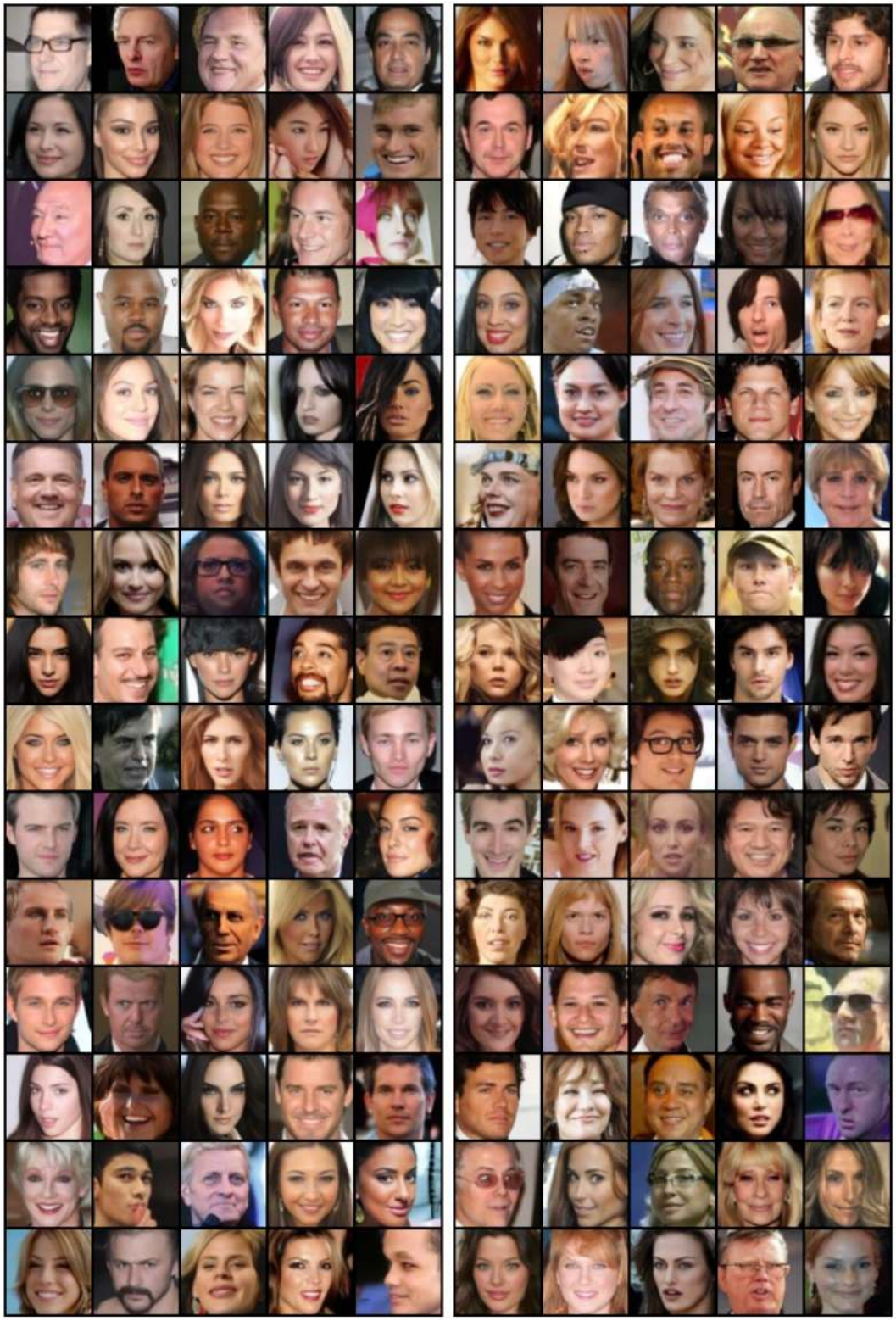


Figure 14: Random CelebA samples from Flux Matching (left) and DSM (right).

Algorithm 3 Predictor–Corrector sampler [59]

Set $\sigma_0 = \sigma_{\max} > \cdots > \sigma_K = \sigma_{\min}$ on a logarithmic grid.
Sample $x \sim \mathcal{N}(0, \sigma_{\max}^2 I)$.
for $k = 0, \ldots, K-1$ **do**
    Predictor step
    $\Delta_k \leftarrow \sigma_k^2 - \sigma_{k+1}^2$.
    if $k < K-1$ **then**
        Sample $z \sim \mathcal{N}(0, I)$.
        $x \leftarrow x + \Delta_k f_\theta^{\sigma_k}(x) + \sqrt{\Delta_k}\, z$.
    else
        $x \leftarrow x + \Delta_k f_\theta^{\sigma_k}(x)$.
    end if
    if $k < K-1$ **then**
        Corrector step
        Sample $z' \sim \mathcal{N}(0, I)$.
        $\epsilon_k \leftarrow 2\left(\rho \|z'\|_2 / (\|f_\theta^{\sigma_{k+1}}(x)\|_2)\right)^2$, with $\rho = 0.16$ (Table 5 of [59]).
        $x \leftarrow x + \epsilon_k f_\theta^{\sigma_{k+1}}(x) + \sqrt{2\epsilon_k}\, z'$.
    end if
end for

where SD is the Sinkhorn divergence and $s_{\text{mixing}}(\sigma)$ is a learned normalizer (similar as Section 3.4, parameterized as a 1-layer MLP and trained simultaneously with the main model) that keeps the mixing loss on a comparable scale across noise levels σ. We compute the Sinkhorn divergence with regularization parameter $\varepsilon = 0.05$ and 50 Sinkhorn iterations.

The final training objective is

$$\mathcal{L}_{\text{flux-fast}} = \mathcal{L}_{\text{flux-noise}} + \lambda_{\text{mix}}\, \mathcal{L}_{\text{mix}}, \tag{45}$$

with $\lambda_{\text{mix}} = 0.01$.

Sampling. We evaluate fast mixing using the predictor–corrector (PC) sampler in Algorithm 3, following [59]. We use a PC sampler rather than adaptive Heun because mixing speed concerns how quickly the sampling dynamics induced by a field converge to their stationary distribution. In the noised setting, the relevant stationary distribution at noise level σ is the noised marginal p_σ. Thus, a fast-mixing field should be able to take a sample that is not exactly distributed according to p_σ and quickly move it toward p_σ. This is precisely the role of the corrector step in PC sampling, while the predictor moves samples across noise levels. When we use fewer sampling steps, these moves between noise levels become larger and can move samples farther from the next noised marginal. The corrector then uses the learned field at the current noise level to pull samples back toward that marginal. Therefore, faster-mixing fields should enable fewer and larger predictor steps because even when the predictor introduces more error, the faster mixing corrector can remove that error more quickly.

For K PC steps, the sampler uses $2K$ evaluations of f_θ^σ: K predictor evaluations and K corrector evaluations.

B.5 Application 5: *Embedding Structure* in Generative Fields

Goal. This experiment tests whether Flux Matching can incorporate directed structural relationships between variables while still generating all variables in parallel. The data is trajectories, so a natural inductive bias is temporal directionality—the field at a given physical time point should depend only on the current and previous time points. Standard diffusion samplers also generate all time points in parallel, but DSM restricts the learned field to be the score function, which have symmetric Jacobians by equality of mixed partial derivatives, making strictly directed dependencies such as temporal autoregression incompatible with the true score. Flux Matching has no such constraint, so we can impose a causal temporal mask on f_θ^σ, while still generating the entire trajectory in one parallel sampling procedure. This masking restricts the effective hypothesis class toward fields that respect the temporal ordering of the data, which can improve data efficiency [4].

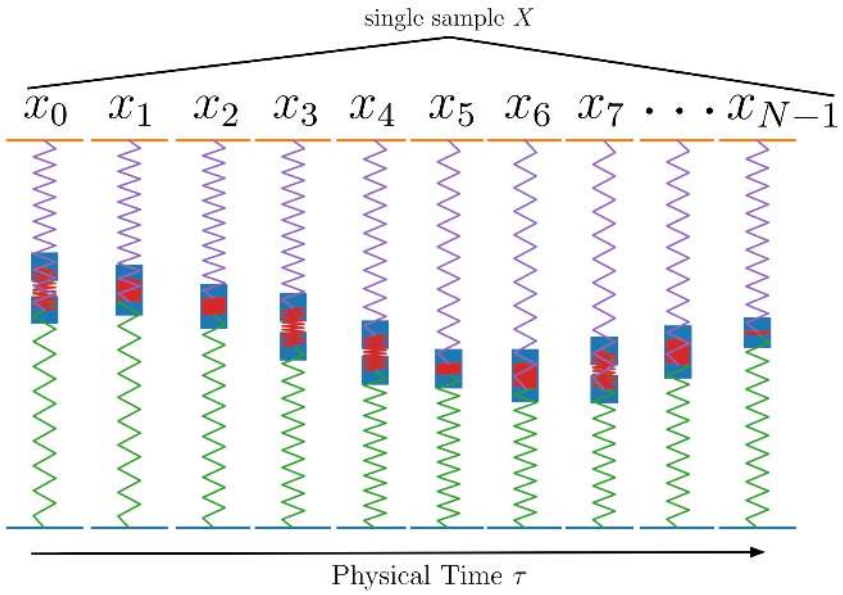


Figure 15: A single sample from the spring simulation. The system contains two masses, shown as blue blocks. Each mass is connected to a wall by a spring, shown in purple and green, and the two masses are connected to each other by a spring, shown in red. Each column corresponds to one physical time point and contains four features, which are the positions and velocities of the two masses. Collectively, these time points form one trajectory, which is one sample in the dataset.

Table 5: Dataset and simulator parameters for the nonlinear spring experiment.

Parameter	Value
Number of training trajectories	2000
Trajectory length	$N = 50$
State dimension per time point	4
Trajectory dimension after flattening	200
Wall spring constant	$k_{\text{wall}} = 1.0$
Coupling spring constant	$k_{\text{couple}} = 0.7$
Damping coefficient	$\gamma = 0.08$
Nonlinear spring coefficient	$c = 0.08$
Initial positions	$q_i(0) \sim \mathcal{N}(0, 3.0^2)$
Initial velocities	$v_i(0) \sim \mathcal{N}(0, 1.5^2)$

Dataset. Our dataset consists of simulated trajectories of two masses connected by nonlinear springs. Each data sample is a full trajectory over physical time. Let $x(\tau) = (q_1(\tau), v_1(\tau), q_2(\tau), v_2(\tau)) \in \mathbb{R}^4$ denote the simulator state at physical time τ, where $q_i(\tau)$ and $v_i(\tau)$ are the position and velocity of mass i. We record trajectories $X = (x_0, \ldots, x_{N-1}) \in \mathbb{R}^{N\times 4}$, where $x_n := x(n\Delta\tau)$. The trajectories are generated by integrating the ODE below with RK4 using physical step size $\Delta\tau = 0.10$:

$$\begin{aligned} \frac{dq_1}{d\tau} &= v_1, & \frac{dv_1}{d\tau} &= -k_{\text{wall}} q_1 - k_{\text{couple}}(q_1 - q_2) - \gamma v_1 - c q_1^3, \\ \frac{dq_2}{d\tau} &= v_2, & \frac{dv_2}{d\tau} &= -k_{\text{wall}} q_2 - k_{\text{couple}}(q_2 - q_1) - \gamma v_2 - c q_2^3. \end{aligned} \tag{46}$$

We use the simulator and dataset parameters in Table 5. Figure 15 visualizes one simulated trajectory.

Model parameterization. We use the same transformer architecture for Flux Matching and DSM. Each trajectory is represented as a sequence of physical-time tokens, where each token contains the state (q_1, v_1, q_2, v_2). The model first projects each token to a hidden representation of 512 dimensions, adds a sinusoidal positional embedding over physical time, and adds a learned embedding of the

Table 6: Training and sampling hyperparameters for Experiment 5.

Parameter	Value
Noise sampling	$\log \sigma \sim \mathrm{Unif}(\log \sigma_{\min}, \log \sigma_{\max})$
$\sigma_{\min}$	0.002
$\sigma_{\max}$	80.0
Optimizer	Adam
Training iterations	$10,000$
Batch size	256
Learning rate	3×10^{-4}
EMA rate	0.99
Sampler	Reverse VE sampler [59]
Sampling steps	512

noise level σ. The resulting sequence is processed by a stack of 8 heads of 12 residual self-attention blocks with LayerNorm, self-attention, and a GELU MLP. The final hidden states are normalized and projected back to the state dimension, producing one output vector per physical time point.

We compare two attention patterns. The noncausal model uses full self-attention across all physical time points. The causal model applies an upper-triangular attention mask over the physical-time dimension. In each attention block, if A_{ij} denotes the attention logit from time point i to time point j, then the causal model sets

$$A_{ij} = -\infty \qquad \text{whenever} \qquad j > i \tag{47}$$

before applying the softmax. As a result, the output at time point i can depend only on time points $0, \ldots, i$ (visualized on the left side of Figure 7). This enforces directed temporal dependence while preserving parallel generation, since all time points are still produced in a single forward pass.

Training objective. Since this experiment is not image-based, we use the standard variance-exploding (VE) noise distribution rather than the EDM noise distribution. Flux Matching is trained with the noised annealed Flux Matching loss in Section 3.4. For noise annealed DSM, we use the stronger baseline of the stable-target variant from [65], as was done in Section B.3.

We train four models: Flux Matching with full attention, Flux Matching with causal attention, DSM with full attention, and DSM with causal attention (all of which are noise annealed versions). This isolates the effect of adding a directed temporal mask under each training objective.

Training, sampling, and evaluation hyperparameters. All training and sampling hyperparameters are shown in Table 6. At sampling time, we use the reverse VE sampler (Eq. 9 from [59]). We evaluate sample quality using Wasserstein distance $\mathcal{W}_2$ between 2000 generated trajectories and training trajectories.

C Other Applications of Flux Matching

We briefly outline additional potential applications of Flux Matching beyond those presented in the main text. These are directions we find exciting but were unable to pursue in this paper, and we hope they spark future research.

C.1 Causality

Vector fields, expressed as the drift of ordinary differential equations (ODEs) and stochastic differential equations (SDEs) [55, 23], can be used to represent causal structure. Recent works [42, 6] have explored learning generative vector fields to discover causal structures and perform inference by simulating the learned SDE. The core limitation of these works [42, 6] is their inability to scale to high-dimensional data. For instance, [42, 6] primarily evaluated on data with 20 dimensions, far fewer than what is needed in settings like single-cell transcriptomics (via Perturb-seq [14]), which involves ~ 20000 dimensions. Notably, we can simply replace their loss function—KDS in [42] and

SKDS in [6]—with the Flux Matching loss to scale causal learning to very high dimensions, since, as shown in Section 4.3, Flux Matching naturally scales to high-dimensional image datasets.

C.2 Generative Modeling in Constrained Domains

Prior work has studied diffusion models on constrained domains by either designing custom diffusion processes that respect the constraint set Ω [43] or by correcting invalid proposals through rejection or Metropolis-style steps when samples leave Ω [18]. Flux Matching could be a complementary approach. Even when the target distribution is supported on a constrained domain, there are many distribution generating vector fields that behave very differently near the boundary $\partial\Omega$. Under a finite-step sampler, a field with large outward components near the boundary is more likely to produce invalid samples, while a field that is tangent to the boundary or points inward is more likely to remain inside the domain. Since Flux Matching does not require the learned field to equal the score, we can use the additional degrees of freedom to favor boundary-respecting dynamics. For example, the training objective could be designed so we penalize the learned field when it violates the support under the chosen discretization.

C.3 Spatially Structured Dependencies in Images

Many generative problems have known structure among different parts of the sample like what we have outlined in Section 4.5. We focus here specifically on the application of images, where one example of structured dependencies mean allowing some regions of the image to influence others, while other interactions are not allowed. For example, the appearance of a human face may influence the appearance of sunglasses, but it should not directly alter the background. Flux Matching makes it possible to encode such region-to-region relationships directly in the architecture of the learned generative vector field (via, for example, masked attention).

C.4 Augmenting Existing Score-Based Models

As shown in Equation (3), we can add any flux divergence-free term to $\nabla \log p_{\mathrm{data}}$ while still preserving the target distribution. Suppose we already have a trained off-the-shelf score-based model, and we want to leverage the benefits of Flux Matching (like accelerated sampling). Rather than training a new model from scratch using Flux Matching, we can train a network v_ϕ such that $\nabla \cdot (p_{\mathrm{data}} v_\phi) = 0$ using an analogous version of the Flux Matching loss:

$$\mathcal{L}_{\mathrm{flux-aug}}(\theta) := -\mathbb{E}_{\substack{t\sim q \\ x_0\sim p_{\mathrm{data}},\, x_t|x_0}} \left[\frac{1}{q(t)}\, v_\phi(x_0)^\top \,\mathrm{sg}\left(\frac{\partial x_t}{\partial x_0}^\top \nabla_{x_t} \mathcal{F}(x_t) \right) \right], \tag{48}$$

where $\mathcal{F}(x_t) := \nabla \cdot v_\phi(x_t) + v_\phi(x_t) \cdot \nabla \log p_{\mathrm{data}}(x_t)$. Then, after v_ϕ is trained using $\mathcal{L}_{\mathrm{flux-aug}}(\theta)$, add v_ϕ to the already trained score model, $f_\theta = \nabla \log p + v_\phi$, and proceed to sample using f_θ.

This application can be useful in, for example, the case of faster mixing fields for accelerated sampling. We can simply train a fast mixing acceleration layer that we can "tack" onto an existing diffusion model we want to accelerate.

D Flux Matching Details

D.1 Component Calculation Specifications

We provide an extended description of the different components in Section 3.3.

D.1.1 Sampling MCMC horizon t from importance sampler q

As mentioned in the main text, even though q is supported on $[0, \infty)$, in practice, we find that defining the simulation horizon sampler q to be either a truncated uniform or exponential on $[0, T]$ and setting $T = 4\sigma^2$ is sufficient. The bottom row of Figure 16 supports this truncation: empirically, $\mathcal{L}_{\mathrm{flux}}$ decays approximately exponentially in t. Thus, after a sufficiently large cutoff, the remaining tail contribution is negligible. We find that setting $T = 4\sigma^2$ is sufficient, as shown in Figure 16 since little mass remains after the red cutoff (set at $T = 4\sigma^2$). The simplest distribution for q is the uniform

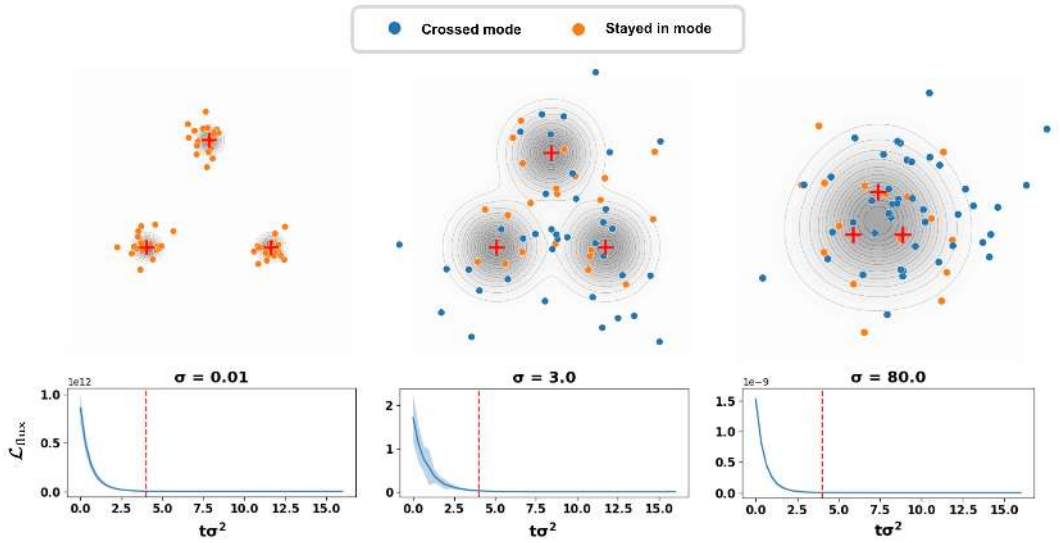


Figure 16: Mode crossing and loss variance in a three-component Gaussian mixture. The component means are fixed, while the component variance increases with σ. (**Top row**) show terminal samples after running Langevin dynamics for 100 steps at different noise levels. Samples are colored blue if their terminal point is assigned to a different mode than their initial point, and orange otherwise. (**Bottom row**) shows the corresponding $\mathcal{L}_{\text{flux}}$ as a function of the simulation horizon, with shaded bands denoting standard deviation.

density $\mathcal{U}[0, T]$, but since the loss follows an exponential shape over t, a lower variance alternative is to sample from a truncated exponential distribution

$$q_\lambda(t) = \lambda e^{-\lambda t} / \left(1 - e^{-\lambda T}\right), \qquad t \in [0, T], \quad \lambda > 0, \tag{49}$$

where λ can be cheaply fitted during training as a single scalar parameter via

$$\mathcal{L}(\lambda) := -\mathbb{E}_{t\sim\bar{q}}\left[\text{sg}\Big(\widehat{\mathcal{L}}_{\text{flux}}(t)\Big)\log q_\lambda(t)\right]. \tag{50}$$

where $\bar{q}$ is a fixed copy used to draw the current horizon t and $\widehat{\mathcal{L}}_{\text{flux}}(t)$ is the realized Flux Matching loss. In the case of noise annealed Flux matching, we similarly learn $\lambda_\phi(\sigma)$, the rate of the truncated-exponential sampler $q_{\lambda_\phi(\sigma)}$ that is now dependent on σ. Let $\bar{q}_{\lambda_\phi(\sigma)}$ denote the fixed copy of this sampler used to draw the current horizon t, and let $\widehat{\mathcal{L}}^\sigma_{\text{flux}}(t)$ be the realized noise annealed Flux Matching loss, already importance reweighted by the sampling density $\bar{q}_{\lambda_\phi(\sigma)}$. We fit λ_ϕ with

$$\mathcal{L}_q(\phi) := -\mathbb{E}_{\sigma\sim\mathcal{P},\, t\sim\bar{q}_{\lambda_\phi(\sigma)}}\left[\text{sg}\left(\frac{\widehat{\mathcal{L}}^\sigma_{\text{flux}}(t)}{\exp(s_\eta(\sigma))}\right)\log q_{\lambda_\phi(\sigma)}(t)\right]. \tag{51}$$

where $s_\eta(\sigma)$ is the learned normalizer (single-layer MLP) that reweighs losses from different noise levels to be on comparable scales [34]. $\lambda_\phi(\sigma)$ is also parameterized by a single-layer MLP and fitted simultaneously with the main network.

D.1.2 Estimating ${}^{\partial x_t}/_{\partial x_0}{}^\top \nabla_{x_t} r_\theta(x_t)$

We set weights w_{ij} to be

$$w_{ij}(t) = \text{softmax}_i\left(-\left\|x_t^{(j)} - x_0^{(i)} - \sigma^2(1-e^{-t})\nabla \log p_\sigma(x_0^{(i)})\right\|^2 / \left[2\sigma^2(1-e^{-2t})\right]\right). \tag{52}$$

D.2 Variance at Intermediate Noise Levels

In the noise annealed version of Flux Matching, we observe that the variance of the objective can vary substantially across noise levels. The bottom row of Figure 16 illustrates this effect. At very low noise

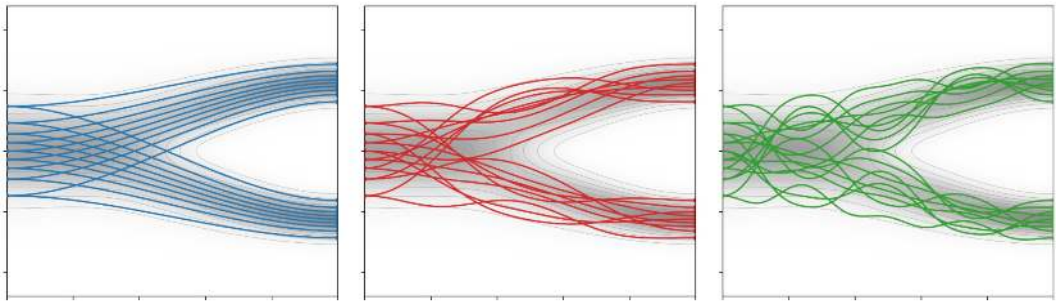


Figure 17: Distribution flows can have the same marginal evolution but different particle trajectories. The gray background shows the same path of marginals, while the solid colored curves show different individual transports that realize this path.

($\sigma = 0.01$) and very high noise ($\sigma = 80.0$), the empirical $\mathcal{L}_{\text{flux}}$ has relatively small variance, as indicated by the narrow standard-deviation bands. In contrast, at intermediate noise levels ($\sigma \approx 3.0$), the standard-deviation bands are much larger across simulation horizons, suggesting that this regime produces a substantially higher-variance estimator.

The top row of Figure 16 provides intuition for this phenomenon. At low noise, the modes are well separated, and Langevin chains tend to remain in the same mode even after many simulation steps. At high noise, although chains mix more readily, the noised distribution is already close to a single Gaussian, so mode crossing is less informative and less variable. The intermediate regime is a difficult sweet spot where the effective supports of the modes begin to overlap, but the modes still remain distinct. As a result, some chains cross between modes while others do not, producing large sample-to-sample variation in the loss estimate. A related observation was also made by [65], who found that intermediate noise levels can have higher variance in their learning objective.

This intermediate regime is also where the cross-chain minibatch estimator in Section 3.2, used to estimate ${\partial x_t}/{\partial x_0}^\top \nabla_{x_t} r_\theta(x_t)$, is most useful. By averaging information across chains in the minibatch, the estimator is less sensitive to whether any single chain crosses modes. In contrast, we found that the single-chain estimator $\nabla_{x_0} r_\theta(x_t)$ is often sufficient in the low- and high-noise regimes, where the loss variance is much smaller.

D.3 Architectures Need to Be Gradient Friendly

The Flux Matching loss in Equation (6) requires differentiating r_θ with respect to the input. Since r_θ itself contains a divergence of the learned vector field, training depends on input derivatives of f_θ beyond the vector-field level. Thus, architectures used to parameterize f_θ should have well-behaved input gradients and divergences. We did not study this issue systematically, but we found that using the specific architecture of ViT [3] to parameterize f_θ does not work well (while UNet did) in Flux Matching. We hypothesize that the patching procedure used in ViT may be problematic, but this remains an open question for future work.

E v-Flux Matching for Distribution Flows

Flux Matching is not restricted to matching the score. More generally, it can match the flux divergence induced by any vector field. This allows us to start from a distribution flow model, such as flow matching [40, 60] or related constructions [45, 2, 41], and learn alternative particle dynamics that preserve the same marginal path.

New Capability: *Same* Marginal Path, *Many* Particle Paths. Flux Matching can learn different individual transports while preserving the same evolution of distributions.

E.1 Distribution Flows and Equivalent Vector Fields

Let $a \in [0,1]$ denote distribution flow time, and let $\{p_a\}_{a\in[0,1]}$ be a path of densities. A velocity field v_a induces this path if it satisfies the continuity equation

$$\frac{\partial p_a(x)}{\partial a} = -\nabla \cdot (p_a(x) v_a(x)) \,. \tag{53}$$

The velocity field that realizes a given marginal path is not unique: any two vector fields v_a and f_a induce the same instantaneous distributional change whenever

$$\nabla \cdot (p_a f_a) = \nabla \cdot (p_a v_a) \,. \tag{54}$$

Assume $\nabla \cdot (p_a f_a) = \nabla \cdot (p_a v_a)$ for all $a \in [0,1]$. If v_a and f_a are initialized with the same density p_0, they induce the same marginal path $\{p_a\}_{a\in[0,1]}$, and in particular the same terminal distribution p_1.

E.2 v-Flux Matching at a Single Marginal

For clarity, we first fix a distribution flow time a and a reference velocity field v_a. Let $(x_t)_{t\geq 0}$ denote the diffusion $dx_t = \nabla \log p_a(x_t)\, dt + \sqrt{2}\, dW_t$ with stationary density p_a. To emphasize, the diffusion is only used to obtain the correct geometry on the loss; the target vector field being flux matched is v_a, not $\nabla \log p_a$. Let

$$u_{\theta,a}(x) = f_\theta(x,a) - v_a(x), \qquad r_{\theta,a}(x) = \nabla \cdot u_{\theta,a}(x) + u_{\theta,a}(x) \cdot \nabla \log p_a(x). \tag{55}$$

Define

$$\mathcal{L}^a_{\text{v-flux}}(\theta) := -\mathbb{E}_{\substack{t\sim q \\ x_0\sim p_a,\, x_t|x_0}} \left[\frac{1}{q(t)} u_{\theta,a}(x_0)^\top \operatorname{sg}\left(\frac{\partial x_t}{\partial x_0}^\top \nabla_{x_t} r_{\theta,a}(x_t) \right) \right]. \tag{56}$$

This is the same objective as Equation (6), with p_{data} replaced by p_a and the score target replaced by v_a.

Corollary E.1 (v-Flux Matching)**.** *Fix $a \in [0,1]$. Assume $p_a > 0$ on $\mathbb{R}^d$ and boundary terms in integration-by-parts arguments vanish. Let $\Pi^{p_a}_{\text{flux}}$ follow the same definition as Equation* (4) *with respect to p_a. Let*

$$\widetilde{\mathcal{J}}^a_{\text{v-flux}}(\theta) := \mathbb{E}_{x\sim p_a} \left[\left\| \Pi^{p_a}_{\text{flux}} f_\theta(\cdot, a)(x) - \Pi^{p_a}_{\text{flux}} v_a(x) \right\|^2 \right]. \tag{57}$$

Then

$$\nabla_\theta \widetilde{\mathcal{J}}^a_{\text{v-flux}}(\theta) = 2 \nabla_\theta \mathcal{L}^a_{\text{v-flux}}(\theta). \tag{58}$$

Proof. This is a direct application of Theorem 3.1. The proof of Theorem 3.1 only depends on the density p_{data} and the mismatch field u_θ. Setting $p_{\text{data}} = p_a$ and $u_\theta = u_{\theta,a} = f_\theta(\cdot, a) - v_a$ gives exactly Equation (56) and Equation (57). □

Marginal Velocity for Flow Matching Paths. The marginal velocity depends on the path used to define the interpolating marginals p_a. As a simple example, consider the conditional flow matching path of [40] with $x_1 \sim p_1$ and

$$p_a(x \mid x_1) = \mathcal{N}\big(x;\, a x_1,\, (1-a)^2 I\big) \,. \tag{59}$$

Equivalently, $x_a = a x_1 + (1-a)\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$. The marginal velocity can be approximated by the minibatch $\{x_1^{(i)}\}_{i=1}^B$, which gives

$$v_a(x) \approx \sum_{i=1}^B w_i(x,a)\, \frac{x_1^{(i)} - x}{1-a}, \qquad w_i(x,a) = \frac{p_a(x \mid x_1^{(i)})}{\sum_{j=1}^B p_a(x \mid x_1^{(j)})}. \tag{60}$$

which is a common approximation used in few-step generative models (e.g. [20]).

E.3 Pathwise v-Flux Matching

The previous subsection defined v-Flux Matching at a fixed marginal p_a. To match an entire distribution flow, we apply the same objective independently along the path $\{p_a\}_{a\in[0,1]}$. The pathwise v-Flux Matching objective is

$$\mathcal{L}_{\text{path-v-flux}}(\theta,\phi,\eta) := \mathbb{E}_{a\sim\nu}\left[\mathcal{L}^a_{\text{v-flux}}(\theta,\phi)/\exp(s_\eta(a)) + s_\eta(a)\right], \tag{61}$$

where ν is the sampling distribution over flow times $a \in [0,1]$, and $\mathcal{L}^a_{\text{v-flux}}(\theta,\phi)$ denotes the fixed-a loss in Equation (56) with $f^a_\theta(x) := f_\theta(x,a)$ and diffusion simulation horizon distribution $q_{\lambda_\phi(a)}$. All diffusion simulations are performed with the score $\nabla \log p_a$. The learned normalizer $s_\eta(a)$ reweighs losses from different marginals so that they remain on comparable scales [34], while $\lambda_\phi(a)$ parameterizes the diffusion horizon sampler at each point along the path.

F Related Work

F.1 *Analyzing* Non-Conservative Generative Vector Fields

A long line of work from the MCMC and statistical physics literature has analyzed the space of vector fields that preserve a given stationary distribution and characterized the benefits of departing from reversibility. Augmenting the score with a divergence-free flux component has been shown to accelerate convergence to the stationary distribution [30, 53] (which we empirically corroborate in Section 4.4) and to reduce the asymptotic variance of resulting estimators [16, 17]. [44] gives a complete parameterization of all continuous Markov processes admitting a prescribed stationary distribution. Crucially, these works assume a closed-form target distribution and prescribe the non-conservative drift analytically rather than learning it from data. Flux Matching is, to our knowledge, the first objective to operationalize this body of theory into a learning objective for non-conservative generative dynamics that scales to high-dimensional distributions.

F.2 *Learning* Non-Conservative Generative Vector Fields

A handful of prior works have explored learning non-score generative vector fields. [66] learns a non-conservative vector field to assign dynamics to snapshot data, similar to our RNA velocity experiment, but their method is restricted to 2D. The objectives of [42, 6], while originally proposed for causal SDEs, could in principle learn arbitrary generative vector fields. However, neither scales beyond 20D. Flux Matching is novel not by the goal of learning non-gradient fields, but by being the first objective to do so while scaling to high-dimensional, complex distributions.

[52] frame their method as learning non-gradient field dynamics, yet it fundamentally matches a prior to a terminal distribution via a learned (rather than predefined) interpolation, a special case of [46]. Crucially, their method cannot learn non-gradient fields given a single distribution. The two approaches are in fact complementary, with [52] producing a bridge of distributions and Flux Matching providing flexibility over the individual particle trajectories that realize the given bridge.

F.3 Enforcing the Fokker–Planck (or Continuity) Equation

[38] and [28] enforce the Fokker–Planck equation (respectively, the continuity equation) as a regularizer on top of a primary score matching or flow matching objective, providing tighter control over the induced PDE. In contrast, Flux Matching is a standalone generative objective: [38, 28] add a regularizer to a generative loss, whereas Flux Matching *is* the generative loss. Moreover, both prior methods are restricted to score matching/flow matching-style trajectories, while Flux Matching admits arbitrary trajectories realizing the same marginals.

Via the Fokker–Planck, [27] observe that diffusion models possess a gauge degree of freedom (as we formalize in Proposition 2.1), but treat this purely as an observation with no associated learning procedure.

G Limitations

Flux Matching is most useful when the goal is not only to model a distribution, but also to learn a generative vector field with additional desired structure. If one only cares about unrestricted

generative modeling, then standard DSM already provides a highly optimized objective. Flux Matching should be viewed less as a replacement for DSM and more as a paradigm that enables use cases where the vector field itself matters. The main practical limitation is computational cost, with our implementation being roughly 2–$4\times$ more expensive than DSM in both runtime and memory. Reducing this overhead is an important direction for future work.

Further, the flexibility that Flux Matching provides is only useful when paired with a meaningful inductive bias or auxiliary objective. Flux Matching expands the class of learnable generative vector fields but does not automatically identify the best member for a given application, and designing architectures or regularizers that exploit this extra freedom remains problem dependent. Finding clever and elegant ways of constraining the learned vector field to exhibit a desired attribute is itself the central task facing a practitioner using Flux Matching. We expect that individual instantiations of such constructions—each tailored to a particular structural or application-specific goal—can constitute a substantial contribution in their own right.