Riemannian MeanFlow

Anonymous Authors[1]

Abstract

Diffusion and flow models have become the dominant paradigm for generative modeling on Riemannian manifolds, with successful applications in protein backbone generation and DNA sequence design. However, these methods require tens to hundreds of neural network evaluations at inference time, which can become a computational bottleneck in large-scale scientific sampling workflows. We introduce Riemannian MeanFlow (RMF), a framework for learning flow maps directly on manifolds, enabling high-quality generations with as few as one forward pass. We derive three equivalent characterizations of the manifold average velocity (Eulerian, Lagrangian, and semigroup identities), and analyze parameterizations and stabilization techniques to improve training on high-dimensional manifolds. In promoter DNA design and protein backbone generation settings, RMF achieves comparable sample quality to prior methods while requiring up to $10\times$ fewer function evaluations. Finally, we show that few-step flow maps enable efficient reward-guided design through reward look-ahead, where terminal states can be predicted from intermediate steps at minimal additional cost.

1. Introduction

Many scientific data types possess intrinsic geometric structure that is not faithfully captured by Euclidean representations. For example, protein backbones are naturally described as sequences of rigid-body transformations, where each residue frame encodes both position and orientation (Jumper et al., 2021; Watson et al., 2022; Yim et al., 2023a;b; Bose et al., 2023). DNA and RNA sequences are distributions over nucleotides constrained to the probability simplex (Stark et al., 2024; Davis et al., 2024; Cheng et al., 2024), while molecular conformations are parameterized by torsion angles lying on circles (Jing et al., 2022). Naively embedding such data in $\mathbb{R}^d$ ignores these constraints, leading to invalid samples, e.g., unnormalized token probabilities or discontinuous angles, and inefficient learning.

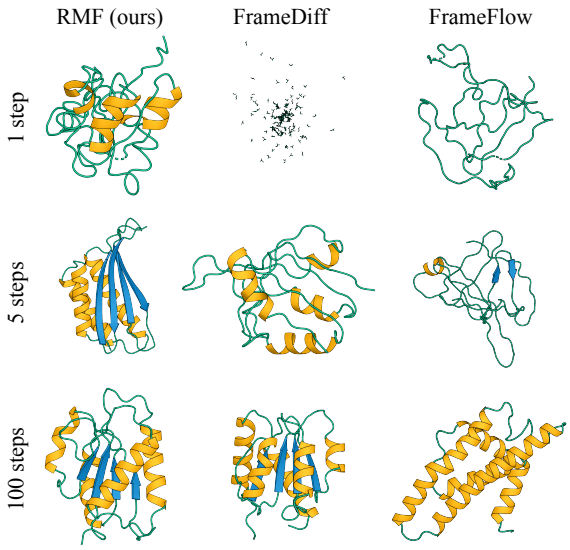


Figure 1. Protein backbone samples from RMF (ours), FrameDiff, and FrameFlow for different inference budgets. RMF produces well-formed structures in one step, while baselines require more.

Riemannian geometry provides a natural mathematical framework for modeling such data. By defining generative models directly on the appropriate manifold, such as $\mathrm{SE}(3)^N$ for protein backbones or the simplex Δ^{d-1} for sequences, geometric constraints are satisfied by construction. Building on the success of diffusion models and flow matching in Euclidean settings (Song et al., 2020; Lipman et al., 2022; Albergo et al., 2023), recent work has extended these continuous-time generative models to Riemannian manifolds, learning vector fields that transport noise to data along geodesic paths (Huang et al., 2022; De Bortoli et al., 2022; Chen & Lipman, 2023). These geometric generative models have achieved notable success in protein backbone generation (Yim et al., 2023a;b; Bose et al., 2023) and DNA sequence design (Davis et al., 2024; Cheng et al., 2024).

Despite this progress, the inference cost of manifold generative models remains a critical bottleneck. Sampling requires numerically integrating an ODE or SDE along the manifold, often demanding tens to hundreds of neural network evalu-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ations (Song et al., 2020; Lipman et al., 2022). This computational burden is particularly problematic in scientific design pipelines, where generative models serve not as one-off samplers, but as a proposal mechanism within iterative loops, e.g., for property-guided optimization (Yang et al., 2020; Pacesa et al., 2024; Han et al., 2025). When each proposal requires hundreds of forward passes, the scope of exploration can become limited.

In Euclidean settings, this challenge has driven extensive research into *few-step* generation. Consistency models (Song et al., 2023; Song & Dhariwal, 2023) enforce self-consistency along trajectory paths, while flow map methods (Geng et al., 2025; Zhou et al., 2025; Guo et al., 2025; Boffi et al., 2025) learn to transport between arbitrary time points via average-velocity regression. These approaches achieve high-quality generation with far fewer function evaluations, narrowing the gap with multi-step methods. However, these methods remain underexplored in the Riemannian manifold setting, limiting their applicability in scientific domains.

Contributions. To address this gap, we present Riemannian MeanFlow (RMF), a framework for few-step generation on Riemannian manifolds. Our contributions are:

1. **Riemannian MeanFlow identities:** We derive three equivalent and *complete* characterizations of the manifold average velocity, each with a distinct training objective: Eulerian, Lagrangian, and semigroup. We find that differential objectives (Eulerian, Lagrangian) can exhibit high variance due to differential terms related to manifold curvature in the regression targets, while the algebraic semigroup objective avoids this and provides more stable optimization in high dimensions.

2. **Scalable parameterization:** We propose x_1-prediction for manifold flow maps, where the network predicts a manifold-valued endpoint rather than a tangent vector. In our experiments, x_1-prediction performs comparably or better than v-prediction and scales well to high dimensions (up to $D = 2048$), making it compatible with existing scientific architectures that output manifold-valued points. We identify that the best combination for stable training on high-dimensional manifolds is the semigroup objective with x_1-prediction.

3. **Applications in scientific design:** Recent works have generalized consistency models (Cheng et al., 2025) and flow-map learning (Davis et al., 2025) to manifolds, but validation remains limited to low-dimensional benchmarks ($\mathbb{S}^2$, $\mathrm{SO}(3)$, torsion angles). We demonstrate that RMF can scale to real scientific tasks, showing that few-step generation is viable in high dimensions. On DNA promoter design (simplex in $\mathbb{R}^{1024\times 4}$) and protein backbone generation ($\mathrm{SE}(3)^N$ with N up to 128), RMF matches the performance of the state-of-the-art multi-step generative models (Yim et al., 2023a; Davis et al., 2024) with up to $10\times$ fewer function evaluations. We further demonstrate reward-guided generation via reward look-ahead, which has not been previously shown on manifolds.

2. Riemannian MeanFlow Identities

2.1. Background on Riemannian Geometry

We consider a smooth, connected Riemannian manifold $\mathcal{M}$ and its Riemannian metric g. At each point $x \in \mathcal{M}$, the tangent space $T_x\mathcal{M}$ is a vector space of velocities with inner product $\langle \cdot, \cdot \rangle_g$ and norm $\| \cdot \|_g$. The disjoint union of all tangent spaces forms a tangent bundle $T\mathcal{M} := \bigsqcup_{x\in\mathcal{M}} T_x\mathcal{M}$. For manifolds embedded in ambient d-dimensional Euclidean space $\mathcal{M} \subset \mathbb{R}^d$, we let $\mathrm{Proj}_x : \mathbb{R}^d \to T_x\mathcal{M}$ denote tangential projection of vectors from the ambient space onto the tangent space at x. We provide a self-contained tutorial on Riemannian geometry in App. A and summarize key concepts needed for our framework below.

Exponential and logarithmic maps. The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ takes a tangent vector v and returns the endpoint of the unit-time geodesic (locally shortest path) starting at x with initial velocity v. Its local inverse, the logarithmic map $\log_x : \mathcal{M} \to T_x\mathcal{M}$, returns the initial velocity needed to reach a target point.

Covariant derivatives. To realize differentiation of vector fields on the manifold, we use the Levi-Civita connection ∇. For a vector field $V(t)$ along a curve $\gamma(t)$, the covariant derivative $D_tV := \nabla_{\dot\gamma} V$ measures how the vector field V changes along the curve γ while accounting for the manifold's curvature. For a manifold embedded in Euclidean space, this corresponds to projecting the standard Euclidean derivative onto the tangent space: $D_tV = \mathrm{Proj}_{\gamma(t)}(\frac{d}{dt}V)$.

Derivatives of the logarithmic map. Since $\log : \mathcal{M} \times \mathcal{M} \to T\mathcal{M}$ is a function of two arguments, we distinguish its partial derivatives: $\nabla^1_v \log_x(y)$ denotes the covariant derivative with respect to the first argument x in direction v, while $d(\log_x)_y[w]$ denotes the differential with respect to the second argument y in direction w. These quantities appear in our training objectives and have closed-form expressions for manifolds of interest (spheres, simplices, $\mathrm{SE}(3)$); see App. A for details.

2.2. Flow Maps and Average Velocity on Manifolds

We now define the central objects of our framework. Our goal is to learn a generative model that transports a prior distribution p_0 to a data distribution p_1. This transport is governed by the ODE $\frac{dx_t}{dt} = v_t(x_t)$, where $v_t : \mathcal{M} \to T\mathcal{M}$ is a time-dependent vector field. We assume a chosen

interpolant determines a family of intermediate marginals $\{p_t\}_{t\in[0,1]}$, where p_t denotes the distribution at time t. In this setting, rather than learning v_t and integrating it at inference time (as in flow matching), we directly learn the *flow map* that transports from one time point to another.

Definition 2.1 (Integral curve)**.** Given a time-dependent vector field $v : [0,1] \times \mathcal{M} \to T\mathcal{M}$, an integral curve is a smooth path $x : [0,1] \to \mathcal{M}$ satisfying $\frac{d}{dt}x_t = v_t(x_t)$. We use the notation x_s, x_t, x_r to denote points on the same integral curve at times s, t, r respectively.

Definition 2.2 (Flow map)**.** The flow map $\Phi_{s,t} : \mathcal{M} \to \mathcal{M}$ of a vector field v_t is the mapping that transports points along integral curves: $\Phi_{s,t}(x_s) = x_t$ for any integral curve $(x_t)_{t\in[0,1]}$. The flow map satisfies the semigroup property $\Phi_{r,t} \circ \Phi_{s,r} = \Phi_{s,t}$, which states that flowing from s to r and then from r to t agrees with the direct flow from s to t.

In Euclidean space, the flow map can be parameterized through the *average velocity*: the constant velocity that, if maintained from time s to t, would transport x_s to the same final point x_t. On manifolds, constant-velocity motion corresponds to traveling along geodesics. The Euclidean difference $x_t - x_s$ generalizes to the logarithmic map $\log_{x_s} x_t$, leading to the following definition:

Definition 2.3 (Average velocity)**.** The average velocity $u_{s,t} : \mathcal{M} \to T\mathcal{M}$ for a vector field v_t is defined as

$$u_{s,t}(x_s) = \begin{cases} \dfrac{1}{t-s}\log_{x_s} x_t, & t \neq s, \\ v_s(x_s), & t = s, \end{cases} \tag{1}$$

for any integral curve $(x_t)_{t\in[0,1]}$ and times $s, t \in [0,1]$.

Geometrically, $u_{s,t}(x_s)$ is the constant velocity that would transport x_s to x_t over time of $t-s$ along a geodesic. The flow map can be recovered via the exponential map:

$$\Phi_{s,t}(x) = \exp_x\big((t-s)\,u_{s,t}(x)\big), \quad \forall\, s,t \in [0,1]. \tag{2}$$

2.3. Riemannian MeanFlow Identities

We present three equivalent characterizations of the average velocity. Each identity provides a necessary and sufficient condition: any vector field satisfying the identity must be the true average velocity. These identities form the basis of our training objectives in Sec. 3.1. The first two identities are obtained from differentiating the defining relation:

$$(t-s)\,u_{s,t}(x_s) = \log_{x_s} x_t \tag{3}$$

with respect to either s or t. Following conventions in fluid mechanics, we call differentiation with respect to the source time s the *Eulerian* perspective, and differentiation with respect to the target time t the *Lagrangian* perspective.

Proposition 2.1 (Eulerian RMF)**.** *A vector field $u_{s,t} : \mathcal{M} \to T\mathcal{M}$ is the average velocity associated with v_t* ***if and only if*** *it satisfies*

$$u_{s,t}(x_s) = (t-s)\,D_s u_{s,t}(x_s) - \nabla^1_{v_s}\log_{x_s} x_t, \tag{4}$$

for any integral curve $(x_t)_{t\in[0,1]}$ and any $s, t \in [0,1]$.

Proof sketch. Differentiating Eq. (3) with respect to s gives

$$-u_{s,t}(x_s) + (t-s)\,D_s u_{s,t}(x_s) = D_s(\log_{x_s} x_t), \tag{5}$$

where the covariant derivatives appear due to differentiating vector fields along the integral curve at x_s. The right-hand side equals $\nabla^1_{v_s}\log_{x_s} x_t$ by the chain rule for covariant derivatives, where $v_s = \frac{d}{ds}x_s$ is the velocity along the integral curve. Rearranging yields Eq. (4). See App. B.1 for the complete proof. □

Proposition 2.2 (Lagrangian RMF)**.** *A vector field $u_{s,t} : \mathcal{M} \to T\mathcal{M}$ is the average velocity associated with v_t* ***if and only if it*** *satisfies*

$$u_{s,t}(x_s) = d(\log_{x_s})_{x_t}[v_t] - (t-s)\,\partial_t u_{s,t}(x_s), \tag{6}$$

for any integral curve $(x_t)_{t\in[0,1]}$ and any $s, t \in [0,1]$.

Proof sketch. Differentiating Eq. (3) with respect to t gives

$$u_{s,t}(x_s) + (t-s)\,\partial_t u_{s,t}(x_s) = d(\log_{x_s})_{x_t}[v_t], \tag{7}$$

where the right-hand side is the differential of $\log_{x_s}$ at x_t applied to $v_t = \frac{d}{dt}x_t$. Rearranging yields Eq. (6). For the complete proof, refer to App. B.1. □

The third identity is algebraic rather than differential, following directly from the semigroup property $\Phi_{r,t} \circ \Phi_{s,r} = \Phi_{s,t}$. We provide the proof in App. B.1.

Proposition 2.3 (Semigroup RMF)**.** *A vector field $u_{s,t} : \mathcal{M} \to T\mathcal{M}$ is the average velocity associated with v_t* ***if and only if*** *the following two conditions hold:*

(i) Boundary condition: $u_{s,s}(x) = v_s(x),\ \forall\, x \in \mathcal{M}$;

(ii) Semigroup consistency: for any $s \neq t$ and intermediate time $r \in [s,t]$,

$$u_{s,t}(x_s) = \frac{1}{t-s}\log_{x_s}\Phi_{r,t}\big(\Phi_{s,r}(x_s)\big), \tag{8}$$

where $\Phi_{s,t}(x) \coloneqq \exp_x\big((t-s)\,u_{s,t}(x)\big)$ is the flow map induced by $u_{s,t}$.

3. Flow Map Learning with Riemannian MF

3.1. Training Objectives

To learn a flow map transporting a prior p_0 to data distribution p_1, we parameterize the average velocity via a neural network $u^\theta_{s,t} : \mathcal{M} \to T\mathcal{M}$. The induced flow map is then:

$$\Phi^\theta_{s,t}(x) \coloneqq \exp_x\big((t-s), u^\theta_{s,t}(x)\big). \tag{9}$$

From identities to objectives. Each identity from Sec. 2.3 yields a training objective by converting the consistency condition into a regression target.

Proposition 3.1 (Riemannian MeanFlow objectives)**.** *Let $u^\theta_{s,t} : \mathcal{M} \to T\mathcal{M}$ be a parameterized average velocity with induced flow map $\Phi^\theta_{s,t}$ as in Eq. (9). The following objectives are valid for learning the average velocity:*

1. ***Eulerian RMF:*** *Sample $x_s \sim p_s$ and $s, t \sim p(s,t)$. The objective is*

$$\mathcal{L}_{\mathrm{EMF}}(\theta) = \mathbb{E}_{x_s,s,t}\Big[\big\|u^\theta_{s,t}(x_s) - \mathrm{sg}(\hat{u}_{\mathrm{tgt}})\big\|_g^2\Big], \tag{10}$$

where $\hat{u}_{\mathrm{tgt}} = (t-s)\, D_s u^\theta_{s,t}(x_s) - \nabla^1_{v_s} \log_{x_s} \Phi^\theta_{s,t}(x_s)$.

2. ***Lagrangian RMF:*** *Sample $x_t \sim p_t$ and $s, t \sim p(s,t)$. Let $\hat{x}_s = \Phi^\theta_{t,s}(x_t)$. The objective is*

$$\mathcal{L}_{\mathrm{LMF}}(\theta) = \mathbb{E}_{x_t,s,t}\Big[\big\|u^\theta_{s,t}(\hat{x}_s) - \mathrm{sg}(\hat{u}_{\mathrm{tgt}})\big\|_g^2\Big] + \mathcal{L}_{\mathrm{cyc}}(\theta), \tag{11}$$

where $\hat{u}_{\mathrm{tgt}} = \mathrm{d}(\log_{\hat{x}_s})_{x_t}[v_t] - (t-s)\,\partial_t u^\theta_{s,t}(\hat{x}_s)$, and the cycle-consistency regularizer is defined as:

$$\mathcal{L}_{\mathrm{cyc}}(\theta) = \mathbb{E}_{x_t,s,t}\big[d_g(\Phi^\theta_{s,t}(\Phi^\theta_{t,s}(x_t)), x_t)^2\big], \tag{12}$$

where d_g denotes the geodesic distance.

3. ***Semigroup RMF:*** *Sample $x_s \sim p_s$ and times $s, r, t \sim p(s,r,t)$. The objective is*

$$\mathcal{L}_{\mathrm{SMF}}(\theta) = \mathbb{E}_{x_s,s,r,t}\Big[\big\|u^\theta_{s,t}(x_s) - \mathrm{sg}(\hat{u}_{\mathrm{tgt}})\big\|_g^2\Big], \tag{13}$$

where $\hat{u}_{\mathrm{tgt}} = \frac{1}{t-s}\log_{x_s} \Phi^\theta_{r,t}\big(\Phi^\theta_{s,r}(x_s)\big)$ for $t \neq s$, and $\hat{u}_{\mathrm{tgt}} = v_s$ for $t = s$.

Here, $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator.

Full proofs are in App. B.2. The key step in deriving our objectives lies in converting the average velocity characterizations from Sec. 2.3 into self-consistent regression targets. Specifically, the Eulerian characterization requires evaluation at x_t, necessitating integration over the unknown v_t starting from x_s. Instead, we replace x_t with the current model prediction $\Phi^\theta_{s,t}(x_s)$. The Lagrangian case is analogous, with $(x_s, x_t, \Phi^\theta_{s,t}(x_s))$ replaced by $(x_t, x_s, \Phi^\theta_{t,s}(x_t))$. For Lagrangian RMF, we add a cycle-consistency loss to encourage weak invertibility, as the regression input is model-predicted. Conversely, the semigroup identity is inherently self-consistent and directly yields a regression target.

Stop-gradient and bias. The stop-gradient operator $\mathrm{sg}(\cdot)$ treats the target as a constant during backpropagation, preventing gradients from flowing through $\hat{u}_{\mathrm{tgt}}$. This avoids expensive higher-order derivatives (e.g., gradients through Jacobian–vector products (JVPs)) and stabilizes optimization. Importantly, this is unbiased at convergence: when $u^\theta_{s,t}$ matches the true average velocity, the underlying identity is satisfied, and the gradient of the loss vanishes, regardless of whether gradients are propagated through the target.

Approximating the marginal velocity. In practice, the marginal velocity v_s or v_t is intractable. As in flow matching, we replace it with a tractable conditional velocity. Importantly, in all of our objectives, the velocity appears only through linear differential operators. As a result, taking the expectation over the conditioning variable commutes with these operators, so this replacement does not affect the objective in expectation. We show this proof in App. B.2.

Computation of differential terms. The covariant derivative $D_s u^\theta_{s,t}$ and partial derivative $\partial_t u^\theta_{s,t}$ in the differential objectives can be computed efficiently using forward-mode automatic differentiation via JVPs, which adds less than 20% overhead compared with a standard forward pass (Geng et al., 2025). For embedded manifolds, the covariant derivative can be obtained by computing the JVP in the ambient space and projecting the result onto the tangent space: $D_s u^\theta_{s,t}(x_s) = \mathrm{Proj}_{x_s}\big(\frac{d}{ds} u^\theta_{s,t}(x_s)\big)$. The differential quantities $\nabla^1 \log$ and $\mathrm{d}(\log_{x_s})$ admit closed-form expressions for manifolds of interest. In practice, we implement these operations using automatic differentiation, enabling efficient and flexible evaluation across different manifold choices.

3.2. Parameterization of the Flow Map

We consider three parameterizations for the average velocity $u^\theta_{s,t}$: prediction of v, x_t, or x_1, and identify x_1-prediction as the best-suited for manifold settings.

v-prediction. The most direct approach parametrizes the average velocity:

$$u^\theta_{s,t}(x_s) = \mathrm{Proj}_{x_s}\big(\mathtt{net}^\theta(x_s, s, t)\big) \in T_{x_s}\mathcal{M}, \tag{14}$$

where Proj_{x_s} projects the network output onto the tangent space. The instantaneous velocity is recovered as $v^\theta_s(x_s) = u^\theta_{s,s}(x_s)$, and the flow map follows from Eq. (9). This parameterization is conceptually simple and commonly adopted in prior work on Euclidean flow maps.

x_t-prediction. An alternative is to directly model the flow map: $\Phi^\theta_{s,t}(x_s) = \mathtt{net}^\theta(x_s, s, t)$. The average velocity is then recovered via the logarithmic map. However, training

requires enforcing the boundary condition $\Phi^{\theta}_{s,s}(x_s) = x_s$ and matching the instantaneous velocity $\frac{d}{dt}\Phi^{\theta}_{s,t}(x_s)\big|_{t=s} = v_s(x_s)$, which requires differentiating through the network at every step. This introduces significant computational overhead and instability.

x_1-prediction. We propose an x_1-prediction scheme that inherits the benefits of endpoint prediction from flow matching while accommodating the two-time structure of flow maps. The network predicts a point on the manifold, which we interpret as an estimate of the trajectory endpoint:

$$\hat{x}^{\theta}_1(x_s, s, t) = \texttt{net}^{\theta}(x_s, s, t), \tag{15}$$

$$u^{\theta}_{s,t}(x_s) = \frac{1}{1-s}\log_{x_s}\hat{x}^{\theta}_1(x_s, s, t). \tag{16}$$

Unlike standard x_1-prediction in flow matching, the predicted endpoint depends on both times s and t, since $u_{s,t}$ is a function of both time variables. A practical advantage of this parameterization is compatibility with existing architectures that output manifold-valued points. For protein backbones, models such as FrameDiff (Yim et al., 2023b), FrameFlow (Yim et al., 2023a) and FoldFlow (Bose et al., 2023) naturally predict $\mathrm{SE}(3)$ frames. Thus, x_1-prediction can reuse such architectures without modification, which is convenient for repurposing pre-trained flow matching models as flow maps, whereas v-prediction would require modifying the output to produce tangent vectors.

Numerical stability near $s = 1$. The factor $(1-s)^{-1}$ in Eq. (16) diverges as $s \to 1$, which can destabilize training. To mitigate this issue, we reweight the per-sample error inside the norm using a factor $w(s)$:

$$w(s) = \frac{1-s}{\max(1-s, \epsilon)}, \tag{17}$$

where we set $\epsilon \in \{0.05, 0.1\}$ in our experiments, following common practices in flow matching with x_1-prediction (Yim et al., 2023a; Bose et al., 2023; Li & He, 2025). In practice, this weighting stabilizes x_1-prediction in all tested settings.

3.3. Stabilizing Riemannian MF Training

We identify several sources of optimization difficulty and present practical stabilization techniques below. App. G.1 provides empirical support for these choices.

Time sampling distribution. We find that stable optimization requires different time-sampling schemes across the three objectives. For Eulerian MF, while the objective is agnostic to time ordering, we sample ordered time pairs with $s \le t$, covering half the unit square $(s, t) \in [0, 1]^2$. For Lagrangian MF, we sample time pairs in both directions, including both $s < t$ and $t < s$. Finally, semigroup MF also samples an intermediate time r such that $s < r < t$.

Adaptive loss weighting. Differential objectives, particularly Eulerian MF, can suffer from high variance in derivative-dependent regression targets. To stabilize training, we adopt adaptive loss weighting:

$$\mathcal{L} = \mathrm{sg}(w)\,\|\Delta\|_g^2, \quad w = (\|\Delta\|_g^2 + c)^{-p}, \tag{18}$$

with $p = 0.5$. This substantially improves sample quality while maintaining stability, consistent with prior observations (Song & Dhariwal, 2023; Geng et al., 2025).

Time-derivative control. We find that bounding the time derivative of the network output is crucial for differential objectives. Using low-frequency time embeddings (e.g., $\omega = 0.02$) significantly stabilizes training compared to high-frequency embeddings (e.g., $\omega = 30$), which is in-line with consistency-model literature (Lu & Song, 2024).

3.4. Reward-guided Inference with Flow Maps

Finally, we study reward-guided inference in the manifold setting for steering generations toward downstream objectives without model retraining (Skreta et al., 2025; Hasan et al., 2026). A common strategy is to use gradients of a differentiable reward to bias the generative dynamics when sampling. We incorporate this signal directly into the learned flow field during inference by perturbing the transport with guidance vector $\zeta_t \in T_{x_t}\mathcal{M}$ and guidance scale λ:

$$x_{t+\Delta t} = \exp_{x_t}\left(\Delta t\left(u^{\theta}_{t,t+\Delta t}(x_t) + \lambda \cdot \zeta_t\right)\right). \tag{19}$$

In practice, the performance of the guided generation significantly depends on the choice of ζ_t. The naive approach is to define the guidance vector as the tangential projection of the Riemannian reward gradient $\nabla_{x_t} r(x_t)$ evaluated at the current state, i.e. $\mathrm{Proj}_{x_t}(\nabla_{x_t} r(x_t))$. However, evaluating the reward on x_t is suboptimal, especially at small t, since it is only partially denoised.

Sabour et al. (2025a), using their flow map model, proved that evaluating the reward using an x_1 look-ahead generated samples from the product density $\propto\ p_1(x)\exp(r(x))$. Similarly, we demonstrate that using our flow map model in the manifold setting for reward evaluation is beneficial for guidance. In our setting, we use the guidance vector $\zeta_t = \mathrm{Proj}_{x_t}(\nabla_{x_t} r(\Phi^{\theta}_{t,1}(x_t)))$ as a simple heuristic.

4. Experiments

We first conduct an empirical analysis of key design choices to establish practical guidelines for flow map learning. Using these findings, we then evaluate RMF on two biological tasks: promoter DNA design and protein backbone generation. Finally, we show that we can do flow map-based reward-guided inference for both applications.

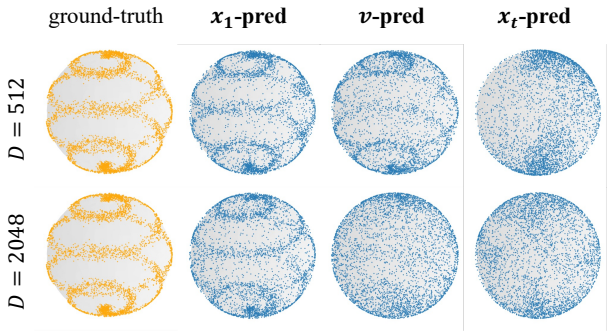


Figure 2. Parameterization choices: One-step generation results from models trained with different parameterizations (x_1-, v-, and x_t-pred) across ambient dimensions $D \in \{512, 2048\}$.

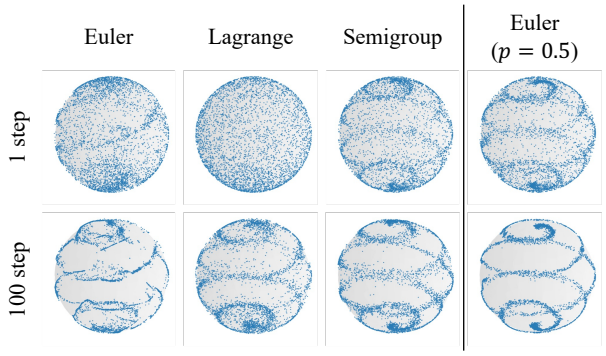


Figure 3. Objective choices: (Left; Columns 1-3) Samples generated by models trained with different objectives using 1-step (top) or 100-step (bottom) sampling on $D = 512$. (Right; Column 4) Adaptive loss weighting for Eulerian RMF substantially improves sample quality.

4.1. Ablation of Design Choices

We empirically study key design choices in flow map learning, focusing on parameterization and training objectives. Ablations for stabilization techniques are in App. G.1.

Task description. To emulate the manifold hypothesis, we use a synthetic 2D spherical helix embedded in a high-dimensional space. Points in the spherical helix $x \in \mathbb{S}^2 \subset \mathbb{R}^3$ are mapped to $y = Ux \in \mathbb{S}^{D-1} \subset \mathbb{R}^D$ via an unknown fixed column-orthogonal matrix $U \in \mathbb{R}^{D\times 3}$. For visualization, samples are projected back via $\tilde{x} = U^\top y \in \mathbb{R}^3$ and normalized as $x = \tilde{x}/\|\tilde{x}\|_2 \in \mathbb{S}^2$. Performance is evaluated across $D \in \{512, 2048\}$ using a 256-wide, 5-layer MLP.

Parameterization choices. We compare x_1-, x_t-, and v-prediction under the semigroup RMF objective. As shown in Fig. 2, x_1-prediction remains stable across both dimensions, while x_t-prediction completely fails. v-prediction degrades as D increases; x_1-prediction performs well even at $D = 2048$, despite under-parameterization (i.e., network width $\ll D$).

Objective choices. We compare Eulerian, Lagrangian, and semigroup RMF objectives at ambient dimension $D = 512$ under the v-prediction. The semigroup objective yields the most consistent training behavior and the highest sample quality in our experiments (Fig. 3). While Eulerian alone is unstable and produces poor one-step samples, applying adaptive loss weighting substantially mitigates target variance and enables it to better capture the data distribution.

Table 1. Promoter DNA sequence generation results on the test set averaged over three runs. Fisher FM is reproduced in our setup; the remaining baseline results follow Davis et al. (2024).

Method		NFE	MSE (↓)	k-mer corr. (↑)
Dirichlet FM		100	$0.034_{\pm 0.004}$	N/A
Fisher FM		100	$0.030_{\pm 0.001}$	$0.96_{\pm 0.01}$
Eulerian RMF	x_1-pred	1	$0.030_{\pm 0.000}$	$0.96_{\pm 0.01}$
	v-pred	1	$0.031_{\pm 0.001}$	$0.96_{\pm 0.00}$
Lagrangian RMF	x_1-pred	1	$0.027_{\pm 0.001}$	$0.88_{\pm 0.00}$
	v-pred	1	$0.027_{\pm 0.001}$	$0.85_{\pm 0.01}$
Semigroup RMF	x_1-pred	1	$0.030_{\pm 0.001}$	$0.84_{\pm 0.03}$
	v-pred	1	$0.030_{\pm 0.001}$	$0.93_{\pm 0.02}$

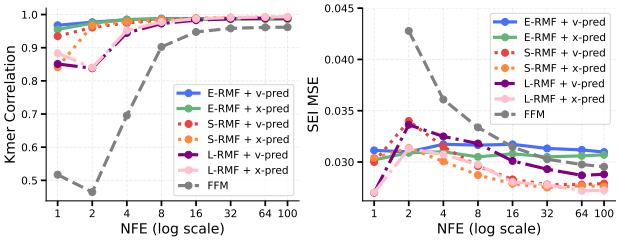


Figure 4. Performance vs. NFE. RMF variants outperform Fisher FM (FFM) in k-mer correlation ($k = 6$) and MSE. RMF achieves high accuracy at 1 NFE, whereas FFM requires ≥ 32 steps for comparable performance.

4.2. Promoter DNA Design

We evaluate RMF for generating human promoter sequences of length 1,024 conditioned on target transcription signal profiles. We train on 88,470 sequences from FANTOM5 (Hon et al., 2017) and compare against Dirichlet FM (Stark et al., 2024) and Fisher FM (Davis et al., 2024). We report (i) the mean squared error (MSE) between the signal profiles predicted by a pre-trained Sei model (Chen et al., 2022) for the generated and target human promoter sequences, and (ii) k-mer correlation ($k = 6$) between generated and real sequence distributions at different numbers of function evaluations (NFEs). A full task description and evaluation details are provided in Apps. E and F.

Results. Table 1 shows that all RMF variants using one-step generation match the 100-step performance of Fisher FM while consistently outperforming Dirichlet FM. Performance remains robust across NFEs (Fig. 4). Furthermore, x_1-prediction matches v-prediction across objectives; we provide a detailed analysis of its advantages in App. G.3.

Reward guidance results. We extend the DNA design

Table 2. Reward-guided inference improves alignment between Sei signal profiles of generated and reference promoter sequences, measured by mean squared error ± standard deviation across 60 batches of 128 samples. Lower MSE is better. "—" is no guidance.

NFE	—	$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$
1	$0.033_{\pm 0.015}$	$0.033_{\pm 0.015}$	$0.025_{\pm 0.011}$
5	$0.031_{\pm 0.014}$	$0.017_{\pm 0.009}$	$0.013_{\pm 0.005}$
10	$0.031_{\pm 0.013}$	$0.008_{\pm 0.002}$	$0.008_{\pm 0.003}$

setting to evaluate our reward guidance approach on the following task: given a target regulatory behavior, can we refine samples to better match it? For every inference step, we compute the MSE between the Sei profiles of the x_1-prediction and target sequence, and use its gradient to update samples according to Eq. (19). In Table 2, we report the MSE of the final sequence profiles to the test targets in Hon et al. (2017). Reward-guided sampling consistently reduces MSE compared to unguided generation, even for one-step sampling. Furthermore, naive guidance on the current state x_t does worse than using the x_1-prediction. We show an ablation over guidance scales and reward details in App. E.2.

4.3. Protein Backbone Design

Finally, we evaluate our method on unconditional *de novo* protein backbone generation. We train on the SCOPe dataset (Chandonia et al., 2022) and benchmark three state-of-the-art baselines: GENIE (Lin & AlQuraishi, 2023), FrameDiff (Yim et al., 2023b), and FrameFlow (Yim et al., 2023a). We report standard metrics: *designability*, *novelty*, and *diversity*. For additional details, see Apps. E and F.

We find that using the semigroup RMF objective with x_1-prediction leads to the most stable training dynamics of the protein backbone generative model. We attribute this to the fact that the semigroup objective minimizes the number of differential operations required for the optimization. Furthermore, the x_1-prediction scalability agrees with the results in Sec. 4.1 and allows for out of the box usage of the standard protein backbone architectures used in prior work (Yim et al., 2023a;b; Bose et al., 2023).

Main results. In Table 3, we report results across different numbers of function evaluations (NFE). Our method maintains high designability in the few-step regime: at 5 steps, it achieves 82% designability (where designability is defined as the percentage of structures that have an RMSD <2 Åwhen refolded using ESMFold (Lin et al., 2023)), while baselines drop sharply (e.g., 9% for FrameDiff and 4% for FrameFlow). Even in the extreme one-step setting, our method still generates 35% designable samples, whereas baseline methods completely collapse. Since we optimize for designability, we observe a mild reduction in novelty, but maintain competitive diversity relative to baselines.

Effect of model size. In Fig. 5a, we compare the RMSD of refolded structures (`scRMSD`) for models of different sizes. We observe that increasing the model size significantly improves the designability in the low-step regime, reducing `scRMSD` at 1–10 sampling steps. For 100 function evaluations, the gap in performance diminishes and all models perform on-par.

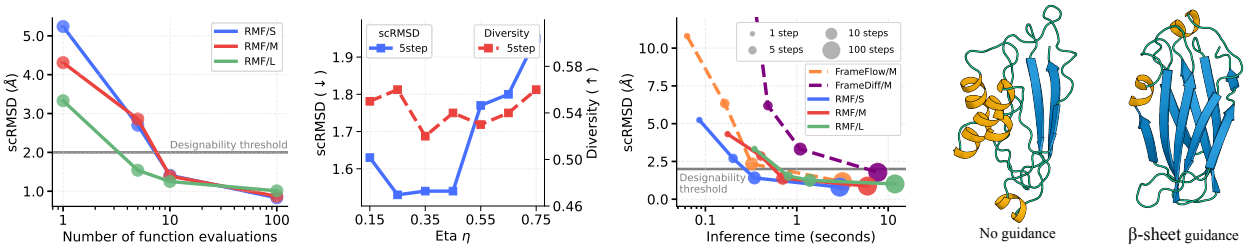
Table 3. Protein backbone generation results. Rows are grouped by inference regime (NFE). We highlight in **bold** the best designability ($<$ 2Å) within each regime, our primary metric. We mark not applicable (N/A) when no designable samples are generated or when metrics are not reported in prior work.

Model	NFE	Designability		Diversity		Novelty
		<2Å (↑)	`scRMSD` (↓)	Max. Cluster (↑)	Pairwise `scTM` (↓)	Max. `scTM` (↓)
Many-step regime (NFE $\geq$ 100)						
GENIE	1000	0.22	N/A	0.76	N/A	0.54
	750	0.11	N/A	0.79	N/A	0.51
	500	0.00	N/A	N/A	N/A	N/A
FrameDiff	500	0.80	1.63	0.36	0.34	0.68
	100	0.74	1.78	1.74	0.34	0.51
FrameFlow	100	0.93	1.16	0.41	0.30	0.77
RMF (Ours)	100	**0.94**	**1.01**	0.55	0.27	0.89
Moderate regime (10 $\leq$ NFE $<$ 100)						
FrameDiff	10	0.47	3.32	0.42	0.28	0.52
FrameFlow	10	0.61	2.34	0.54	0.26	0.67
RMF (Ours)	10	**0.87**	**1.25**	0.55	0.27	0.87
Few-step regime (NFE $<$ 10)						
FrameFlow	5	0.04	6.53	0.68	0.22	0.74
FrameDiff	5	0.09	6.19	0.54	0.24	0.96
RMF (Ours)	5	**0.82**	**1.54**	0.54	0.27	0.85
FrameFlow	2	0.00	N/A	N/A	N/A	N/A
RMF (Ours)	1	**0.35**	3.33	0.60	0.24	0.76

Effect of inference techniques. We apply a low-noise inference scheme at sampling time, a common technique in protein backbone generation (Yim et al., 2023a; Bose et al., 2023; Xie et al., 2025). We control the amount of injected noise using η, where $\eta = 1$ corresponds to full noise and $\eta = 0$ to no noise. In Fig. 5b, setting η to intermediate values ($\eta \approx 0.25$–0.45) gave the best outcomes of low `scRMSD` and high diversity. We therefore used $\eta = 0.45$ in all following protein experiments. Additional details of the inference scheme are provided in App. F.2.

Faster inference. In Fig. 5c, we compare `scRMSD` against wall-clock time for different NFEs. Notably, a medium-sized RMF consistently outperforms prior methods across all step counts: it obtains better designability under a comparable compute budget and reaches the designability threshold faster than FrameDiff and FrameFlow. The large version of RMF has a larger per-step cost, although it achieves the best designability in the low-step regime.

Reward guidance results. In protein design settings, it may be important to control the composition of structural motifs, as these motifs can influence protein function. Furthermore, generative models tend to oversample α-helices (Faltings et al., 2025) and undersample other structural motifs (Lu et al., 2025). We consider secondary structure optimization



(a) Model size *(b)* Inference technique *(c)* Inference steps *(d)* Reward guidance

Figure 5. Quantitative results of protein backbone design. **(a)** We report the designability performance against the model size with diverse inference steps, where the performance scales proportionally with the model size. **(b)** For various eta values on the inference technique, a value between 0.25 and 0.45 shows the best performance. **(c)** When compared to baselines, RMF shows better generation performance in terms of designability for identical inference steps. **(d)** Example of protein generation from the same initial state using the base model (left) and reward-guided inference (right) toward higher β-sheet content with 10 NFE. β-sheets are in yellow, α-helices are in pink.

Table 4. Reward guidance increases the percentage of amino acids assigned to a target secondary structure composition. In each setting, 100 sequences of length 128 were generated using the RMF/M model. ζ_t set to "—" corresponds to no guidance.

Reward	NFE	ζ_t	Mean (↑)	Top-10 mean (↑)	Max (↑)	Frac. improved (↑)
`β-sheet`	5	—	$0.18_{\pm 0.12}$	$0.41_{\pm 0.06}$	0.51	—
		$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$0.18_{\pm 0.12}$	$0.42_{\pm 0.05}$	0.51	0.27
		$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$	$0.22_{\pm 0.13}$	$0.46_{\pm 0.05}$	0.58	0.60
	10	—	$0.20_{\pm 0.13}$	$0.45_{\pm 0.04}$	0.52	—
		$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$0.21_{\pm 0.13}$	$0.45_{\pm 0.04}$	0.52	0.24
		$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$	$0.24_{\pm 0.15}$	$0.51_{\pm 0.04}$	0.60	0.56
`α-helix`	5	—	$0.29_{\pm 0.20}$	$0.68_{\pm 0.08}$	0.80	—
		$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$0.29_{\pm 0.20}$	$0.68_{\pm 0.08}$	0.80	0.27
		$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$	$0.33_{\pm 0.22}$	$0.73_{\pm 0.04}$	0.80	0.58
	10	—	$0.30_{\pm 0.20}$	$0.70_{\pm 0.07}$	0.80	—
		$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$0.30_{\pm 0.20}$	$0.71_{\pm 0.08}$	0.80	0.27
		$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$	$0.36_{\pm 0.22}$	$0.76_{\pm 0.04}$	0.80	0.64

as an illustrative task for controlling protein design (Hartman et al., 2025); we developed a differentiable secondary structure reward to guide towards higher compositions of β-sheets or α-helices using x_1-prediction (see App. E.4), and evaluated final structures using DSSP (Kabsch & Sander, 1983). Table 4 shows that reward guidance using $\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$ increases composition of both target secondary structures, while using $\mathrm{Proj}_{x_t}(\nabla r(x_t))$ is similar to not doing guidance. A visual example is shown in Fig. 5d.

5. Related Work

Consistency models and flow-map learning. Consistency models (CMs) reduce inference costs by mapping noise directly to data (Song et al., 2023; Song & Dhariwal, 2023; Lu & Song, 2024; Geng et al., 2024), but often lack explicit finite-time transport. Recent Euclidean flow map learning regresses *average velocities* for integration-free generation between arbitrary time points (Boffi et al., 2025; Geng et al., 2025; Guo et al., 2025; Zhou et al., 2025), which enhances multi-step robustness (Sabour et al., 2025b). However, these formulations assume flat geometry and do not naturally extend to Riemannian manifolds.

Generative modeling on manifolds. Manifold-aware frameworks based on normalizing flows (Lou et al., 2020; Mathieu & Nickel, 2020), diffusion (Huang et al., 2022; De Bortoli et al., 2022), and flow matching (Chen & Lipman, 2023) have been shown to work in scientific domains such as biological sequence generation (Stark et al., 2024; Davis et al., 2024; Yim et al., 2023a;b; Bose et al., 2023). However, these methods often require computationally expensive numerical ODE/SDE integration during inference, which can be prohibitive in high dimensions.

Few-step generation on manifolds. Riemannian Consistency Models (Cheng et al., 2025) adapt CM objectives to manifolds but lack explicit flow map modeling. Concurrently, Generalized Flow Map (GFM) (Davis et al., 2025) extends flow map learning beyond Euclidean spaces, but uses suboptimal design choices for the training objective and flow map parameterization, which increase the computational cost (by requiring extra backpropagations) and compromise training stability. In contrast, our RMF treats JVP-related terms as fixed targets, bypassing higher-order derivatives and enabling scalable, stable learning for high-dimensional scientific tasks. We write a formal theoretical connection between our identities and the GFM formulation in App. C.2 and provide an empirical comparison in App. D.

6. Conclusion

In this work, we introduce RMF, a principled framework for one- and few-step generative modeling on manifolds, informed by a systematic study of intrinsic manifold identities, parameterizations, and stabilization techniques. We find that aligning all these three components is important for scaling few-step generation to the high-dimensional geometries present in biological domains. Empirically, RMF matches the performance of strong multi-step baselines with up to $10\times$ fewer function evaluations, enabling efficient reward-guided exploration. Overall, our work establishes the theoretical foundation and practical guidelines for fast sampling on manifolds, which we envision will facilitate AI-aided discoveries in natural sciences.

Impact Statement

This paper presents Riemannian MeanFlow, a principled framework for efficient generative modeling on manifolds. Our work contributes to the advancement of machine learning with specific applications in scientific domains such as protein backbone generation and DNA sequence design. While the advancement of these generative capabilities offers significant benefits for medicine and biotechnology, we recognize that the increased efficiency of biological design tools underscores the importance of maintaining rigorous ethical oversight and bio-security screening protocols. This work establishes a principled foundation for fast sampling on manifolds, which may lead to broader implications in scientific design pipelines where generative models serve as critical proposal mechanisms for property-guided optimization.

References

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. 1

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000. 34

Boffi, N. M., Albergo, M. S., and Vanden-Eijnden, E. How to build a consistency model: Learning flow maps via self-distillation. *arXiv preprint arXiv:2505.18825*, 2025. 2, 8

Bose, A. J., Akhound-Sadegh, T., Huguet, G., Fatras, K., Rector-Brooks, J., Liu, C.-H., Nica, A. C., Korablyov, M., Bronstein, M., and Tong, A. Se (3)-stochastic flow matching for protein backbone generation. *arXiv preprint arXiv:2310.02391*, 2023. 1, 5, 7, 8, 35

Burley, S. K., Bhikadiya, C., Bi, C., Bittrich, S., Chen, L., Crichlow, G. V., Christie, C. H., Dalenberg, K., Di Costanzo, L., Duarte, J. M., et al. Rcsb protein data bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic acids research*, 49(D1):D437–D451, 2021. 34

Chandonia, J.-M., Guan, L., Lin, S., Yu, C., Fox, N. K., and Brenner, S. E. Scope: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic acids research*, 50(D1):D553–D559, 2022. 7, 34

Chen, K. M., Wong, A. K., Troyanskaya, O. G., and Zhou, J. A sequence-based global map of regulatory activity for deciphering human genetics. *Nature genetics*, 54(7): 940–949, 2022. 6

Chen, R. T. and Lipman, Y. Flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023. 1, 8, 22, 28

Cheng, C., Li, J., Peng, J., and Liu, G. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024. 1

Cheng, C., Wang, Y., Chen, Y., Zhou, X., Zheng, N., and Liu, G. Riemannian consistency model. *arXiv preprint arXiv:2510.00983*, 2025. 2, 8

Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I., Courbet, A., de Haas, R. J., Bethel, N., et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378 (6615):49–56, 2022. 32

Davis, O., Kessler, S., Petrache, M., Ceylan, İ. İ., Bronstein, M., and Bose, A. J. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084, 2024. 1, 2, 6, 8, 31, 34

Davis, O., Albergo, M. S., Boffi, N. M., Bronstein, M. M., and Bose, A. J. Generalised flow maps for few-step generative modelling on riemannian manifolds. *arXiv preprint arXiv:2510.21608*, 2025. 2, 8, 26, 28, 29

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. Riemannian score-based generative modelling. *Advances in neural information processing systems*, 35:2406–2422, 2022. 1, 8

Faltings, F., Stark, H., Jaakkola, T., and Barzilay, R. Protein fid: Improved evaluation of protein structure generative models. *arXiv preprint arXiv:2505.08041*, 2025. 7

Geffner, T., Didi, K., Cao, Z., Reidenbach, D., Zhang, Z., Dallago, C., Kucukbenli, E., Kreis, K., and Vahdat, A. La-proteina: Atomistic protein generation via partially latent flow matching. *arXiv preprint arXiv:2507.09466*, 2025a. 32

Geffner, T., Didi, K., Zhang, Z., Reidenbach, D., Cao, Z., Yim, J., Geiger, M., Dallago, C., Kucukbenli, E., Vahdat, A., et al. Proteina: Scaling flow-based protein structure generative models. *arXiv preprint arXiv:2503.00710*, 2025b. 35

Geng, Z., Pokle, A., Luo, W., Lin, J., and Kolter, J. Z. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024. 8

Geng, Z., Deng, M., Bai, X., Kolter, J. Z., and He, K. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025. 2, 4, 5, 8, 23, 24, 26, 34, 38

Guo, Y., Wang, W., Yuan, Z., Cao, R., Chen, K., Chen, Z., Huo, Y., Zhang, Y., Wang, Y., Liu, S., et al. Splitmeanflow: Interval splitting consistency in few-step generative modeling. *arXiv preprint arXiv:2507.16884*, 2025. 2, 8

Haberle, V. and Stark, A. Eukaryotic core promoters and the functional basis of transcription initiation. *Nature reviews Molecular cell biology*, 19(10):621–637, 2018. 31

Han, X.-Q., Guo, P.-J., Gao, Z.-F., Sun, H., and Lu, Z.-Y. Invdesflow-al: active learning-based workflow for inverse design of functional materials. *npj Computational Materials*, 2025. 2

Hartman, E., Wallin, J., Malmström, J., and Olsson, J. Controllable protein design through feynman-kac steering. *arXiv preprint arXiv:2511.09216*, 2025. 8

Hasan, M., Ohanesian, V., Gazizov, A., Bengio, Y., Aspuru-Guzik, A., Bondesan, R., Skreta, M., and Neklyudov, K. Discrete feynman-kac correctors. *arXiv preprint arXiv:2601.10403*, 2026. 5

Herbert, A. and Sternberg, M. Maxcluster: a tool for protein structure comparison and clustering, 2008. 32

Hon, C.-C., Ramilowski, J. A., Harshbarger, J., Bertin, N., Rackham, O. J., Gough, J., Denisenko, E., Schmeier, S., Poulsen, T. M., Severin, J., et al. An atlas of human long non-coding rnas with accurate 5′ ends. *Nature*, 543 (7644):199–204, 2017. 6, 7, 34

Huang, C.-W., Aghajohari, M., Bose, J., Panangaden, P., and Courville, A. C. Riemannian diffusion models. *Advances in Neural Information Processing Systems*, 35: 2750–2761, 2022. 1, 8

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. Torsional diffusion for molecular conformer generation. *Advances in neural information processing systems*, 35: 24240–24253, 2022. 1

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021. 1

Kabsch, W. and Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637, 1983. 8, 32, 33

Li, T. and He, K. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025. 5

Lin, Y. and AlQuraishi, M. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds. *arXiv preprint arXiv:2301.12485*, 2023. 7, 34

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023. 7, 32

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1, 2

Lou, A., Lim, D., Katsman, I., Huang, L., Jiang, Q., Lim, S. N., and De Sa, C. M. Neural manifold ordinary differential equations. *Advances in Neural Information Processing Systems*, 33:17548–17558, 2020. 8

Lu, C. and Song, Y. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024. 5, 8, 36

Lu, T., Liu, M., Chen, Y., Kim, J., and Huang, P.-S. Assessing generative model coverage of protein structures with shapes. *bioRxiv*, 2025. 7

Mathieu, E. and Nickel, M. Riemannian continuous normalizing flows. *Advances in neural information processing systems*, 33:2503–2515, 2020. 8, 28

Pacesa, M., Nickel, L., Schellhaas, C., Schmidt, J., Pyatova, E., Kissling, L., Barendse, P., Choudhury, J., Kapoor, S., Alcaraz-Serna, A., et al. Bindcraft: one-shot design of functional protein binders. *bioRxiv*, pp. 2024–09, 2024. 2

Rehman, D., Akhound-Sadegh, T., Gazizov, A., Bengio, Y., and Tong, A. Falcon: Few-step accurate likelihoods for continuous flows. *arXiv preprint arXiv:2512.09914*, 2025. 28

Sabour, A., Albergo, M. S., Domingo-Enrich, C., Boffi, N. M., Fidler, S., Kreis, K., and Vanden-Eijnden, E. Test-time scaling of diffusions with flow maps. *arXiv preprint arXiv:2511.22688*, 2025a. 5

Sabour, A., Fidler, S., and Kreis, K. Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*, 2025b. 8

Skreta, M., Akhound-Sadegh, T., Ohanesian, V., Bondesan, R., Aspuru-Guzik, A., Doucet, A., Brekelmans, R.,

Tong, A., and Neklyudov, K. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Vhc0KrcqWu. 5

Song, Y. and Dhariwal, P. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023. 2, 5, 8

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1, 2

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html. 2, 8, 24

Stark, H., Jing, B., Wang, C., Corso, G., Berger, B., Barzilay, R., and Jaakkola, T. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024. 1, 6, 8, 31, 34

Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., and Jaakkola, T. S. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *ICLR*, 2023. 32

Van Kempen, M., Kim, S. S., Tumescheit, C., Mirdita, M., Lee, J., Gilchrist, C. L., Söding, J., and Steinegger, M. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024. 32

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *BioRxiv*, pp. 2022–12, 2022. 1

Xie, L., Zhang, H., Wang, Z., Tansey, W., and Zhou, M. Distilled protein backbone generation. *arXiv preprint arXiv:2510.03095*, 2025. 7, 35

Yang, K., Jin, W., Swanson, K., Barzilay, R., and Jaakkola, T. Improving molecular design by stochastic iterative target augmentation. In *International Conference on Machine Learning*, pp. 10716–10726. PMLR, 2020. 2

Yim, J., Campbell, A., Foong, A. Y., Gastegger, M., Jiménez-Luna, J., Lewis, S., Satorras, V. G., Veeling, B. S., Barzilay, R., Jaakkola, T., et al. Fast protein backbone generation with se (3) flow matching. *Proceedings of Neural Information Processing Systems Workshop: Machine Learning in Structural Biology Workshop*, 2023a. 1, 2, 5, 7, 8, 32, 34, 35

Yim, J., Trippe, B. L., De Bortoli, V., Mathieu, E., Doucet, A., Barzilay, R., and Jaakkola, T. SE(3) diffusion model with application to protein backbone generation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40001–40039. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/yim23a.html. 1, 5, 7, 8, 32, 34, 35

Zhang, Y. and Skolnick, J. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005. 32

Zhou, L., Parger, M., Haque, A., and Song, J. Terminal velocity matching. *arXiv preprint arXiv:2511.19797*, 2025. 2, 8

A. Tutorial on Riemannian Manifolds

This appendix provides a self-contained introduction to Riemannian geometry, covering the essential concepts needed to understand flow-based generative models on manifolds. We aim to build intuition while maintaining mathematical rigor.

A.1. Smooth Manifolds

A **smooth manifold** $\mathcal{M}$ of dimension d is a topological space that locally resembles Euclidean space $\mathbb{R}^d$. Formally, every point $x \in \mathcal{M}$ has a neighborhood that can be mapped smoothly and bijectively to an open subset of $\mathbb{R}^d$. These local maps are called *charts*, and a collection of charts covering $\mathcal{M}$ is called an *atlas*.

Example A.1 (The d-dimensional sphere). The unit sphere $\mathbb{S}^d = \{x \in \mathbb{R}^{d+1} : \|x\|_2 = 1\}$ is a d-dimensional manifold embedded in $\mathbb{R}^{d+1}$. Although it lives in $(d+1)$-dimensional space, it has only d degrees of freedom.

Example A.2 (The probability simplex). The probability simplex $\Delta^{d-1} = \{p \in \mathbb{R}^d : p_i \geq 0, \sum_i p_i = 1\}$ represents discrete probability distributions over d categories. This is the natural space for DNA sequences (with $d = 4$ for nucleotides A, C, G, T).

Example A.3 (Special Euclidean group $\mathrm{SE}(3)$). The group $\mathrm{SE}(3) = \mathrm{SO}(3) \ltimes \mathbb{R}^3$ describes rigid body transformations in 3D space, consisting of rotations and translations. For protein backbone generation, we work on the product manifold $\mathrm{SE}(3)^N$ representing N residue frames.

A.2. Tangent Spaces and Tangent Bundles

At each point $x \in \mathcal{M}$, the **tangent space** $T_x\mathcal{M}$ is a vector space of the same dimension as $\mathcal{M}$. Intuitively, $T_x\mathcal{M}$ collects all possible infinitesimal directions in which one can move on the manifold starting from x.

Definition A.1 (Tangent vector)**.** A tangent vector $v \in T_x\mathcal{M}$ can be defined as the velocity of a smooth curve $\gamma : (-\epsilon, \epsilon) \to \mathcal{M}$ with $\gamma(0) = x$:

$$v = \dot{\gamma}(0) = \left.\frac{d\gamma(t)}{dt}\right|_{t=0}. \tag{20}$$

Under this definition, a tangent vector captures only the *first-order behavior* of a curve at the point x; different curves may represent the same tangent vector as long as they induce the same instantaneous change at $t = 0$.

Tangent vectors and differentials. Tangent vectors describe infinitesimal changes at a point. Differentials describe how such changes propagate through functions. Concretely, let $f : \mathcal{M} \to \mathcal{N}$ be a smooth map, and let $v \in T_x\mathcal{M}$ be represented by a curve $\gamma(t)$ with $\dot{\gamma}(0) = v$. Composing the curve with f yields a new curve $f \circ \gamma(t)$ in $\mathcal{N}$, whose velocity at $t = 0$ defines the induced change in the output:

$$df_x(v) := \left.\frac{d}{dt} f(\gamma(t))\right|_{t=0} \in T_{f(x)}\mathcal{N}. \tag{21}$$

Thus, a tangent vector specifies *what infinitesimal change is applied at the input*, while the differential specifies *how the function responds to that change*.

For manifolds embedded in $\mathbb{R}^D$, the tangent space can often be realized as a linear subspace of the ambient space. For example, for the sphere $\mathbb{S}^d \subset \mathbb{R}^{d+1}$, the tangent space at x consists of all vectors orthogonal to x:

$$T_x\mathbb{S}^d = \{v \in \mathbb{R}^{d+1} : \langle v, x\rangle = 0\}. \tag{22}$$

The collection of all tangent spaces forms the **tangent bundle** as follows:

$$T\mathcal{M} = \bigsqcup_{x \in \mathcal{M}} T_x\mathcal{M}, \tag{23}$$

which itself has the structure of a smooth manifold of dimension $2d$. Points in $T\mathcal{M}$ are pairs (x, v) consisting of a base point and a tangent vector attached to it.

Definition A.2 (Projection onto tangent space)**.** For embedded manifolds, ambient vectors in $\mathbb{R}^D$ can be mapped to the tangent space via orthogonal projection. The projection $\mathrm{Proj}_x : \mathbb{R}^D \to T_x\mathcal{M}$ extracts the tangential component of a vector. For the sphere, it is given by

$$\mathrm{Proj}_x(v) = v - \langle v, x\rangle x. \tag{24}$$

A.3. Riemannian Metrics

A **Riemannian metric** g assigns to each point $x \in \mathcal{M}$ an inner product $\langle\cdot,\cdot\rangle_x$ on the tangent space $T_x\mathcal{M}$, varying smoothly with x. This metric allows us to measure lengths, angles, and volumes on the manifold.

Definition A.3 (Riemannian manifold)**.** A Riemannian manifold $(\mathcal{M}, g)$ is a smooth manifold $\mathcal{M}$ equipped with a Riemannian metric g. The induced norm on $T_x\mathcal{M}$ is $\|v\|_x = \sqrt{\langle v, v\rangle_x}$.

Example A.4 (Euclidean metric)*.* For $\mathbb{R}^d$, the standard Euclidean metric is $\langle u, v\rangle_x = u^\top v$, independent of x.

Example A.5 (Induced metric on spheres)*.* For $\mathbb{S}^d \subset \mathbb{R}^{d+1}$, the induced metric is simply the restriction of the Euclidean inner product: $\langle u, v\rangle_x = u^\top v$ for $u, v \in T_x\mathbb{S}^d$.

Example A.6 (Fisher-Rao metric on the simplex)*.* On the probability simplex Δ^{d-1}, the Fisher-Rao metric is:

$$\langle u, v\rangle_p = \sum_{i=1}^{d} \frac{u_i v_i}{p_i}, \tag{25}$$

where $u, v \in T_p\Delta^{d-1}$ satisfy $\sum_i u_i = \sum_i v_i = 0$. This metric is natural for statistical manifolds and is used in DNA sequence modeling.

A.4. Geodesics

A **geodesic** is a curve that locally minimizes distance on a Riemannian manifold—the generalization of straight lines to curved spaces.

Definition A.4 (Geodesic)**.** A smooth curve $\gamma : [0, 1] \to \mathcal{M}$ is a geodesic if it satisfies the geodesic equation:

$$\nabla_{\dot\gamma}\dot\gamma = 0, \tag{26}$$

where ∇ is the Levi-Civita connection (defined below). Intuitively, this means the velocity vector undergoes parallel transport along the curve.

Example A.7 (Geodesics on the sphere)*.* On the unit sphere $\mathbb{S}^d$, geodesics are great circles. The geodesic from x to y (assuming they are not antipodal) lies in the plane spanned by x, y, and the origin.

The **geodesic distance** $d_g(x, y)$ between two points is the length of the shortest geodesic connecting them:

$$d_g(x, y) = \inf_\gamma \int_0^1 \|\dot\gamma(t)\|_{\gamma(t)}\, dt, \tag{27}$$

where the infimum is over all smooth curves γ with $\gamma(0) = x$ and $\gamma(1) = y$.

A.5. Exponential and Logarithmic Maps

The exponential and logarithmic maps provide a way to move between the tangent space and the manifold, which is essential for defining interpolations and flow maps.

Definition A.5 (Exponential map)**.** The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ maps a tangent vector v to the endpoint of the geodesic starting at x with initial velocity v, evaluated at time $t = 1$:

$$\exp_x(v) = \gamma(1), \quad \text{where } \gamma(0) = x,\ \dot{\gamma}(0) = v. \tag{28}$$

More generally, $\exp_x(tv)$ traces out the geodesic for $t \in [0, 1]$.

Definition A.6 (Logarithmic map)**.** The logarithmic map $\log_x : \mathcal{M} \to T_x\mathcal{M}$ is the (local) inverse of the exponential map:

$$\log_x(y) = v \quad \text{such that} \quad \exp_x(v) = y. \tag{29}$$

The logarithmic map exists and is unique in a neighborhood of x (within the *injectivity radius*).

Proposition A.1 (Properties)**.** *The exponential and logarithmic maps satisfy:*

1. $\exp_x(\log_x(y)) = y$ *for* y *sufficiently close to* x
2. $\log_x(\exp_x(v)) = v$ *for* $\|v\|_x$ *sufficiently small*
3. $\|\log_x(y)\|_x = d_g(x, y)$ *(the norm of the log equals geodesic distance)*
4. $\exp_x(0) = x$ *and* $\log_x(x) = 0$

Example A.8 (Exponential and logarithmic maps on $\mathbb{S}^d$). For the unit sphere, let $x \in \mathbb{S}^d$ and $v \in T_x\mathbb{S}^d$ with $\|v\| \neq 0$:

$$\exp_x(v) = \cos(\|v\|)x + \sin(\|v\|)\frac{v}{\|v\|}, \tag{30}$$

$$\log_x(y) = \frac{\theta}{\sin\theta}(y - \cos\theta \cdot x), \quad \theta = \arccos(\langle x, y\rangle). \tag{31}$$

When $v = 0$, we have $\exp_x(0) = x$. When $y = x$, we have $\log_x(x) = 0$.

A.6. Geodesic Interpolation

Using the exponential and logarithmic maps, we can define geodesic interpolation between two points, which is fundamental for flow matching on manifolds.

Definition A.7 (Geodesic interpolant)**.** Given $x_0, x_1 \in \mathcal{M}$, the geodesic interpolant at time $t \in [0, 1]$ is:

$$x_t = \exp_{x_0}(t \log_{x_0}(x_1)). \tag{32}$$

This traces out the geodesic from x_0 to x_1 with constant speed.

The velocity of this geodesic is constant along the path:

$$\dot{x}_t = \frac{d}{dt}x_t = P_{x_0 \to x_t}(\log_{x_0}(x_1)), \tag{33}$$

where $P_{x_0 \to x_t}$ denotes parallel transport (defined below).

A.7. Covariant Derivatives and Connections

To rigorously formalize the geometric notions introduced above, we require the concept of a *connection*. In Euclidean space, geometric quantities such as velocity and acceleration along a curve are defined using ordinary derivatives: acceleration is simply the derivative of velocity. On a manifold, however, this notion breaks down. The velocity $\dot{\gamma}(t)$ of a curve $\gamma(t)$ lies in the tangent space $T_{\gamma(t)}\mathcal{M}$, which varies with t, so ordinary differentiation would attempt to compare vectors living in different tangent spaces.

A **connection** resolves this issue by prescribing a consistent way to differentiate vector fields on a manifold, thereby enabling meaningful notions of acceleration and variation of tangent vectors.

A.7.1. Vector Fields as Differential Operators

Before introducing connections, it is useful to recall a fundamental viewpoint from differential geometry: *vector fields act as differential operators on functions*. This operator perspective will be central to the definition of covariant derivatives.

Definition A.8 (Directional derivative along a vector field)**.** Let X be a vector field on $\mathcal{M}$ and $f : \mathcal{M} \to \mathbb{R}$ a smooth function. The **directional derivative** of f along X is the function $Xf : \mathcal{M} \to \mathbb{R}$ defined by

$$(Xf)(x) = df_x(X(x)), \tag{34}$$

where $df_x : T_x\mathcal{M} \to \mathbb{R}$ denotes the differential of f at x.

Equivalently, if $\gamma(t)$ is any curve satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = X(x)$, then

$$(Xf)(x) = \left.\frac{d}{dt}\right|_{t=0} f(\gamma(t)), \tag{35}$$

which represents the rate of change of f in the direction $X(x)$.

Proposition A.2 (Properties of directional derivatives)**.** *For vector fields X, Y, functions f, g, and constant c:*

1. ***Linearity in** X:* $(X + Y)f = Xf + Yf$ *and* $(cX)f = c(Xf)$.
2. ***Linearity in** f:* $X(f + g) = Xf + Xg$ *and* $X(cf) = c(Xf)$.
3. ***Leibniz rule:*** $X(fg) = (Xf)g + f(Xg)$.
4. ***Constants:*** $X(c) = 0$ *for any constant function c.*

These properties show that X acts as a *derivation* on the algebra of smooth functions. In fact, one may equivalently define vector fields as derivations satisfying these properties.

A.7.2. Affine Connections

Directional derivatives allow us to differentiate scalar functions, but do not provide a way to differentiate vector fields themselves. An affine connection fills this gap.

Definition A.9 (Affine connection)**.** An **affine connection** ∇ on a smooth manifold $\mathcal{M}$ assigns to each pair of vector fields X, Y a new vector field $\nabla_X Y$, called the *covariant derivative of Y in the direction X*, such that for all vector fields X, Y, Z and smooth functions f, g:

1. **Linearity in X:** $\nabla_{fX+gZ}Y = f\nabla_X Y + g\nabla_Z Y$.
2. **Linearity in Y:** $\nabla_X(Y + Z) = \nabla_X Y + \nabla_X Z$.
3. **Leibniz rule:** $\nabla_X(fY) = (Xf)Y + f\nabla_X Y$.

The Leibniz rule reflects the operator nature of ∇_X: when differentiating fY, the derivative acts both on the scalar coefficient f and on the vector field Y itself.

Intuitively, $\nabla_X Y$ measures how the vector field Y changes as one moves in the direction X, with the connection specifying how to compare vectors in neighboring tangent spaces.

A.7.3. Metric Compatibility

On a Riemannian manifold $(\mathcal{M}, g)$, it is natural to require the connection to interact consistently with the metric structure.

Definition A.10 (Metric compatibility)**.** A connection ∇ is **metric-compatible** if for all vector fields X, Y, Z,

$$X\langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle. \tag{36}$$

This condition is a product rule for the Riemannian inner product, ensuring that differentiation commutes with taking inner products. It is a local compatibility requirement between the connection and the metric, and by itself does not uniquely determine the connection.

A.7.4. The Levi–Civita Connection

Metric compatibility alone does not specify a unique connection. An additional natural requirement is the absence of torsion, which enforces symmetry of differentiation and generalizes the commutativity of partial derivatives in Euclidean space.

A classical result in Riemannian geometry shows that these two conditions together uniquely determine the connection.

Theorem A.9 (Fundamental theorem of Riemannian geometry)**.** *On any Riemannian manifold $(\mathcal{M}, g)$, there exists a unique affine connection ∇, called the **Levi–Civita connection**, satisfying:*

1. ***Torsion-free:***
$$\nabla_X Y - \nabla_Y X = [X, Y],$$
2. ***Metric-compatible:***
$$X\langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle.$$

Intuition. The Levi–Civita connection is the canonical choice of differentiation that depends only on the Riemannian metric. It generalizes ordinary derivatives in Euclidean space and provides a consistent notion of differentiation intrinsic to the manifold geometry.

A.7.5. Covariant Derivative Along a Curve

While an affine connection defines differentiation between vector fields, in dynamical and flow-based settings we primarily require differentiation *along a given trajectory* on the manifold.

Definition A.11 (Covariant derivative along a curve)**.** Let $\gamma : [0, 1] \to \mathcal{M}$ be a smooth curve and $V(t) \in T_{\gamma(t)}\mathcal{M}$ a vector field along γ. The **covariant derivative** of V along γ is defined as

$$D_t V := \nabla_{\dot\gamma(t)} V. \tag{37}$$

The operator D_t provides an intrinsic notion of time differentiation for vector-valued quantities whose ambient space varies along the curve. In particular, $D_t V$ can be interpreted as the *acceleration* of $V(t)$ along $\gamma(t)$.

Importantly, $D_t V$ depends only on the values of V along γ, and not on how V is extended to a neighborhood of the curve.

For embedded submanifolds $\mathcal{M} \subset \mathbb{R}^D$, the covariant derivative admits a simple expression:

$$D_t V = \mathrm{Proj}_{\gamma(t)}\left(\frac{dV}{dt}\right), \tag{38}$$

where $\mathrm{Proj}_{\gamma(t)}$ denotes the orthogonal projection onto the tangent space $T_{\gamma(t)}\mathcal{M}$.

Example A.10 (Covariant derivative on $\mathbb{S}^d$). For the unit sphere $\mathbb{S}^d \subset \mathbb{R}^{d+1}$, let $V(t)$ be a vector field along a curve $\gamma(t)$. Then

$$D_t V = \frac{dV}{dt} - \left\langle \frac{dV}{dt}, \gamma(t) \right\rangle \gamma(t), \tag{39}$$

which subtracts the normal component of the Euclidean derivative to ensure tangency.

Remark. In our framework, D_t will serve as the intrinsic analogue of time derivatives appearing in flow-map dynamics, enabling consistent definitions of velocity and acceleration fields on curved spaces.

A.8. Parallel Transport

The covariant derivative allows us to interpret $D_t V$ as the *acceleration* of a vector field transported along a curve. A vector field satisfying $D_t V = 0$ is said to be *parallel* along γ; such fields change as little as possible while remaining tangent to the manifold. This notion of parallel transport provides a canonical way to compare tangent vectors at different points on $\mathcal{M}$ and will play a central role in defining consistent dynamics and flow-based constructions on curved spaces.

Definition A.12 (Parallel vector field and parallel transport)**.** Let $\gamma : [0,1] \to \mathcal{M}$ be a smooth curve and let $V(t) \in T_{\gamma(t)}\mathcal{M}$ be a vector field along γ. We say that V is *parallel* along γ if it satisfies

$$D_t V(t) = 0 \qquad \text{for all } t \in [0,1]. \tag{40}$$

Given an initial vector $v \in T_{\gamma(0)}\mathcal{M}$, there exists a unique parallel vector field $V(t)$ along γ with $V(0) = v$. The *parallel transport* along γ is the linear map

$$P_{\gamma,\,0\to t} : T_{\gamma(0)}\mathcal{M} \to T_{\gamma(t)}\mathcal{M}, \qquad P_{\gamma,\,0\to t}(v) := V(t), \tag{41}$$

where V is the unique solution of $D_t V = 0$ with $V(0) = v$.

A.9. Connection to Flow Map Learning

In our Eulerian MeanFlow objective, we encounter the covariant derivative $D_s u^\theta_{s,t}(x_s)$, where $u^\theta_{s,t}$ is the learned average velocity and x_s moves along an integral curve. This derivative measures how the predicted average velocity changes as we move along the flow.

For embedded manifolds, this can be computed as:

$$D_s u^\theta_{s,t}(x_s) = \mathrm{Proj}_{x_s}\left(\frac{d}{ds} u^\theta_{s,t}(x_s)\right), \tag{42}$$

where the total derivative $\frac{d}{ds}$ includes both explicit dependence on s and implicit dependence through x_s:

$$\frac{d}{ds} u^\theta_{s,t}(x_s) = \partial_s u^\theta_{s,t}(x_s) + du^\theta_{s,t}(x_s)[v_s(x_s)]. \tag{43}$$

In practice, this is computed efficiently using forward-mode automatic differentiation (Jacobian-vector products), followed by projection onto the tangent space.

A.10. Differentials of the Exponential and Logarithmic Maps

For flow map learning, we need to differentiate through the exponential and logarithmic maps. Let $f : \mathcal{M} \to \mathcal{M}$ be a smooth map. The **differential** $df_x : T_x\mathcal{M} \to T_{f(x)}\mathcal{M}$ is defined by:

$$df_x(v) = \left.\frac{d}{dt}\right|_{t=0} f(\gamma(t)), \tag{44}$$

where γ is any curve with $\gamma(0) = x$ and $\dot\gamma(0) = v$.

Definition A.13 (Derivatives of the logarithmic map)**.** For the logarithmic map $\log : \mathcal{M} \times \mathcal{M} \to T\mathcal{M}$, we denote:

- $\nabla^1_v \log_x(y)$: derivative with respect to the first argument x in direction v
- $d(\log_x)_y[w]$: derivative with respect to the second argument y in direction w

These derivatives appear in our Eulerian and Lagrangian MeanFlow objectives. For many manifolds of interest (spheres, Lie groups, symmetric spaces), these derivatives have closed-form expressions.

A.11. Product Manifolds

Many applications involve product manifolds $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_N$.

Proposition A.3 (Geometry of product manifolds)**.** *For a product manifold $\mathcal{M} = \prod_{i=1}^N \mathcal{M}_i$:*

- *Tangent space:* $T_x\mathcal{M} = \prod_{i=1}^N T_{x_i}\mathcal{M}_i$
- *Metric:* $\langle u, v\rangle_x = \sum_{i=1}^N \langle u_i, v_i\rangle_{x_i}$
- *Exponential map:* $\exp_x(v) = (\exp_{x_1}(v_1), \ldots, \exp_{x_N}(v_N))$
- *Logarithmic map:* $\log_x(y) = (\log_{x_1}(y_1), \ldots, \log_{x_N}(y_N))$

This decomposition is used for protein backbone generation on $\mathrm{SE}(3)^N$, where each factor represents a residue frame.

B. Derivation of Riemannian MeanFlow Identities and Objectives

B.1. Riemannian MeanFlow Identities

In this section, we derive equivalent characterizations of the average velocity field on a Riemannian manifold. This appendix provides detailed proofs for Propositions Props. 2.1 to 2.3 stated in Sec. 2.3.

Proposition B.1 (Riemannian MeanFlow identities)**.** *A vector field $u_{s,t} : \mathcal{M} \to T\mathcal{M}$ is the average velocity associated with a time-dependent vector field v_t if and only if one (and hence all) of the following conditions holds:*

1. ***(Eulerian condition).*** *For any integral curve $(x_t)_{t\in[0,1]}$ of v_t and any $s, t \in [0, 1]$,*

$$u_{s,t}(x_s) = (t-s)\, D_s u_{s,t}(x_s) - \nabla^1_{v_s} \log_{x_s} x_t. \tag{45}$$

2. ***(Lagrangian condition).*** *For any integral curve $(x_t)_{t\in[0,1]}$ and any $s, t \in [0, 1]$,*

$$u_{s,t}(x_s) = \mathrm{d}(\log_{x_s})_{x_t}[v_t] - (t-s)\, \partial_t u_{s,t}(x_s). \tag{46}$$

3. ***(Semigroup condition).*** *For any $x_s \in \mathcal{M}$ and any $s, t \in [0, 1]$, we have $u_{s,s} = v_s$ and*

$$u_{s,t}(x_s) = \frac{1}{t-s} \log_{x_s}\big(\Phi_{r,t}\big(\Phi_{s,r}(x_s)\big)\big), \quad s \neq t, \tag{47}$$

where $\Phi_{s,t}(x) \coloneqq \exp_x\big((t-s)u_{s,t}(x)\big)$ denotes the flow map induced by $u_{s,t}$.

Proof. Throughout the proof, we assume that $u_{s,t}$ is smooth. Under this assumption, any vector field satisfying the defining relation

$$(t-s)\, u_{s,t}(x_s) = \log_{x_s} x_t. \tag{48}$$

coincides with the average velocity.

For each identity, we first derive the identity from the defining relation of the average velocity, and then argue that each condition uniquely recovers the true average velocity field.

(⇒) Eulerian identity. Both sides of Eq. (48) define vector fields along the curve $s \mapsto x_s$, so we apply the covariant derivative D_s:

$$-u_{s,t}(x_s) + (t-s)\, D_s u_{s,t}(x_s) = D_s(\log_{x_s} x_t). \tag{49}$$

Define the vector field $f : \mathcal{M} \to T\mathcal{M}$ by $f(x) \coloneqq \log_x x_t$. By definition of the covariant derivative along a curve,

$$D_s(\log_{x_s} x_t) = \nabla_{\dot{x}_s} f(x_s) = \nabla_{v_s} f(x_s) =: \nabla^1_{v_s} \log_{x_s} x_t.$$

Rearranging terms yields Eq. (45).

(⇐) Converse. For the proof of converse, assume that Eq. (45) holds along every integral curve. Fix an integral curve $(x_s)_s$ and define a vector field along it by

$$X(s) \coloneqq (t-s)u_{s,t}(x_s) - \log_{x_s} x_t \in T_{x_s}\mathcal{M}.$$

Differentiating along the curve yields

$$D_s X(s) = -u_{s,t}(x_s) + (t-s) D_s u_{s,t}(x_s) - D_s(\log_{x_s} x_t) = 0,$$

where the last equality follows from Eq. (45). Hence X is parallel along $s \mapsto x_s$. Since

$$X(t) = (t-t)u_{t,t}(x_t) - \log_{x_t} x_t = 0,$$

uniqueness of solutions to the parallel transport equation $D_s X = 0$ implies $X(s) \equiv 0$ for all s. Therefore,

$$(t-s)u_{s,t}(x_s) = \log_{x_s} x_t,$$

which is exactly the defining relation Eq. (48).

(⇒) Lagrangian identity. Differentiating Eq. (48) with respect to t, both sides lie in the fixed vector space $T_{x_s}\mathcal{M}$, so ordinary differentiation applies:

$$u_{s,t}(x_s) + (t-s)\,\partial_t u_{s,t}(x_s) = \frac{\mathrm{d}}{\mathrm{d}t}\big(\log_{x_s} x_t\big). \tag{50}$$

By the chain rule,

$$\frac{\mathrm{d}}{\mathrm{d}t}\big(\log_{x_s} x_t\big) = \mathrm{d}(\log_{x_s})_{x_t}[v_t],$$

which gives Eq. (46).

(⇐) Converse. Assume that Eq. (46) holds along every integral curve. Fix $s \in [0,1]$ and an integral curve $(x_t)_t$. Define a curve in the fixed tangent space $T_{x_s}\mathcal{M}$ by

$$X(t) \coloneqq (t-s)u_{s,t}(x_s) - \log_{x_s} x_t.$$

Differentiating with respect to t yields

$$\frac{\mathrm{d}}{\mathrm{d}t}X(t) = u_{s,t}(x_s) + (t-s)\partial_t u_{s,t}(x_s) - \mathrm{d}(\log_{x_s})_{x_t}[v_t] = 0,$$

where the last equality follows from Eq. (46). Since $X(s) = 0$, we conclude that $X(t) \equiv 0$ for all t. Therefore,

$$(t-s)u_{s,t}(x_s) = \log_{x_s} x_t,$$

which is exactly the defining relation Eq. (48).

(⇒) Semigroup identity. Let $\Phi_{s,t}$ denote the true flow map induced by v_t. By the semigroup property of the flow,

$$\Phi_{r,t}(\Phi_{s,r}(x_s)) = x_t.$$

Substituting this into the right-hand side of Eq. (47) recovers $\log_{x_s} x_t$, which is equivalent to Eq. (48). Hence the true average velocity satisfies the semigroup condition.

(⇐) Converse. Assume that the semigroup condition Eq. (47) holds for every $x \in \mathcal{M}$, and that $u_{s,s} = v_s$. We show that the induced map

$$\Phi_{s,t}(x) \coloneqq \exp_x\big((t-s)u_{s,t}(x)\big)$$

coincides with the true flow map of the time-dependent vector field v_t.

By definition,

$$\Phi_{s,s}(x) = \exp_x 0 = x,$$

so $\Phi_{s,s} = \mathrm{Id}_{\mathcal{M}}$. Moreover, by the chain rule,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\Phi_{s,t}(x)\Big|_{t=s} &= \mathrm{d}(\exp_x)_{(t-s)u_{s,t}(x)}\big[u_{s,t}(x) + (t-s)\partial_t u_{s,t}(x)\big]\big|_{t=s} \\ &= \mathrm{d}(\exp_x)_0\,[u_{s,s}(x)] \\ &= v_s(x),\end{aligned}$$

where we used the boundary condition $u_{s,s} = v_s$ and the identity $\mathrm{d}(\exp_x)_0 = \mathrm{Id}_{T_x\mathcal{M}}$. Thus,

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_{s,t}(x)\Big|_{t=s} = v_s(x). \tag{51}$$

Next, the semigroup condition implies that for any fixed r,

$$\Phi_{s,t} = \Phi_{r,t} \circ \Phi_{s,r}.$$

Differentiating both sides with respect to t yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_{s,t}(x) = \frac{\mathrm{d}}{\mathrm{d}t}\Phi_{r,t}(\Phi_{s,r}(x)).$$

Evaluating this identity at $r = t$ and using Eq. (51), we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_{s,t}(x) = v_t\big(\Phi_{s,t}(x)\big).$$

Therefore, $\Phi_{s,t}$ satisfies the ODE associated with v_t with initial condition $\Phi_{s,s} = \mathrm{Id}_{\mathcal{M}}$, and hence coincides with the true flow map. Consequently, the corresponding average velocity $u_{s,t}$ satisfies the defining relation Eq. (48) and is the true average velocity. □

B.2. Riemannian MeanFlow Objectives

Now, we give the proof of the validity of the training objective (i.e., the minimizer with each proposed training objective is the average velocity).

Proposition B.2 (Riemannian MeanFlow objectives)**.** *Let $u^\theta_{s,t} : \mathcal{M} \to T\mathcal{M}$ be a parameterized average velocity, and define the induced flow map*

$$\Phi^\theta_{s,t}(x) \coloneqq \exp_x\big((t-s)\, u^\theta_{s,t}(x)\big).$$

Consider objectives of the form

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\big\|u^\theta_{s,t}(\hat{x}_s) - \mathrm{sg}(\hat{u}_{\mathrm{tgt}})\big\|_g^2\Big], \tag{52}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator and the components are defined as follows:

1. ***Eulerian RMF:*** *$\mathbb{E} = \mathbb{E}_{x_s,s,t}$, $\hat{x}_s = x_s$, and*

$$\hat{u}_{\mathrm{tgt}} = (t-s)\, D_s u^\theta_{s,t}(x_s) - \nabla^1_{v_s(x_s)} \log_{x_s} \Phi^\theta_{s,t}(x_s).$$

2. ***Lagrangian RMF:*** *$\mathbb{E} = \mathbb{E}_{x_t,s,t}$, $\hat{x}_s = \Phi^\theta_{t,s}(x_t)$, and*

$$\hat{u}_{\mathrm{tgt}} = \mathrm{d}(\log_{\hat{x}_s})_{x_t}[v_t(x_t)] - (t-s)\, \partial_t u^\theta_{s,t}(\hat{x}_s).$$

   To promote invertibility of the induced flow maps, we additionally introduce a cycle-consistency regularizer

$$\mathcal{L}_{\mathrm{cycle}}(\theta) \;=\; \mathbb{E}_{x_t,s,t}\Big[d_g\big(\Phi^\theta_{s,t}\big(\Phi^\theta_{t,s}(x_t)\big),\, x_t\big)^2\Big],$$

   which encourages $\Phi^\theta_{t,s}$ to act as an approximate inverse of $\Phi^\theta_{s,t}$ on the data distribution.

3. ***Semigroup RMF:*** *$\mathbb{E} = \mathbb{E}_{x_s,s,r,t}$, $\hat{x}_s = x_s$, and*

$$\hat{u}_{\mathrm{tgt}} = \frac{1}{t-s} \log_{x_s} \Phi^\theta_{r,t}\big(\Phi^\theta_{s,r}(x_s)\big) \quad (t \neq s),$$

   while $\hat{u}_{\mathrm{tgt}} = v_s(x_s)$ for $t = s$.

Then any global minimizer of Eq. (52) satisfies the corresponding identity and hence recovers the average velocity.

Proof. Assume that the sampling distribution of $\hat{x}_s$ has full support on $\mathcal{M}$. Then any global minimizer of Eq. (52) satisfies

$$u^\theta_{s,t}(x_s) = \hat{u}_{\mathrm{tgt}} \quad \text{for all } x_s \in \mathcal{M},\ s,t \in [0,1].$$

The stop-gradient operator does not affect the set of global minimizers and can therefore be omitted in the analysis. For the semigroup RMF, the equality $u^\theta_{s,t}(x_s) = \hat{u}_{\mathrm{tgt}}$ directly yields the semigroup identity, which implies that $u^\theta_{s,t}$ recovers the average velocity. We therefore focus on the Eulerian and Lagrangian formulations.

Eulerian RMF. Assume that

$$u^\theta_{s,t}(x_s) = (t-s)\, D_s u^\theta_{s,t}(x_s) - \nabla^1_{v_s(x_s)} \log_{x_s} \Phi^\theta_{s,t}(x_s) \tag{53}$$

holds for all $x_s \in \mathcal{M}$ and $s, t \in [0,1]$. We show that the induced flow map $\Phi^\theta_{s,t}(x_s)$ is independent of s along any integral curve $(x_s)_s$ of v_s. This is sufficient since $\Phi^\theta_{t,t}(x_t) = x_t$, implying $\Phi^\theta_{s,t}(x_s) = x_t$ for all s.

Rearranging Eq. (53) yields

$$-u^\theta_{s,t}(x_s) + (t-s)\, D_s u^\theta_{s,t}(x_s) = \nabla^1_{v_s(x_s)} \log_{x_s} \Phi^\theta_{s,t}(x_s). \tag{54}$$

By the product rule, the left-hand side can be written as

$$-u^\theta_{s,t}(x_s) + (t-s)\, D_s u^\theta_{s,t}(x_s) = D_s\big((t-s) u^\theta_{s,t}(x_s)\big). \tag{55}$$

Moreover, by definition of the induced flow map,

$$\log_{x_s} \Phi^\theta_{s,t}(x_s) = (t-s) u^\theta_{s,t}(x_s). \tag{56}$$

Combining Eqs. (55) and (56) with Eq. (54), we obtain

$$D_s\big(\log_{x_s} \Phi^\theta_{s,t}(x_s)\big) = \nabla^1_{v_s(x_s)} \log_{x_s} \Phi^\theta_{s,t}(x_s). \tag{57}$$

Applying the chain rule to the left-hand side of Eq. (57) yields

$$D_s(\log_{x_s} \Phi^\theta_{s,t}(x_s)) = \nabla^1_{v_s(x_s)} \log_{x_s} \Phi^\theta_{s,t}(x_s) + \mathrm{d}(\log_{x_s})_{\Phi^\theta_{s,t}(x_s)} \left[\frac{\mathrm{d}}{\mathrm{d}s} \Phi^\theta_{s,t}(x_s)\right]. \tag{58}$$

Comparing Eqs. (57) and (58), the terms involving $\nabla^1 \log$ cancel, leaving

$$\mathrm{d}(\log_{x_s})_{\Phi^\theta_{s,t}(x_s)} \left[\frac{\mathrm{d}}{\mathrm{d}s} \Phi^\theta_{s,t}(x_s)\right] = 0. \tag{59}$$

Since the logarithmic map is a local diffeomorphism, its differential is invertible. Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}s} \Phi^\theta_{s,t}(x_s) = 0 \in T_{\Phi^\theta_{s,t}(x_s)} \mathcal{M}, \tag{60}$$

and $\Phi^\theta_{s,t}(x_s)$ is constant with respect to s. This completes the proof.

Lagrangian RMF. Assume that the Lagrangian identity

$$u^\theta_{s,t}(\hat{x}_s) = \mathrm{d}(\log_{\hat{x}_s})_{x_t}[v_t(x_t)] - (t-s)\, \partial_t u^\theta_{s,t}(\hat{x}_s), \qquad \hat{x}_s := \Phi^\theta_{t,s}(x_t), \tag{61}$$

holds for all $x_t \in \mathcal{M}$ and $s, t \in [0,1]$. We additionally assume a (local) invertibility condition on the induced flow maps,

$$\Phi^\theta_{s,t}\big(\Phi^\theta_{t,s}(x_t)\big) = x_t, \tag{62}$$

which is encouraged in practice by the cycle-consistency regularizer $\mathcal{L}_{\text{cycle}}(\theta) = \mathbb{E}_{x_t,s,t}\big[d_g(\Phi^\theta_{s,t}(\Phi^\theta_{t,s}(x_t)), x_t)^2\big]$.

Rearranging Eq. (61) yields

$$u^\theta_{s,t}(\hat{x}_s) + (t-s)\, \partial_t u^\theta_{s,t}(\hat{x}_s) = \mathrm{d}(\log_{\hat{x}_s})_{x_t}[v_t(x_t)].$$

Since the base point $\hat{x}_s$ is fixed when taking ∂_t, the left-hand side can be written using the product rule as

$$\frac{\mathrm{d}}{\mathrm{d}t}\big((t-s)\, u^\theta_{s,t}(x)\big)\Big|_{x=\hat{x}_s}.$$

On the other hand, by definition of the induced flow map,

$$\log_x \Phi^\theta_{s,t}(x) = (t-s)\, u^\theta_{s,t}(x).$$

Differentiating this identity with respect to t while holding x fixed and evaluating at $x = \hat{x}_s$ gives

$$\frac{\mathrm{d}}{\mathrm{d}t} \log_x \Phi^\theta_{s,t}(x)\Big|_{x=\hat{x}_s} = \mathrm{d}(\log_{\hat{x}_s})_{\Phi^\theta_{s,t}(\hat{x}_s)} \left[\frac{\mathrm{d}}{\mathrm{d}t}\Phi^\theta_{s,t}(\hat{x}_s)\right].$$

Combining the two expressions above, we obtain

$$\mathrm{d}(\log_{\hat{x}_s})_{\Phi^\theta_{s,t}(\hat{x}_s)} \left[\frac{\mathrm{d}}{\mathrm{d}t}\Phi^\theta_{s,t}(\hat{x}_s)\right] = \mathrm{d}(\log_{\hat{x}_s})_{x_t}[v_t(x_t)].$$

Using the invertibility assumption Eq. (62), we have $\Phi^\theta_{s,t}(\hat{x}_s) = x_t$, so both differentials of the logarithmic map are evaluated at the same point. Since the logarithmic map is a local diffeomorphism (away from the cut locus), its differential is invertible, which implies

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi^\theta_{s,t}(\hat{x}_s) = v_t(x_t) = v_t\big(\Phi^\theta_{s,t}(\hat{x}_s)\big).$$

Finally, since $\hat{x}_s$ ranges over $\mathcal{M}$ as x_t does under the invertibility assumption, we may relabel $\hat{x}_s$ as an arbitrary $x_s \in \mathcal{M}$ to conclude that

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi^\theta_{s,t}(x_s) = v_t\big(\Phi^\theta_{s,t}(x_s)\big).$$

This is precisely the defining ODE of the true flow map associated with v_t. Together with $\Phi^\theta_{s,s} = \mathrm{Id}$, this implies that $\Phi^\theta_{s,t}$ coincides with the true flow map, and hence $u^\theta_{s,t}$ recovers the average velocity.

Practical remark. In practice, we find that the Lagrangian objective often trains stably even without explicitly enforcing $\mathcal{L}_{\text{cycle}}$; empirically, the learned maps can become approximately cycle-consistent over the data distribution. We therefore treat $\mathcal{L}_{\text{cycle}}$ as an optional regularizer that can be enabled when stronger invertibility is desired. □

Marginal velocity approximation with conditional velocity. Here we justify that, in the training objectives Eq. (52), the marginal velocity field can be replaced by a conditional velocity without affecting the solution characterized by the objective.

Here, X_t denotes the random variable induced by the interpolant process used to couple samples at different times. Specifically, X_t is obtained by sampling a data point and evolving it according to the chosen interpolant between times 0 and 1, as in Riemannian flow matching (Chen & Lipman, 2023). The marginal velocity field is defined as the conditional expectation

$$v_t(x) \;\coloneqq\; \mathbb{E}\Big[\dot{X}_t \,\Big|\, X_t = x\Big], \tag{63}$$

interpreted as a tangent vector at x.

Fix any smooth operator $L_{x,y} : T_y\mathcal{M} \to T_x\mathcal{M}$ that is *linear* in its input tangent vector, such as $\mathrm{d}(\log_x)_y$ or, more generally, the map $w \mapsto \nabla^1_w \log_x(y)$ for fixed (x, y). By linearity of $L_{x,y}$ and the tower property of conditional expectation, we have

$$\mathbb{E}\Big[L_{X_t,\,y}\big(\dot{X}_t\big) \,\Big|\, X_t\Big] = L_{X_t,\,y}\Big(\mathbb{E}\Big[\dot{X}_t \,\Big|\, X_t\Big]\Big) = L_{X_t,\,y}\big(v_t(X_t)\big), \tag{64}$$

where the first equality follows from linearity of $L_{x,y}$, and the second from the definition Eq. (63). Taking expectations of both sides yields

$$\mathbb{E}\Big[L_{X_t,\,y}\big(\dot{X}_t\big)\Big] = \mathbb{E}\big[L_{X_t,\,y}\big(v_t(X_t)\big)\big]\,. \tag{65}$$

Therefore, in objectives where the velocity field appears only through such linear operators—as is the case for the Eulerian and Lagrangian RMF objectives— the marginal velocity $v_t(X_t)$ can be replaced by the unbiased estimator of conditional velocity sample $\dot{X}_t$. without changing the expected regression target; the replacement only affects its variance.

C. Theoretical Connections to Existing Flow-map Learning Methods

C.1. Eulerian RMF as a Riemannian Generalization of MeanFlow

In this section, we show that the proposed Eulerian RMF reduces exactly to the Euclidean MeanFlow objective when the underlying manifold is $\mathbb{R}^d$. This establishes Eulerian RMF as a direct Riemannian generalization of MeanFlow.

Recall that the Eulerian RMF regression target is given by

$$\hat{u}_{\mathrm{tgt}} = (t-s)\, D_s u^{\theta}_{s,t}(x_s) - \nabla^1_{v_s(x_s)} \log_{x_s}\big(\Phi^{\theta}_{s,t}(x_s)\big), \tag{66}$$

where D_s denotes the covariant derivative along the integral curve x_s and ∇^1 denotes the covariant derivative of the logarithmic map with respect to its first (base-point) argument.

We now specialize to the Euclidean setting $\mathcal{M} = \mathbb{R}^d$. In this case, the Levi–Civita connection is flat, and the covariant derivative reduces to the ordinary derivative. In particular, we have

$$D_s u^{\theta}_{s,t}(x_s) = \frac{d}{ds} u^{\theta}_{s,t}(x_s). \tag{67}$$

Moreover, the logarithmic map in Euclidean space is given by $\log_x(y) = y - x$, and therefore its derivative with respect to the base point satisfies

$$\nabla^1_{v_s(x_s)} \log_{x_s}\big(\Phi^{\theta}_{s,t}(x_s)\big) = -\, v_s(x_s). \tag{68}$$

Substituting these identities into (66), the Eulerian RMF target reduces to

$$\hat{u}_{\mathrm{tgt}} = (t-s)\, \frac{d}{ds} u^{\theta}_{s,t}(x_s) + v_s(x_s), \tag{69}$$

which is exactly the regression target used in Euclidean MeanFlow (Geng et al., 2025). This shows that Eulerian RMF recovers MeanFlow in the Euclidean case, while providing a principled extension to general Riemannian manifolds through intrinsic geometric operators.

C.2. Connection to Generalized Flow Map

In this section, we detail a theoretical connection between Riemannian MeanFlow and Generalized Flow Map (GFM) objectives. We first derive our objectives from GFM self-distillation objectives and show that our objective can be viewed as GFM objectives with a properly applied stop-gradient operation, thereby entirely avoiding backpropagation through Jacobian-vector products (JVPs). This suggests that our derivation reveals a further, practically important design space that is not focused on GFM objectives.

C.2.1. Brief derivation of GFM objectives

We begin by briefly reviewing the objectives proposed in GFM. Their derivation starts from the defining relation of the flow map introduced in Definition 2.2:

$$\Phi_{s,t}(x_s) = x_t, \tag{70}$$

which holds for any integral curve $(x_t)_{t\in[0,1]}$ and any $s, t \in [0,1]$. To construct learning objectives, GFM differentiates this identity with respect to the time variables s and t. Differentiation with respect to the source time s yields Eulerian-type objectives, while differentiation with respect to the target time t yields Lagrangian-type objectives.

Specifically, differentiating (70) with respect to s gives the generalized Eulerian characterization

$$\frac{d}{ds}\, \Phi_{s,t}(x_s) = \partial_s \Phi_{s,t}(x_s) + d\Phi_{s,t}(x_s)[v_s] = 0, \tag{71}$$

where v_s denotes the velocity along the integral curve. GFM enforces this identity via the regression objective

$$\mathcal{L}_{\text{G-ESD}}(\theta) = \mathbb{E}_{x_s,s,t}\left[\left\|\partial_s \Phi^{\theta}_{s,t}(x_s) + d\Phi^{\theta}_{s,t}(x_s)\big[v^{\theta}_s(x_s)\big]\right\|_g^2\right], \tag{72}$$

referred to as the *G-ESD* objective.

Likewise, differentiating (70) with respect to t yields the Lagrangian characterization

$$\partial_t \Phi_{s,t}(x_s) = v_t(x_t) = v_t(\Phi_{s,t}(x_s))\,, \tag{73}$$

which is precisely the defining property of an integral curve, i.e., $\Phi_{s,t}(x_s)$ solves the ODE $\frac{d}{dt}x_t = v_t(x_t)$. Enforcing this identity via regression leads to the *G-LSD* objective,

$$\mathcal{L}_{\text{G-LSD}}(\theta) = \mathbb{E}_{x_s,s,t}\left[\left\|\partial_t \Phi^\theta_{s,t}(x_s) - v^\theta_t\big(\Phi^\theta_{s,t}(x_s)\big)\right\|_g^2\right]. \tag{74}$$

Finally, GFM introduces a semigroup objective that enforces the semigroup property of the flow map, $\Phi_{r,t}(\Phi_{s,r}(x_s)) = \Phi_{s,t}(x_s)$, via

$$\mathcal{L}_{\text{G-PSD}}(\theta) = \mathbb{E}_{x_s,s,r,t}\left[d_g^2\big(\Phi^\theta_{s,t}(x_s), \Phi^\theta_{r,t}\big(\Phi^\theta_{s,r}(x_s)\big)\big)\right]. \tag{75}$$

For practical optimization, GFM applies the stop-gradient operator to obtain the following losses:

$$\mathcal{L}_{\text{G-ESD}}(\theta) = \mathbb{E}_{x_s,s,t}\left[\left\|\partial_s \Phi^\theta_{s,t}(x_s) + \mathrm{sg}\big(d\Phi^\theta_{s,t}(x_s)\big[v^\theta_s(x_s)\big]\big)\right\|_g^2\right], \tag{76}$$

$$\mathcal{L}_{\text{G-LSD}}(\theta) = \mathbb{E}_{x_s,s,t}\left[\left\|\partial_t \Phi^\theta_{s,t}(x_s) - \mathrm{sg}\big(v^\theta_t\big(\Phi^\theta_{s,t}(x_s)\big)\big)\right\|_g^2\right], \tag{77}$$

$$\mathcal{L}_{\text{G-PSD}}(\theta) = \mathbb{E}_{x_s,s,r,t}\left[d_g^2\big(\Phi^\theta_{s,t}(x_s), \mathrm{sg}\big(\Phi^\theta_{r,t}\big(\Phi^\theta_{s,r}(x_s)\big)\big)\big)\right], \tag{78}$$

where sg denotes the stop-gradient operator.

Importantly, the objectives G-ESD, G-LSD, and G-PSD enforce only *flow-map consistency*. This consistency alone does not guarantee that the velocity field v^θ_s corresponds to the true data-generating dynamics. To recover the desired flow map, GFM therefore augments the above objectives with a flow-matching loss that explicitly trains v^θ_s.

C.2.2. RFM AS A PRINCIPLED REFINEMENT OF GFM

In the following, we analyze these objectives in more detail and clarify the role of the stop-gradient operator in GFM. In prior works on consistency models (Song et al., 2023) and MeanFlow (Geng et al., 2025), the stop-gradient operator is primarily introduced to avoid computing expensive higher-order derivatives (e.g., gradients through Jacobian–vector products) and to improve computational efficiency and optimization stability. In GFM, stop-gradient is likewise employed in the differential objectives; however, due to the structure of these objectives, it blocks higher-order derivatives only partially and therefore does not fully eliminate the associated computational overhead. From this perspective, our formulation can be viewed as a principled refinement that yields differential objectives in which higher-order derivatives are avoided by construction.

Finally, for the semigroup objective, we show that the G-PSD objective and our corresponding formulation can be heuristically related through loss weighting with respect to the length of the time interval. We empirically demonstrate that our formulation leads to superior performance in App. D.

C.2.3. FROM G-ESD TO EULERIAN RMF

We show that the Eulerian RMF objective arises as a first-order expansion of the GFM Eulerian self-distillation (G-ESD) objective under an exponential-map parameterization of the flow map. Recall that the G-ESD objective is based on the Eulerian consistency residual

$$\Delta_{\text{G-ESD}}(x) := \partial_s \Phi^\theta_{s,t}(x) + d\Phi^\theta_{s,t}(x)[v_s(x)]\,, \tag{79}$$

where ∂_s denotes the partial derivative with respect to the flow-map parameter s.

We parameterize the flow map using the average velocity field as

$$\Phi^\theta_{s,t}(x) = \exp_x\big((t-s)\,u^\theta_{s,t}(x)\big), \tag{80}$$

and define $\xi(x) := (t-s)\,u^\theta_{s,t}(x)$. In what follows, we evaluate all expressions along the interpolant $x = x_s$.

Expansion of the time derivative. Using the chain rule for the exponential map, we obtain

$$\partial_s \Phi^\theta_{s,t}(x_s) = d_2 \exp_{x_s}(\xi_s)\Big[-\,u^\theta_{s,t}(x_s) + (t-s)\,\partial_s u^\theta_{s,t}(x_s)\Big], \tag{81}$$

where $d_2 \exp$ denotes the differential of $\exp$ with respect to its second (tangent-vector) argument.

Expansion of the pushforward term. Writing $\Phi^\theta_{s,t}(x) = \exp_x(\xi(x))$, the differential with respect to the base point x yields

$$d\Phi^\theta_{s,t}(x_s)[v_s(x_s)] = d_1 \exp_{x_s}(\xi_s)[v_s(x_s)] + d_2 \exp_{x_s}(\xi_s)\left[\nabla_{v_s(x_s)}\xi(x_s)\right], \tag{82}$$

where $d_1 \exp$ denotes the differential with respect to the base point.

Combining (81) and (82), the G-ESD residual becomes

$$\Delta_{\text{G-ESD}}(x_s) = d_1 \exp_{x_s}(\xi_s)[v_s(x_s)] + d_2 \exp_{x_s}(\xi_s)\left[\nabla_{v_s(x_s)}\xi(x_s) - u^\theta_{s,t}(x_s) + (t-s)\,\partial_s u^\theta_{s,t}(x_s)\right]. \tag{83}$$

Pull-back to the tangent space. To express the residual in $T_{x_s}\mathcal{M}$, we pull it back via the differential of the log map at x_s. Let $y_s := \Phi^\theta_{s,t}(x_s) = \exp_{x_s}(\xi_s)$ and define

$$\widehat{\Delta}_{\text{G-ESD}}(x_s) := d(\log_{x_s})_{y_s}[\Delta_{\text{G-ESD}}(x_s)]. \tag{84}$$

Within a normal neighborhood, the identities

$$d(\log_{x_s})_{y_s} \circ d_2 \exp_{x_s}(\xi_s) = \mathrm{Id}_{T_{x_s}\mathcal{M}}, \qquad d(\log_{x_s})_{y_s}\left[d_1 \exp_{x_s}(\xi_s)[v]\right] = -\nabla^1_v \log_{x_s}(y_s) \tag{85}$$

hold, where ∇^1 denotes the covariant derivative with respect to the first argument of the log map.

Applying these identities to (83), we obtain

$$\widehat{\Delta}_{\text{G-ESD}}(x_s) = -\nabla^1_{v_s(x_s)} \log_{x_s}\left(\Phi^\theta_{s,t}(x_s)\right) - u^\theta_{s,t}(x_s) + (t-s)\Big(\partial_s u^\theta_{s,t}(x_s) + \nabla_{v_s(x_s)} u^\theta_{s,t}(x_s)\Big). \tag{86}$$

Introducing the covariant derivative along the interpolant,

$$D_s u^\theta_{s,t}(x_s) := \partial_s u^\theta_{s,t}(x_s) + \nabla_{v_s(x_s)} u^\theta_{s,t}(x_s), \tag{87}$$

the residual can be written compactly as

$$\widehat{\Delta}_{\text{G-ESD}}(x_s) = -\nabla^1_{v_s(x_s)} \log_{x_s}\left(\Phi^\theta_{s,t}(x_s)\right) - u^\theta_{s,t}(x_s) + (t-s)\, D_s u^\theta_{s,t}(x_s). \tag{88}$$

Setting (88) to zero yields

$$u^\theta_{s,t}(x_s) = (t-s)\, D_s u^\theta_{s,t}(x_s) - \nabla^1_{v_s(x_s)} \log_{x_s}\left(\Phi^\theta_{s,t}(x_s)\right), \tag{89}$$

which exactly recovers the Eulerian RMF regression target used in our method.

Relation between G-ESD and Eulerian RMF objectives. The above derivation shows that the G-ESD and Eulerian RMF objectives enforce the same Eulerian consistency condition, differing primarily in the space in which the residual is represented. G-ESD minimizes the residual $\Delta_{\text{G-ESD}}(x_s) \in T_{\Phi^\theta_{s,t}(x_s)}\mathcal{M}$, defined at the transported point $\Phi^\theta_{s,t}(x_s)$. In contrast, Eulerian RMF applies an invertible change of coordinates given by the log-map differential $d(\log_{x_s})_{\Phi^\theta_{s,t}(x_s)}$, which pulls the residual back to the reference tangent space $T_{x_s}\mathcal{M}$. The two residuals are related by

$$\widehat{\Delta}_{\text{G-ESD}}(x_s) = d(\log_{x_s})_{\Phi^\theta_{s,t}(x_s)}\left[\Delta_{\text{G-ESD}}(x_s)\right],$$

so both objectives share the same zero set of the consistency constraint.

This perspective also clarifies the practical effect of stop-gradient. In Eulerian RMF, stop-gradient is applied to the entire regression target. In contrast, G-ESD applies stop-gradient at the level of the Eulerian residual; under the pull-back above, this induces a *partial* stop-gradient in the Eulerian RMF form, acting only on the geometric transport terms $\nabla^1_{v_s(x_s)} \log_{x_s}\left(\Phi^\theta_{s,t}(x_s)\right)$ and $\nabla_{v_s(x_s)} u^\theta_{s,t}(x_s)$, while leaving the explicit time-derivative term $\partial_s u^\theta_{s,t}(x_s)$ differentiable.

C.2.4. From G-LSD to Lagrangian RMF.

We now establish an analogous connection between the Lagrangian objectives of GFM and our Lagrangian RMF. Recall that the GFM Lagrangian self-distillation (G-LSD) objective is defined as

$$\mathcal{L}_{\text{G-LSD}}(\theta) = \mathbb{E}_{x_s,s,t}\left[\left\|\partial_t \Phi^\theta_{s,t}(x_s) - v^\theta_t\big(\Phi^\theta_{s,t}(x_s)\big)\right\|_g^2\right], \tag{90}$$

which enforces the Lagrangian consistency condition $\partial_t \Phi_{s,t}(x_s) = v_t(\Phi_{s,t}(x_s))$ along transported particles.

As in the Eulerian case, we adopt the average-velocity parameterization

$$\Phi^\theta_{s,t}(x) = \exp_x\big((t-s)\, u^\theta_{s,t}(x)\big), \tag{91}$$

and evaluate all quantities along the interpolant $x = x_s$. Differentiating with respect to t yields

$$\partial_t \Phi^\theta_{s,t}(x_s) = d_2 \exp_{x_s}(\xi_s)\Big[u^\theta_{s,t}(x_s) + (t-s)\,\partial_t u^\theta_{s,t}(x_s)\Big], \qquad \xi_s := (t-s)u^\theta_{s,t}(x_s). \tag{92}$$

To express the velocity term in a compatible form, we use the identity $\exp_{x_s}(\log_{x_s} y) = y$, which implies

$$v_t\big(\Phi^\theta_{s,t}(x_s)\big) = d_2 \exp_{x_s}(\xi_s)\Big[d_2 \log_{x_s}\big(\Phi^\theta_{s,t}(x_s)\big)\big[v_t(\Phi^\theta_{s,t}(x_s))\big]\Big]. \tag{93}$$

Substituting (92) and (93) into (90), and using the invertibility of $d_2 \exp_{x_s}(\xi_s)$ within a normal neighborhood, the G-LSD objective reduces to an equivalent regression in $T_{x_s}\mathcal{M}$:

$$\mathcal{L}_{\text{G-LSD}}(\theta) \;\equiv\; \mathbb{E}_{x_s,s,t}\left[\left\|u^\theta_{s,t}(x_s) + (t-s)\,\partial_t u^\theta_{s,t}(x_s) - d_2 \log_{x_s}\big(\Phi^\theta_{s,t}(x_s)\big)\big[v_t(\Phi^\theta_{s,t}(x_s))\big]\right\|_g^2\right]. \tag{94}$$

This expression coincides with the Lagrangian RMF objective up to the treatment of stop-gradient. In particular, while Lagrangian RMF applies stop-gradient to the entire regression target, the stop-gradient G-LSD objective—defined at the flow-map level—induces a *partial* stop-gradient under the above pull-back, affecting the transport and log-map terms while leaving the explicit time-derivative $\partial_t u^\theta_{s,t}(x_s)$ differentiable. As in the Eulerian case, this shows that Lagrangian RMF can be interpreted as a geometrically equivalent reparameterization of G-LSD, expressed in the tangent space of the current state.

C.2.5. Generalised MeanFlow vs. Eulerian RMF.

Furthermore, Davis et al. (2025) proposes a heuristic extension of Euclidean MeanFlow to Riemannian manifolds, referred to as *Generalised MeanFlow* (G-MF). The main difficulty identified in their derivation arises from formulating MeanFlow *via an integral representation of the average velocity* on a Riemannian manifold. In particular, extending Euclidean MeanFlow directly requires defining the integral of a vector field along a curve on $\mathcal{M}$, which involves parallel transport (and thus connections) and introduces non-trivial curvature-dependent terms. As noted by the authors, this makes a direct generalization of Euclidean MeanFlow non-trivial.

To bypass this difficulty, G-MF does not attempt to rigorously define the integral of the average velocity on the manifold. Instead, it heuristically follows the stop-gradient derivation of Euclidean MeanFlow and replaces Euclidean derivatives with covariant derivatives induced by the Levi–Civita connection. This leads to the regression objective

$$\widehat{\mathcal{L}}_{\text{G-MF}}(\theta) = \mathbb{E}_{t,s,x_s}\Big[\big\|u^\theta_{s,t}(x_s) - \text{stopgrad}\big(v_s - (t-s)\nabla_{v_s} u^\theta_{s,t}(x_s)\big)\big\|_g^2\Big]. \tag{95}$$

In contrast, our Eulerian RMF is derived from a different starting point. Rather than relying on an integral formulation of the average velocity field, we base the derivation on identities satisfied by the average velocities along *integral curves* of the flow. This perspective entirely avoids the need to define integrals of vector fields on the manifold. As a result, the characterizing identity of the average velocity can be differentiated directly using covariant derivatives, yielding Eulerian learning objectives that are intrinsic and well-defined on Riemannian manifolds. By grounding the derivation in integral-curve-based identities, Eulerian RMF provides a direct and principled generalization of Euclidean MeanFlow (Geng et al., 2025), effectively replacing the heuristically constructed G-MF objective.

C.2.6. SEMIGROUP RMF VS. G-PSD.

We compare the Semigroup RMF and G-PSD objectives in the Euclidean setting and show that they differ only by a time-dependent loss weighting. In Euclidean space, the Semigroup RMF per-sample loss is given by

$$L_{\text{S-RMF}}(\theta) = \left\| u^{\theta}_{s,t}(x_s) - \text{sg}\left(\frac{r-s}{t-s}\, u^{\theta}_{s,r}(x_s) + \frac{t-r}{t-s}\, u^{\theta}_{r,t}(\hat{x}_r) \right) \right\|_2^2, \tag{96}$$

with $\hat{x}_r = \Phi^{\theta}_{s,r}(x_s) = x_s + (r-s)u^{\theta}_{s,r}(x_s)$. In the same setting, the G-PSD objective reduces to

$$L_{\text{G-PSD}}(\theta) = \left\| (t-s)\, u^{\theta}_{s,t}(x_s) - \text{sg}\big((r-s)\, u^{\theta}_{s,r}(x_s) + (t-r)\, u^{\theta}_{r,t}(\hat{x}_r) \big) \right\|_2^2. \tag{97}$$

The two objectives are related by

$$L_{\text{G-PSD}}(\theta) = (t-s)^2\, L_{\text{S-RMF}}(\theta), \tag{98}$$

indicating that their difference can be fully characterized by a time-dependent loss weighting $w(s,t) = (t-s)^2$.

On general Riemannian manifolds, this equivalence no longer holds exactly due to curvature-dependent effects. In practice, we observe that Semigroup RMF achieves slightly better performance than G-PSD.

D. Empirical Comparison to GFM

In this section, we empirically compare our approach to the concurrent Generalized Flow Matching (GFM) method (Davis et al., 2025) using the geospatial Earth dataset and a high-dimensional DNA promoter design task. Our results indicate that while GFM struggles to scale reliably in the DNA task, our method remains stable and performs better due to our proposed stabilization techniques. Furthermore, we analyze the computational overhead of both methods, specifically comparing training time, memory usage, and NFEs per iteration. These comparisons highlight the superior optimization behavior and scaling properties of our framework over GFM.

D.1. Toy Earth Datasets

To provide a quantitative comparison between our proposed objectives and GFM (Davis et al., 2025), we evaluate our method on the geospatial Earth events benchmark on $\mathbb{S}^2$ (Mathieu & Nickel, 2020), following the evaluation protocols established in Chen & Lipman (2023) and Davis et al. (2025). In these experiments, we fix the parameterization to v-prediction to isolate and investigate the impact of different training objectives.

Metric. We report the empirical Maximum Mean Discrepancy (MMD) between the test data and generated samples, consistent with Davis et al. (2025). For the MMD computation, we employ a geodesic-based RBF kernel, $k(x, y) = \exp(-d_g(x, y)^2/(2\kappa^2))$, with a bandwidth $\kappa = 1$. We omit the test Negative Log-Likelihood (NLL) because exact NLL evaluation is intractable unless the flow map is strictly invertible or satisfies specific regularity conditions (Rehman et al., 2025).

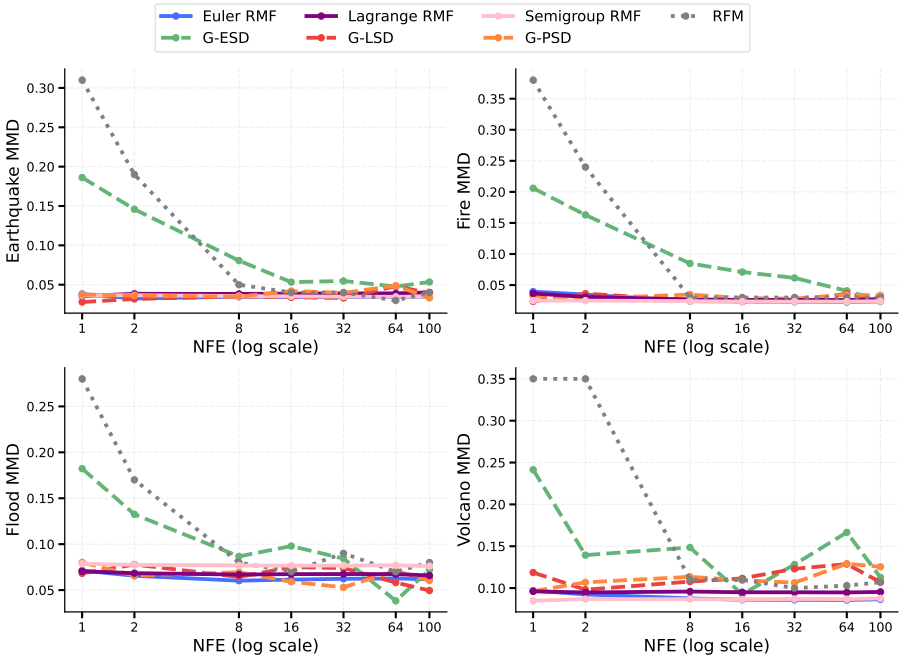


Figure A1. **Inference steps vs. MMD on the Earth dataset.** Empirical MMD (lower is better) as a function of the number of inference steps for four datasets (Volcano, Earthquake, Flood, and Fire).

Results. As illustrated in Fig. A1, our Riemannian MeanFlow objectives achieve competitive or superior MMD values on the Earth dataset compared to the GFM baseline. Notably, all of our objective variants (Eulerian, Lagrangian, and Semigroup) yield comparable results with 100-step sampling. In contrast, GFM-Eulerian exhibits a significantly higher MMD in its 1-step performance. We hypothesize that this discrepancy arises because GFM's formulation does not entirely bypass backpropagation through the Jacobian-vector product (JVP), which potentially destabilizes the optimization process compared to our JVP-free objectives.

Table A1. One-step performance on test-set of promoter DNA sequence.

Method		MSE (↓)	k-mer corr. (↑)
G-ESD	v-pred	0.055	0.13
G-LSD	v-pred	0.046	0.73
G-PSD	v-pred	0.035	0.82
Euler RMF	x_1-pred	$0.030_{\pm 0.000}$	$0.96_{\pm 0.01}$
	v-pred	$0.031_{\pm 0.001}$	$0.96_{\pm 0.00}$
Lagrange RMF	x_1-pred	$0.027_{\pm 0.001}$	$0.88_{\pm 0.00}$
	v-pred	$0.027_{\pm 0.001}$	$0.85_{\pm 0.01}$
Semigroup RMF	x_1-pred	$0.030_{\pm 0.001}$	$0.84_{\pm 0.03}$
	v-pred	$0.030_{\pm 0.001}$	$0.93_{\pm 0.02}$

Table A2. Training cost of G-ESD and Eulerian MF. We measure the memory consumption and training time per iteration on the DNA task.

	NFE/it	Memory (GB) ↓	Training time (s/it) ↓
G-ESD	3	17.7	0.40
E-RMF	1	9.5	0.15
E-RMF self-distillation	2	9.5	0.16

D.2. Promoter DNA Datasets

To demonstrate the scalability of our objectives compared to GFM (Davis et al., 2025), we evaluate both methods on a high-dimensional DNA promoter design task. For fair comparison, we fix the model architecture, optimizer, time sampling scheme, batch size, and the total number of training iterations. The experimental setup for these evaluations is consistent with the protocols described in Sec. 4; further implementation details can be found in Apps. E and F.

Results. As summarized in Table A1, we observe a significant performance gap between the two frameworks. Specifically, GFM's differential objectives—Eulerian (G-ESD) and Lagrangian (G-LSD)—fail to achieve meaningful convergence, resulting in high MSE. We hypothesize that this failure stems from optimization instabilities caused by the high variance of the network output's derivatives in high-dimensional spaces. In contrast, all variants of our Riemannian MeanFlow (Eulerian, Lagrangian, and Semigroup) maintain robust stability throughout training. Our methods consistently outperform GFM across all metrics.

D.3. Computational Efficiency Analysis

To further evaluate the practical utility of our proposed objectives, we conduct a comparative analysis of the computational costs between our Eulerian RMF (Eulerian RMF) and the Generalized Flow Map Eulerian (G-ESD) objective (Davis et al., 2025). As summarized in Table A2, our method demonstrates superior efficiency in terms of both memory consumption and training speed.

Number of function evaluations (NFEs). A significant advantage of our formulation is the reduction in the number of network evaluations per training iteration. G-ESD requires 3 NFEs per iteration: because the stop-gradient operator is applied only to the spatial derivative $\mathrm{d}\Phi^{\theta}_{s,t}$, the time-derivative $\partial_s \Phi^{\theta}_{s,t}$ and the spatial JVP term must be evaluated separately. In contrast, our Eulerian RMF evaluates both the average velocity $u^{\theta}_{s,t}$ and its time derivative $D_s u^{\theta}_{s,t}$ within a single forward pass, requiring only 1 NFE. Even in our self-distillation variant, which utilizes a learned instantaneous velocity, the cost remains at 2 NFEs, still lower than that of the standard GFM Eulerian objective.

Memory usage and optimization. All memory and throughput numbers are measured on an NVIDIA RTX 3090 GPU. As shown in Table A2, G-ESD incurs substantially higher memory usage (17.7 GB) compared to Eulerian RMF (9.5 GB). This disparity arises because G-ESD does not fully avoid backpropagation through the JVP output $\partial_s \Phi^{\theta}_{s,t}$. The resulting computational graph for G-ESD is more complex, requiring approximately twice the memory of our method. Our JVP-free formulation not only reduces the memory footprint but also contributes to the optimization stability observed in high-dimensional tasks like DNA promoter design, where G-ESD failed to converge (Table A1).

Training throughput. The combination of fewer NFEs and lower memory overhead leads to a marked improvement in training speed. Eulerian RMF achieves a training time of 0.15 s/it, which is approximately 2.6× faster than G-ESD's 0.40 s/it. The self-distillation variant of Eulerian RMF also maintains high throughput (0.16 s/it), demonstrating that the efficiency gains are robust across different variants of our objective.

E. Evaluation details

E.1. DNA Promoter Design

Task description. Promoters are critical DNA sequences that dictate the initiation and magnitude of gene transcription (Haberle & Stark, 2018). The objective of this task is to generate promoter sequences conditioned on a desired transcription signal profile. Successful generation of these sequences enables precise control over the expression levels of synthetic genes, which is essential for applications such as controlled gene expression and synthetic biology. For DNA promoter design, we use MSE for evaluation following Davis et al. (2024); Stark et al. (2024), and additionally report k-mer correlation to assess whether local sequence patterns are preserved.

MSE. Following prior work, we evaluate promoter activity using mean squared error (MSE) between transcription signal profiles predicted by a pretrained Sei model from generated sequences and those predicted from the corresponding test-set reference sequences under the same condition. Specifically, for each conditioning signal in the test set, we compute Sei-predicted profiles for both the generated and reference sequences, measure their MSE, and report the average over the test set. This metric quantifies how closely the generated sequences match the regulatory activity of the reference sequences.

k-mer correlation. In addition to MSE, we measure the k-mer correlation between the generated sequences and the empirical test distribution. This metric evaluates whether the generated promoter sequences preserve local sequence patterns, capturing compositional similarity beyond global activity prediction. Concretely, we aggregate the generated sequences into a single k-mer frequency vector, aggregate the test-set sequences into another k-mer frequency vector, and report the Pearson correlation between the two vectors.

E.2. DNA Promoter Reward Guidance

Reward function. For optimizing the regulatory footprint of a DNA sequence based on a target profile, we use the following reward function:

$$r(x) = -\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}\left(\mathrm{Sei}(x)[n] - \mathrm{Sei}(x_{\mathrm{target}})[n]\right)^2, \tag{99}$$

where $\mathcal{N}$ corresponds to the promoter-related features from Sei. Note that lower MSE is better, so we negate the MSE metric to maximize it.

Ablation over guidance scale. We perform a grid search over the guidance scale (λ) in Eq. (19), evaluating values $\lambda \in \{1, 10, 100, 1000\}$ and show the performance in Table A3. For each type of reward guidance at each NFE, we report the best performance in Table 2.

Table A3. Ablation over the guidance scale λ in Eq. (19) for reward-guided promoter DNA generation. We report mean squared error (MSE) $\pm$ standard deviation across 60 batches of 128 samples for two reward evaluations: the naive approach based on the current state ($\mathrm{Proj}_{x_t}(\nabla r(x_t))$) and using x_1 look-ahead ($\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$).

NFE	λ	$\mathrm{Proj}_{x_t}(\nabla r(x_t))$	$\mathrm{Proj}_{x_t}(\nabla r(\Phi^{\theta}_{t,1}(x_t)))$
1	1	$0.033_{\pm 0.015}$	$0.026_{\pm 0.011}$
	10	$0.033_{\pm 0.015}$	$0.025_{\pm 0.011}$
	100	$0.033_{\pm 0.015}$	$0.049_{\pm 0.033}$
	1000	$0.033_{\pm 0.015}$	$0.068_{\pm 0.053}$
5	1	$0.031_{\pm 0.014}$	$0.021_{\pm 0.008}$
	10	$0.031_{\pm 0.014}$	$0.013_{\pm 0.005}$
	100	$0.026_{\pm 0.011}$	$0.024_{\pm 0.013}$
	1000	$0.017_{\pm 0.009}$	$0.048_{\pm 0.039}$
10	1	$0.031_{\pm 0.013}$	$0.017_{\pm 0.007}$
	10	$0.031_{\pm 0.013}$	$0.008_{\pm 0.003}$
	100	$0.025_{\pm 0.010}$	$0.016_{\pm 0.007}$
	1000	$0.008_{\pm 0.002}$	$0.036_{\pm 0.029}$

E.3. Protein Backbone Design

We mainly follow the evaluation pipeline and metric definition of FrameFlow (Yim et al., 2023a), FrameDiff (Yim et al., 2023b), and La Proteina (Geffner et al., 2025a). We sample 10 backbone structures for every length between 60 and 128, and measure three metrics for the generated samples: designability, diversity, and novelty. For Table 3, we have reproduced FrameDiff and FrameFlow, and have report metrics below.

Designability. We assess the designability with the *self-consistency* evaluation from Trippe et al. (2023), measuring how closely a generated backbone can be recovered by sequence design and refolding. To be specific, we use ProteinMPNN (Dauparas et al., 2022) to achieve 8 sequences for each backbone structure and re-fold, i.e., predict their backbone structures using ESMfold (Lin et al., 2023). Afterwards, we compute the root-mean-square-distance (`scRMSD`) between the backbone structures and the generated backbone structure. We also report the designable fraction with a threshold of $\texttt{scRMSD} < 2.0$ Å.

Diversity. Diversity measures how many distinct structural conformations the model generates. For each length, we cluster all the generated backbones using MaxCluster (Herbert & Sternberg, 2008), and report the number of clusters divided by the total number of designable samples. Additionally, we also report the pairwise `scTM`, which quantifies the structural similarity between the generated backbones. A lower pairwise `scTM` indicate higher diversity, as they reflect larger structural deviation between the generated samples . The MaxCluster command used to compute this metric is given by

```
maxcluster -l <pdb file list> -C 3 -Tm 0.8 -noalign
```

where `<pdb file list>` is the path for a text file containing the list of paths to PDB files.

Novelty. Novelty evaluates how different the generated backbones are from known protein structures. For each designable sample, we use FoldSeek (Van Kempen et al., 2024) to search the PDB database and compute the highest TM-score to any matching chain (Zhang & Skolnick, 2005, pdbTM). Afterwards, we report the average pdbTM across all samples. The FoldSeek command used to compute this metric is given by

```
foldseek easy-search <path sample> <reference database path> <result file>
<tmp path> --format-output query,target,alntmscore
```

where `<path sample>` is the path for PDB files containing the generated structure, and `<reference database path>` is the path of the dataset used as reference, for which we use the Protein Data Bank (PDB).

E.4. Protein Reward Guidance

Reward function. We design a differentiable reward function as based on PyDSSP[1], an open-source implementation of the Define Secondary Structure of Proteins (DSSP) algorithm (Kabsch & Sander, 1983), which is the standard method secondary structure assignment to protein residues. PyDSSP implements DSSP in PyTorch.

PyDSSP first computes a hydrogen-bond energy map using the DSSP electrostatic model, where hydrogen bond energies are calculated from interatomic distances between backbone atoms (O–N, C–H, O–H, and C–N), and then uses a smooth, differentiable approximation to determine hydrogen bond presence. Secondary structure elements are then identified from this map following DSSP-style rules: turns are detected from diagonal hydrogen bonds between residues separated by three to five positions, helices are defined by consecutive turns, and β-bridges are identified via parallel and antiparallel hydrogen-bond patterns using a local unfolding window. However, in order to identify these structural elements, non-differentiable boolean tensors are produced and gradients are broken.

Instead, we construct a soft score for each each structural class independently, which we call DiffDSSP. That is, we extract differentiable pattern scores from the hydrogen-bond map corresponding to α-helices and β-sheets. These scores quantify the extent to which each residue participates in helix- or strand-like hydrogen-bond patterns, without requiring hard assignments or binary decisions. The reward then becomes maximizing these score to promote generation of a specific structural class.

Note that one limitation of this reward function is that is inherently sparse, as residues not participating in secondary structure organization have a score of 0.0. We found that using a constrastive-style reward was better, where guidance

[1] https://github.com/ShintaroMinami/PyDSSP

toward a desired secondary structure is accompanied by simultaneous repulsion from the alternative structure:

$$r(x) = w_\beta \cdot \text{DiffDSSP}_\beta(x) + w_\alpha \cdot \text{DiffDSSP}_\alpha(x)\,, \tag{100}$$

where $\text{DiffDSSP}_\beta(x)$ and $\text{DiffDSSP}_\alpha(x)$ output differentiable scores measuring the extent of β-sheet and α-helix content, respectively, in x, and w_β, w_α control their relative contributions. When steering toward increased β-sheet content, we set $w_\beta = 1$ and $w_\alpha = -1$, while for α-helix guidance, we set $w_\beta = -1$ and $w_\alpha = 1$.

Final evaluation. To evaluate the final generated sequences, we used the original DSSP algorithm (Kabsch & Sander, 1983), which is the standard way for assigning structural classes to a protein. We used `dssp-2.0.4-linux-amd64`[2] with the following command:

```
dssp -i <pdb file list>
```

Ablation over guidance scale. For each reward guidance method, NFE, and secondary structure class (β-sheet, α-helix), we computed the performance over $\lambda \in [1, 10, 100, 1000]$ on 100 sequences of length 128 and report the setting with the highest mean structure composition in Table 4.

[2] https://swift.cmbi.umcn.nl/gv/dssp/

F. Experiment Details

F.1. Promoter DNA Design

We model promoter DNA sequences as continuous, relaxed representations of length 1024, i.e., arrays in $\mathbb{R}^{1024\times 4}$ with support restricted to the positive orthant, and interpret them as points on a product of spheres. We learn a time-dependent velocity field on this manifold using the average-velocity parameterization of Riemannian MeanFlow. For n-step evaluation, we discretize the time horizon $[0, 1]$ into n uniform sub-intervals and apply flow-map inference sequentially over this grid. All experiments are conducted on a single NVIDIA RTX 3090 GPU. An anonymized implementation is available at `https://anonymous.4open.science/r/rmf-B2C0`.

Dataset. We use the FANTOM5 (Hon et al., 2017) dataset consisting of 100,000 transcription start sites (TSSs), following the same preprocessing and train/validation/test splits as prior work (Stark et al., 2024) (88,470/3,933/7,497). During training, we apply a random offset of up to ± 10 bp around each TSS, while validation and test splits use fixed windows.

Architecture. We adopt the same 1D CNN backbone as in prior promoter models (Stark et al., 2024; Davis et al., 2024), consisting of an initial embedding layer followed by 20 residual convolutional blocks (kernel size 9) with progressively increasing dilation. The model conditions on two time variables by embedding both the absolute time s and the time gap $(t - s)$ using Gaussian Fourier features, which are concatenated and injected into all time-conditioned layers. This doubles the time-embedding dimension and results in a modest parameter increase (13.27M $\rightarrow$ 14.65M). Depending on the parameterization, the output is either projected onto the tangent space (v-prediction) or mapped back to the manifold (x_1-prediction).

Training and objectives. Models are trained for 200 epochs with batch size 128 (138,400 steps total) using AdamW with learning rate 10^{-3}, zero weight decay, and global-norm gradient clipping at 1.0. For evaluation, we maintain an exponential moving average (EMA) of the model parameters with decay 0.9999 to reduce the variance induced by differential objectives. For the Eulerian and Lagrangian RMF objectives, we apply adaptive loss weighting with exponent $p = 0.5$ and clip neural-network derivatives with threshold 100.0; for this task, Lagrangian RMF is trained without the cyclic consistency loss. For the semigroup RMF objective, we set $w_{\text{semigroup}} = 5.0$ and use adaptive loss weighting ($p = 0.5$), except for the $s = r = t$ (flow-matching) case where adaptive weighting is disabled. For x_1-prediction, we down-weight losses near $s = 1$ using a time clipping threshold $\epsilon = 0.1$. The best checkpoint is selected based on validation MSE between the true promoter signal and the signal predicted by the Sei model conditioned on generated sequences.

Time sampling and interpolant. We sample time points from a log-normal distribution with parameters $\mu = -0.4$ and $\sigma = 1.0$. For objectives requiring two time points, samples are ordered accordingly, with 75% boundary samples following Geng et al. (2025). For the semigroup objective, we sample (s, t) as above and draw the intermediate time r uniformly from $[s, t]$. We use a linear geodesic interpolant throughout, following prior work (Davis et al., 2024).

F.2. Protein Backbone Design

We provide an anonymized implementation for the protein backbone experiments at `https://anonymous.4open.science/r/RMF-protein-71E1`.

SCOPe dataset. The Structural Classification of Proteins—extended (SCOPe) (Chandonia et al., 2022) organizes protein structures from the Protein Data Bank (PDB) (Berman et al., 2000; Burley et al., 2021) into domains according to structural similarity and evolutionary relationships, providing curated coordinate files and hierarchical labels (e.g., class, fold, superfamily, and family). Following Yim et al. (2023a), we use experimentally determined single-chain backbones with sequence lengths between 60 and 128 residues (3,938 examples) and evaluate on the protein monomer generation setting.

Baselines. We compare against three prior methods for backbone generation: GENIE (Lin & AlQuraishi, 2023), FrameDiff (Yim et al., 2023b), and FrameFlow (Yim et al., 2023a).

Model architecture. We adopt the FrameDiff architecture used in FrameFlow (Yim et al., 2023a), an SE(3)-equivariant model built around IPA blocks. We report the architecture variants (S/M/L) in Table A4. To adapt FrameDiff into a flow-map parameterization, we condition each IPA normalization layer via adaptive layer-normalization (AdaLN) scaling applied to the node embeddings.

Training setup. For both translations and rotations, we use a linear noise schedule with a geodesic interpolant. Training minimizes a weighted combination of (i) flow-matching losses on translations and rotations, (ii) a semigroup consistency

Table A4. **Architectural differences across IPA models.**

Model	Total # params	# of blocks	Node emb. dim	Edge emb. dim	Attn. heads
RMF/S	16.3 M	6	256	128	4
RMF/M	91.7 M	12	512	128	8
RMF/L	437.4 M	16	768	384	12

loss, and (iii) auxiliary geometric losses on backbone atom coordinates and local Cα–Cα distances, as commonly used in protein backbone generation (Yim et al., 2023a;b; Bose et al., 2023). Auxiliary losses are applied only at late times ($t > 0.75$). The final objective uses adaptive loss reweighting with exponent $p = 0.5$. Unless otherwise stated, we set the semigroup loss weight to 1.0, the translation loss weight to 2.0, and use x_1-prediction down-weighting with $\epsilon = 0.1$. We maintain an exponential moving average (EMA) of model parameters with decay 0.9999 and use the EMA weights for evaluation. We train with AdamW (learning rate 10^{-4}, no weight decay) for up to 1000 epochs (up to 1000K optimization steps), with gradient clipping and no learning-rate scheduling.

For flow-map training, we sample time points from the beta–uniform mixture proposed by Geffner et al. (2025b): $p(t) = 0.02\,\mathcal{U}[0,1] + 0.98\,\mathcal{B}(1.9, 1.0)$. To form a time interval, we draw two time points i.i.d. from $p(t)$ and sort them to obtain $s < t$. For the intermediate time r, we use the midpoint $r = (s+t)/2$.

Batching. To accommodate variable-length proteins, we construct mini-batches dynamically by enforcing both a maximum batch size of 80 sequences and a global complexity constraint $\sum_i L_i^2 \leq 4 \times 10^5$, where L_i is the length of sequence i. This yields stable memory usage across batches.

Hardware. RMF/S is trained using four NVIDIA RTX 3090 GPUs for 5 days. RMF/M is trained on a single NVIDIA B200 GPU for 10 days, and RMF/L is trained using eight NVIDIA B200 GPUs for 10 days.

Low-noise inference techniques with the flow map. In protein backbone generation with flow matching or diffusion models, inference-time heuristics are commonly used to improve designability. For example, FrameFlow and FoldFlow (Yim et al., 2023a; Bose et al., 2023) apply velocity scaling to the rotational components during sampling. Similarly, Proteina (Geffner et al., 2025b) reports that reducing the magnitude of injected noise at each diffusion step can be beneficial. Motivated by these findings, we adopt the low-noise inference technique of Xie et al. (2025). Instead of strictly following the flow map $\Phi^{\theta}_{s,t}$, we first recover an estimate of the data point and then re-introduce a controlled amount of noise:

$$x_t = t\hat{x}_1 + \eta(1-t)\epsilon, \quad \text{where } \hat{x}_1 = \Phi^{\theta}_{s,1}(x_s) \text{ and } \epsilon \sim \mathcal{N}(0, I). \tag{101}$$

When $\eta = 1$, this update resembles DDIM-style inference, whereas $\eta = 0$ yields a deterministic path with no added noise. In our experiments, we set $\eta = 0.45$ for rotations and $\eta = 1.0$ for translations; following Geffner et al. (2025b); Xie et al. (2025), $\eta = 0.45$ provided robust performance. Overall, this heuristic improves designability, with the largest gains in few-step sampling and for smaller model architectures.

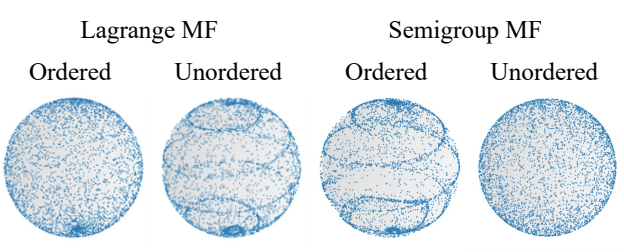


Figure A2. Effect of time sampling order. Comparison of Lagrangian (left) and Semigroup (right) objectives. Lagrangian RMF requires unordered intervals for convergence, whereas Semigroup RMF becomes unstable when trained with unordered intervals.

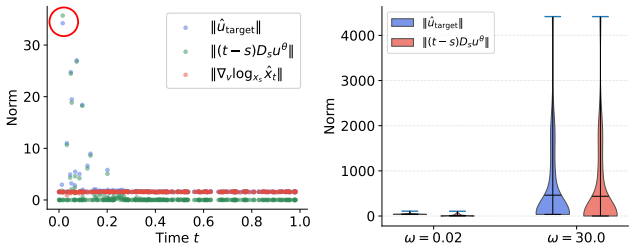


Figure A3. Training instability of Eulerian and Lagrangian RMF: (Left) Variance of the Eulerian RMF regression target. (Right) Reducing the Fourier frequency to $\omega = 0.02$ lowers the regression-target variance.

G. Additional Results

G.1. Empirical Evidence on Training Stabilization Techniques

In this section, we provide empirical evidence for the stabilization techniques introduced in Sec. 3.3. While the effectiveness of adaptive loss weighting was discussed in Sec. 4.1, here we focus on two other critical factors: time-sampling distributions and time-derivative control.

Effect of time sampling distributions. As illustrated in Fig. A2, the choice of time-sampling distribution is critical for the the stability and convergence of the learned flow map. While Eulerian MF remains largely insensitive to the temporal ordering of samples—typically using ordered pairs $s \leq t$, which cover half of the unit square—the other two objectives exhibit strict and contrasting requirements:

1. **Lagrangian MF** necessitates unordered time intervals. Training exclusively with ordered pairs ($s < t$) fails to capture the full dynamics, resulting in failure to converge to a valid flow map. This suggests that the Lagrangian formulation relies on the bidirectional information provided by sampling both $s < t$ and $t < s$ to regularize the path.
2. **Semigroup MF** shows the opposite behavior, where stability is tied to the sequential structure of the triplets. When trained with unordered intervals, the objective becomes highly unstable. The inclusion of an intermediate time r ($s < r < t$) reinforces the compositionality of the flow, and departing from this ordered structure leads to significant performance degradation observed in our sphere visualizations.

Time-derivative control. Differential objectives, such as the Eulerian and Lagrangian formulations, are particularly susceptible to instability arising from time-derivative terms. Fig. A3 (left) elucidates this phenomenon: the norm of the regression target explodes at certain time steps (highlighted by the red circle). This instability stems from the uncontrolled magnitude of the neural network's time derivative, $D_s u^\theta_{s,t}$, which significantly destabilizes the optimization process.

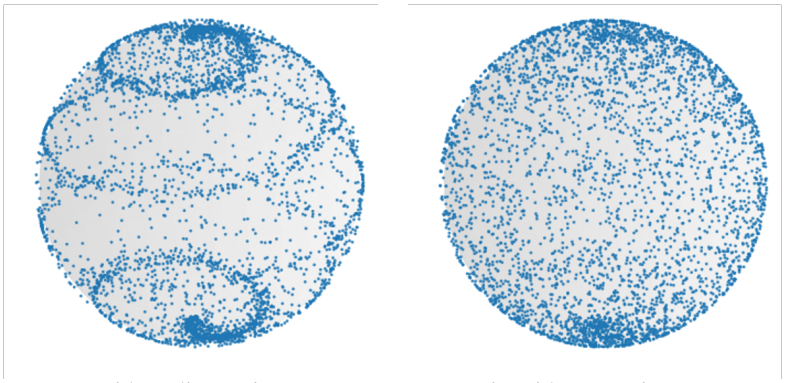
To mitigate this, we bound the derivative magnitude by adjusting the Fourier time embeddings. Since the derivative of a periodic embedding $\frac{d}{dt}\sin(\omega t) = \omega\cos(\omega t)$ scales linearly with the frequency ω, using high frequencies (e.g., $\omega = 30$) leads to high-variance regression targets. By adopting a lower frequency (e.g., $\omega = 0.02$), we effectively stabilize the training process. As shown in Fig. A3 (right), this modification drastically reduces the variance of the regression target, a finding that mirrors observations in consistency model training (Lu & Song, 2024) and proves essential for robust flow map learning.

G.2. Effect of Cyclic Consistency Regularizer on Lagrangian Objective

In this section, we analyze the role of the cycle-consistency regularizer in stabilizing the Lagrangian RMF objective, particularly in high-dimensional settings. As discussed in Sec. 3.1, the Lagrangian objective evaluates the regression loss at a model-predicted input $\hat{x}_s = \Phi^\theta_{t,s}(x_t)$. This introduces an additional source of error accumulation when the learned flow map deviates from invertibility, which can be particularly problematic in high-dimensional settings.

To examine this effect, we compare Lagrangian RMF trained *with* and *without* the proposed cycle-consistency regularizer:

$$\mathcal{L}_{\text{cyc}}(\theta) = \mathbb{E}_{x_t,s,t}\left[d_g(\Phi^\theta_{s,t}(\Phi^\theta_{t,s}(x_t)), x_t)^2\right]. \tag{102}$$



(a) With cyclic consistency *(b)* Without consistency

Figure A5. Effect of cyclic consistency regularization. Samples generated by Lagrangian RMF with and without the proposed cycle-consistency regularizer on a high-dimensional spherical helix dataset ($D = 512$).

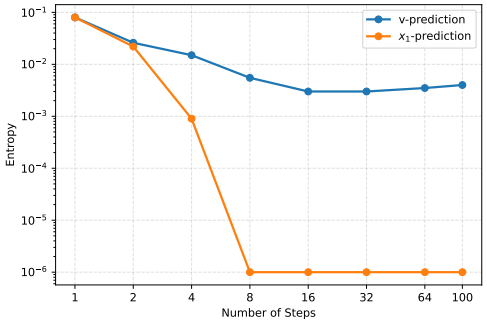


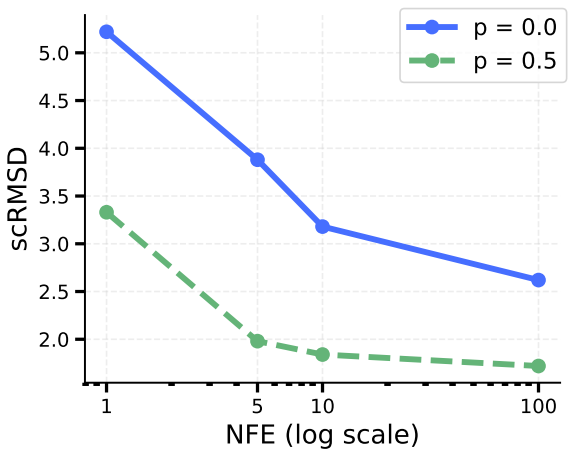
Figure A6. Inference steps vs. entropy. x_1-prediction achieves a significantly lower entropy compared to v-prediction.

All other training configurations are kept identical. We visualize the resulting samples on the spherical helix benchmark introduced in Sec. 4.1, where the data lie on a low-dimensional manifold embedded in a high-dimensional ambient space. As shown in Fig. A5, the cycle-consistency regularizer can be helpful in some settings, leading to a better approximation of the target flow map by Lagrangian RMF in high-dimensional regimes.

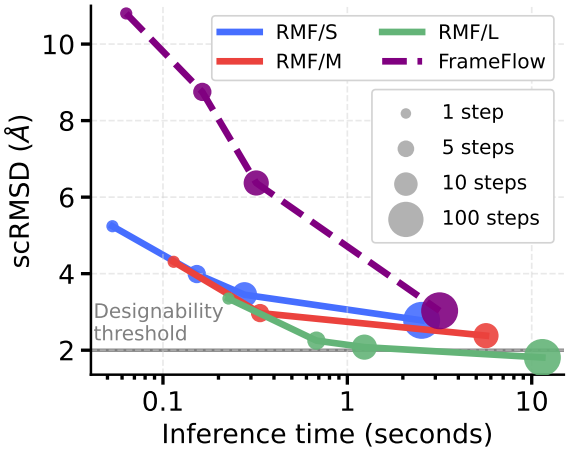
G.3. Parametrization Comparison in Promoter DNA Design

While x_1- and v-prediction achieve comparable standard metrics (e.g., MSE and k-mer correlation), they induce markedly different distributional sharpness. As shown in Fig. A6, x_1-prediction attains near-zero entropy (on the order of 10^{-6}), whereas v-prediction saturates at a substantially higher level (approximately 3×10^{-3}), indicating that x_1-prediction produces outputs closer to one-hot distributions.

In discrete sequence design, this difference is largely collapsed by the final $\arg\max$ discretization step and can therefore be hidden by downstream metrics. However, such increased sharpness may be beneficial in settings where small distributional differences directly affect geometry or structure, for example, protein backbone modeling.



(a) Effect of adaptive loss weighting on generation quality.



(b) Generation quality vs. inference time without inference-time techniques.

Figure A7. **Ablation on protein backbone generation.** (a) Adaptive loss weighting improves `scRMSD` across inference steps. (b) Removing inference-time heuristics reveals Pareto improvements from model scaling.

G.4. Ablations on Protein Backbone Experiments

Adaptive loss weighting. We study the effect of adaptive loss weighting in the protein backbone task, keeping all model architectures, training schedules, and optimization hyperparameters fixed. All results in Fig. A7a are reported for the L model without inference-time techniques. In Fig. A7a, we compare $p = 0.0$ (no weighting) and $p = 0.5$. Across inference step counts, $p = 0.5$ consistently yields lower `scRMSD`, indicating that adaptive weighting improves generation quality.

Model scaling without inference tricks. To isolate the contribution of inference-time techniques and examine the effect of model scaling, we repeat the protein backbone evaluation without applying any inference-time heuristics and report the results in Fig. A7b.

Without inference tricks (e.g., inference-time rotation velocity scaling), the FrameFlow baseline exhibits a substantial performance drop, achieving only 40% designability with an scRMSD of 3.03 even at 100 inference steps. Under the same setting and model size, our flow-map-based model shows clear improvements at few-step regimes (1, 5, and 10 steps), but its multi-step performance remains limited without scaling.

This observation suggests that strong multi-step performance is crucial for effective few-step generation. Motivated by this, we scale the model to the L variant, which significantly improves multi-step sampling, achieving an `scRMSD` of 1.59 and 77% designability at 100 steps. As the multi-step performance improves, few-step generation quality also improves accordingly, reaching `scRMSD` values of 1.98 and 1.84 at 5 and 10 steps, respectively. Overall, evaluating all methods without inference-time tricks makes the impact of model scaling more apparent. As shown in Fig. A7b, increasing model capacity yields consistent improvements in both generation quality and designability across inference budgets, yielding a clear Pareto improvement. A promising direction for future work is to incorporate the effect of inference-time heuristics directly into the learned dynamics (or training objective), so that their benefits do not rely on sampling-time adjustments. Notably, in the Euclidean setting, MeanFlow (Geng et al., 2025) has explored integrating classifier guidance into the model dynamics; analogously, one could aim to internalize common inference-time tricks and potentially accelerate the resulting guided dynamics.

G.5. Qualitative Results on Protein Backbone Generation

In Fig. A8, we visualize the protein backbones generated by our model across different inference steps and protein lengths. All samples were generated using the inference trick with a eta value of 0.45.

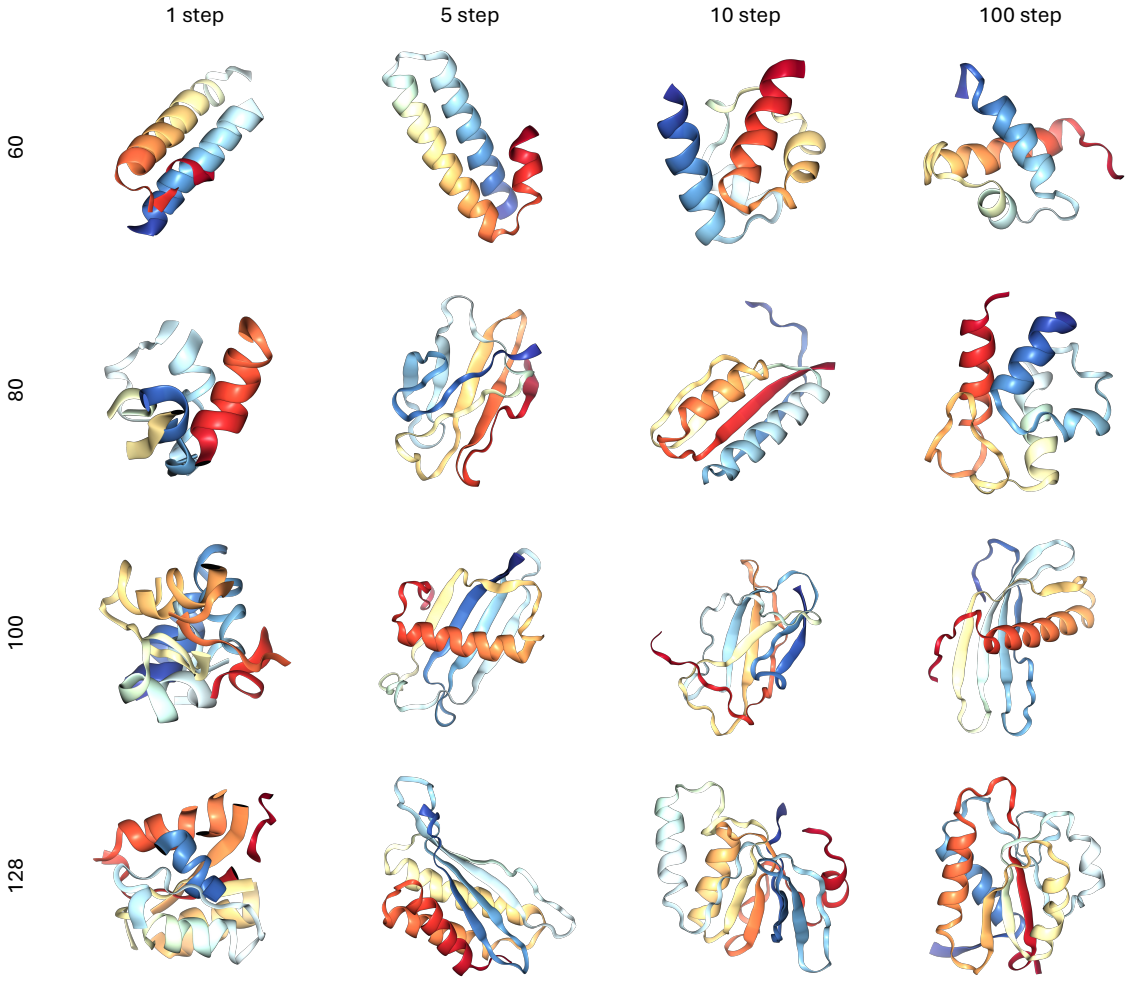


Figure A8. Visualization of generated protein backbones. We visualize the generated protein backbone from our model for steps of {1, 5, 10, 100} and sequence length {60, 80, 100, 128}.