SE(3)-MeanFlow: Few-Step Protein Backbone Generation on Lie Groups

Yikun Bai[1] Binghang Lu[2] Yikai Liu[3] Elaheh Akbari[4] Soheil Kolouri[4]
Linxuan Wang[5] Ping He[4] Shuchan Wang[6] Ruqi Zhang[7*]
Guang Lin[1,8*]

[1]Department of Mathematics, Purdue University
[2]School of Electrical and Computer Engineering, Purdue University
[3]Department of Biochemistry, Purdue University
[4]Department of Computer Science, Vanderbilt University
[5]Department of Statistics, Purdue University
[6]Department of Mathematics and Computer Science, Freie Universität Berlin
[7]Department of Computer Science, Purdue University
[8]School of Mechanical Engineering, Purdue University

Abstract

Generative modeling of protein backbones promises the de novo design of proteins with prescribed structural and functional properties. Existing diffusion and flow-matching models produce high-quality backbones on $\mathrm{SE}(3)^N$, but inference requires numerically integrating an ODE over hundreds of network evaluations, each involving a Lie group exponential map—a bottleneck for high-throughput design campaigns. We introduce **SE(3)-MeanFlow**, a few-step generative framework that extends MeanFlow from Euclidean space to the Lie group geometry of protein frames. Working natively in the Lie algebra $\mathfrak{so}(3)$ and in $\mathbb{R}^3$, we derive closed-form average-velocity identities for rotations and translations, giving simulation-free training targets. We further introduce an SE(3) α-Flow objective that removes the Jacobian–vector product from the rotation branch and serves as a warm-up stage, after which training switches to a small-t stabilized MeanFlow loss that is used for the remainder of pretraining and for rectification-based post-training. In protein backbone generation, SE(3)-MeanFlow matches or exceeds flow-matching baselines that use several times more sampling steps, and its advantage widens in the few-step regime, where rectification lets it lead at every matched budget—at a modest cost in diversity.

1 Introduction

Proteins are one of the basic building blocks of life. Their complex geometric structure enables specific inter-molecular interactions that allow for crucial biological functions—acting as catalysts in chemical reactions, transporters for molecules, and mediators of immune responses. With the emergence of computational techniques, it has become possible to rationally design novel proteins with desired structures that program their functions, opening pathways to solutions for long-standing global health challenges including influenza [Strauch et al., 2017], COVID-19 [Cao et al., 2020] and cancer immunotherapy [Silva et al., 2019].

*Corresponding authors: ruqiz@purdue.edu, guanglin@purdue.edu.

A protein backbone can be modeled as a sequence of N rigid bodies, one per residue, each associated with a frame under orientation-preserving rigid transformations —the special Euclidean group SE(3) [Jumper et al., 2021]. The full backbone is thus described by the product group $\mathrm{SE}(3)^N$, and the problem of de novo protein design reduces to sampling from a learned distribution over this space. Recent work has made substantial progress on generative modeling over $\mathrm{SE}(3)^N$. Diffusion-based methods such as FrameDiff [Yim et al., 2023b] and RFDiffusion [Watson et al., 2023] achieve strong designability, while flow matching approaches—FoldFlow [Bose et al., 2024] and FrameFlow [Yim et al., 2023a]—further improve training stability and flexibility by learning time-dependent vector fields on SE(3) in a simulation-free manner. Despite these advances, all existing SE(3) generative models share a common bottleneck: inference still requires numerically integrating an ODE over many steps, typically 100–500 network function evaluations (NFE). On SE(3), each step involves evaluating the network and applying the Lie group exponential map, making inference substantially more expensive than in Euclidean space. This limits practical deployment in high-throughput drug discovery pipelines, where millions of candidate structures must be generated per campaign.

Reducing the number of inference steps has been studied extensively in Euclidean generative modeling. Consistency models [Song et al., 2023, Song and Dhariwal, 2024] and progressive distillation [Salimans and Ho, 2022] compress inference into one or few steps, but require a pretrained teacher or carefully staged training curricula. MeanFlow [Geng et al., 2026a] offers a more principled and self-contained alternative. Rather than modeling the instantaneous velocity $v(t, x_t)$ as in standard flow matching [Lipman et al., 2022, Tong et al., 2023], MeanFlow introduces the notion of *average velocity* $u(s, t, x_t)$—the mean velocity of the flow trajectory over the interval $[s, t]$. A well-defined identity relating u and v is derived purely from the definition of average velocity via the product rule and the fundamental theorem of calculus. This identity yields a tractable, simulation-free training objective that uses only instantaneous velocity as supervision. At inference, the entire flow path is approximated in a single network evaluation $u_\theta(0, 1, x_1)$, enabling 1-NFE generation without distillation or pretraining. However, MeanFlow is formulated in Euclidean space $\mathbb{R}^d$ and does not account for the non-trivial geometry of SE(3). Naively lifting MeanFlow to a curved manifold is ill-posed: velocities at different points along the trajectory live in different tangent spaces, and their average requires parallel transport along the path.

Riemannian MeanFlow (denoted as RMF-PT) Zhong et al. [2026] addresses this by extending MeanFlow to general Riemannian manifolds, defining average velocity via parallel transport and deriving a corresponding Riemannian MeanFlow identity. While principled, this general-purpose framework does not exploit the specific algebraic structure of SE(3) as a Lie group. Another concurrent work, Riemannian MeanFlow (RMF) Woo et al. [2026], defines the average velocity in the Lie algebra; we compare with it theoretically in Section E.6 and empirically in Section 4.

In this work, we propose **SE(3)-MeanFlow**, a few-step generative model for de novo protein backbone design grounded in the Lie group structure of SE(3). Our central insight is that the decomposition $\mathrm{SE}(3) \cong \mathrm{SO}(3) \times \mathbb{R}^3$ allows the MeanFlow identity to be derived separately in the Lie algebra $\mathfrak{so}(3)$ and in $\mathbb{R}^3$, both admitting closed-form, simulation-free training targets without parallel transport.

Our main contributions are:

- **Integration-based SE(3)-MeanFlow with theory.** We propose an integration-based SE(3)-MeanFlow formulation conditioned on the current state (R_t, x_t), and derive closed-form average-velocity identities in the Lie algebra $\mathfrak{so}(3)$ and in $\mathbb{R}^3$, without parallel transport. We provide theoretical validation showing that the resulting training objectives are well-defined and that the corresponding losses are consistent with the intended MeanFlow targets on SE(3).

- **Stable training algorithms for proteins.** We introduce an SE(3) α-Flow objective together with practical numerical stabilization techniques that make training and few-step generation reliable for protein backbones.
- **Protein design results.** On the SCOPe benchmark, our approach matches or exceeds flow-matching baselines that use several times more sampling steps, with the largest gains in the few-step regime, in both pretraining and post-training settings, at a modest cost in diversity.

2 Background and Notations

2.1 Lie-group notation on SE(3)

In this work, the SE(3) space is a collection of rigid motions in

$$\mathrm{SE}(3) \cong \mathrm{SO}(3) \times \mathbb{R}^3,$$

where SO(3) denotes the space of 3×3 rotation matrices, which forms a Lie group:

$$\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : RR^\top = I,\ \det(R) = 1\}.$$

We follow Bose et al. [2024] and use the decoupled $\mathrm{SO}(3) \times \mathbb{R}^3$ geometry when defining metrics and losses.

A (continuous-time) flow on SO(3) can be defined by a curve $R_t \in \mathrm{SO}(3)$ satisfying the left-trivialized ODE

$$\dot{R}_t = R_t\, \Omega_t, \qquad \Omega_t \in \mathfrak{so}(3), \tag{1}$$

where $\mathfrak{so}(3)$ denotes the corresponding Lie algebra

$$\mathfrak{so}(3) = \{\Omega \in \mathbb{R}^{3\times 3} : \Omega^\top = -\Omega\}.$$

$\mathfrak{so}(3)$ is isomorphic to $\mathbb{R}^3$ space, and we use the hat/vee maps $(\cdot)^\wedge, (\cdot)^\vee$ to identify $\mathbb{R}^3 \leftrightarrow \mathfrak{so}(3)$ (e.g., $\omega \in \mathbb{R}^3 \mapsto \omega^\wedge \in \mathfrak{so}(3)$).

Given the angular velocity field $\Omega_t : [0, 1] \to \mathfrak{so}(3)$ along the trajectory, the solution of (1) can be written as

$$R_t = R_0\, \mathcal{T} \exp\Big(\int_0^t \Omega_\tau\, d\tau\Big),$$

where $\mathcal{T}\exp$ composes the infinitesimal rotations along time. Here exp denotes the (matrix) exponential map from the Lie algebra to the Lie group. In particular, if $\Omega_\tau \equiv \Omega$ is constant, then

$$R_t = R_0 \exp(t\Omega).$$

For a detailed discussion of SO(3)/SE(3) flows (including the closed-form/geodesic expressions under common metrics), see Appendix B[1].

[1]All appendices referenced in this paper are provided in the technical supplement.

2.2 Protein Backbone Parametrization

Following the setting in Bose et al. [2024], we parametrize the protein backbone as a sequence of N rigid bodies. Each residue $i \in [N]$ is associated with a frame $T_i = (R_i, x_i) \in \mathrm{SE}(3)$, where $R_i \in \mathrm{SO}(3)$ represents the orientation and $x_i \in \mathbb{R}^3$ denotes the position of the C_α atom.

The 3D coordinates of the backbone atoms $\{N, C_\alpha, C, O\}_i$ are recovered by applying the rigid transformation T_i to a set of idealized coordinates $\{N^*, C_\alpha^*, C^*, O^*\}$:

$$[N, C_\alpha, C, O]_i = T_i \circ [N^*, C_\alpha^*, C^*, O^*], \tag{2}$$

where $C_\alpha^* = (0, 0, 0)$ is fixed at the origin. For a self-contained review of SO(3)/SE(3) geometry and notation (including the definitions of $(\cdot)^\wedge$ and $(\cdot)^\vee$), see Appendix B–C.

2.3 MeanFlow in Euclidean space

One limitation of the above flow-matching method is that, to guarantee accuracy, we generally require the step size $\frac{1}{T}$ to be sufficiently small, equivalently requiring large inference steps T. To address this issue, in Euclidean space, Geng et al. [2026a,b] propose the MeanFlow method. In particular, given a probability path p_t generated by velocity v_t, it defines the average velocity:

$$(t-s)v^{\mathrm{avg}}(s,t,x_t) = \int_s^t v(\tau, x_\tau)d\tau.$$

It yields the training loss:

$$\mathbb{E}\left[\left\| v_\theta^{\mathrm{avg}}(s,t,x_t) + (t-s)\,\mathrm{sg}\Big(\frac{d}{dt}\hat{v}^{\mathrm{avg}}(s,t,x_t)\Big) - v(t,x_t)\right\|^2\right]. \tag{3}$$

where the randomness is given by $t, s \sim \mathrm{Unif}([0,1]), s \le t$ and $(x_0, x_1) \sim q$ for some joint distribution q. By default, q is an independent coupling between p_{data} and p_{prior},

$$\frac{d}{dt}\hat{v}^{\mathrm{avg}}(s,t,x_t) = \frac{\partial}{\partial t}v_\theta^{\mathrm{avg}}(s,t,x_t) + \nabla_x v_\theta^{\mathrm{avg}}(s,t,x_t)\cdot v_t. \tag{4}$$

In addition, v_t can be replaced $v_\theta^{\mathrm{avg}}(t,t,x_t)$ in Geng et al. [2026b].

Here $x_t = (1-t)x_0 + tx_1$, $v(t,x_t) = x_1 - x_0$, and sg is the stop-gradient operator. Intuitively, the integration is estimated by the parameterized model during the training process; thus, in the inference step, it yields a T-step generation, where $T \in \mathbb{N}$:

$$\begin{aligned} &\hat{x}_s = x_t - \frac{1}{T}v_\theta^{\mathrm{avg}}(s,t,x_t) \\ &t = 1 - i/T, s = t - 1/T, i = 0, 1, ...(T-1). \end{aligned} \tag{5}$$

When we set $T = 1$, it yields a one-step generation.

3 Method: MeanFlow in SE(3) space

Since $\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$, the $\mathbb{R}^3$ component reduces to the conventional Euclidean MeanFlow; hence, we focus on introducing the MeanFlow on SO(3) below.
Let a distribution path $\{p_t \in \mathcal{P}(\mathrm{SO}(3)) : t \in [0,1]\}$ be generated by a velocity field $v_t : [0,1] \times \mathrm{SO}(3) \to \mathcal{T}\mathrm{SO}(3)$. We define the *average velocity* over $[s,t]$ through the time-ordered exponential

$$\exp\big((t-s)\,\Omega^{\mathrm{avg}}(s,t,R_t,x_t)\big) := \mathcal{T}\exp\Big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau\Big). \tag{6}$$

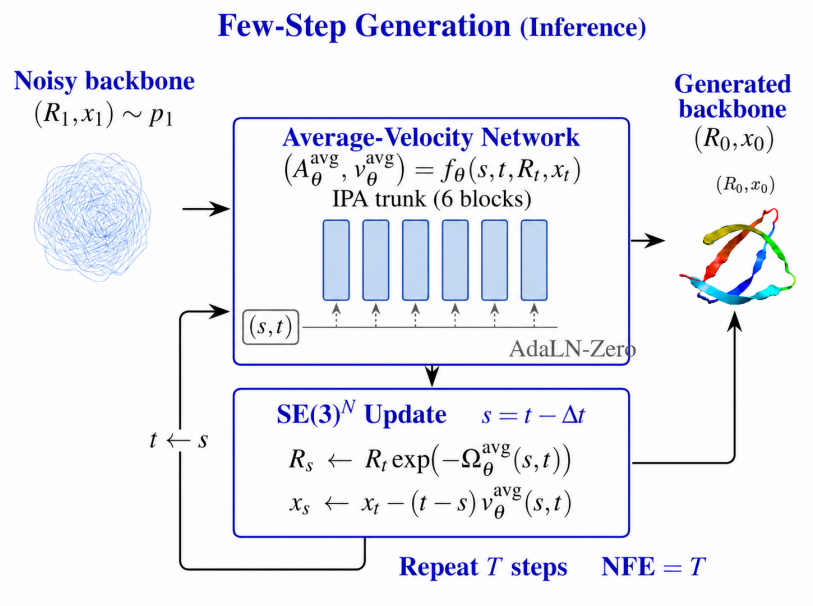

Figure 1: Few-step inference with SE(3)-MeanFlow. The two-time network f_θ (see Appendix N) predicts the average velocity over $[s,t]$ rather than the instantaneous velocity at t, so one evaluation advances the state by the full step—a Lie group exponential on SO(3), a Euclidean update on $\mathbb{R}^3$—giving NFE $= T$. The objective that makes this accurate is Proposition 1 and its JVP-free α-Flow surrogate (Section 3.3).

We denote the instantaneous (body-frame) angular velocity (vector form) by

$$\omega_t := \Omega(t, R_t, x_t)^\vee \in \mathbb{R}^3. \tag{7}$$

Analogous to Euclidean MeanFlow, differentiating (6) gives an identity linking the average velocity, its trajectory derivative, and the instantaneous velocity.

Proposition 1. *Differentiating* (6) *with respect to* t *yields*

$$J\big((t-s)A^{avg}\big)\Big(A^{avg}(s,t,R_t,x_t) + (t-s)\tfrac{d}{dt}A^{avg}\Big) = \omega_t, \tag{8}$$

where $A^{avg} = (\Omega^{avg})^\vee$,

$$\frac{d}{dt}A^{avg} = \frac{\partial}{\partial t}A^{avg} + \langle \nabla_R A^{avg}, \dot{R}_t\rangle + \langle \nabla_x A^{avg}, \dot{x}_t\rangle, \tag{9}$$

and the right Jacobian $J : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ *is*

$$J(A) := I - \frac{1-\cos\|A\|_2}{\|A\|_2^2}A^\wedge + \frac{\|A\|_2 - \sin\|A\|_2}{\|A\|_2^3}(A^\wedge)^2, \tag{10}$$

with $\|A\|_2$ *the* ℓ_2 *norm of* $A \in \mathbb{R}^3$*; as* $\|A\| \to 0$, $J(A) \to I$ *(we use its Taylor expansion there).*

This proposition is split into Propositions 3 and 4 in Appendix F.1, with proofs.

3.1 Model parametrization and training loss

The network exposes two interchangeable heads: an *endpoint* (x-)head predicting the clean state $(\hat{R}_0^\theta, \hat{x}_0^\theta)$, and an *average-velocity* (u-)head $A_\theta^{\text{avg}} : [0,1]\times[0,1]\times \text{SO}(3)\times\mathbb{R}^3 \to \mathbb{R}^3$ and v_θ^{avg}, with

$\Omega_\theta^{\text{avg}} = (A_\theta^{\text{avg}})^\wedge$. The two carry the same information, related by the invertible map

$$A_\theta^{\text{avg}} = \tfrac{1}{t}\log\big((\hat{R}_0^\theta)^\top R_t\big)^\vee \iff \hat{R}_0^\theta = R_t \exp\big(-t\, A_\theta^{\text{avg}\wedge}\big),$$
$$v_\theta^{\text{avg}} = \tfrac{1}{t}\big(x_t - \hat{x}_0^\theta\big) \iff \hat{x}_0^\theta = x_t - t\, v_\theta^{\text{avg}},$$

which, on the constant-velocity (x-prediction) geodesic, is consistent with the $(t-s)$-normalized average of (6). The losses below regress the velocity head; the endpoint head is produced at inference.

Based on (8), we derive two equivalent mean flow training losses:

$$\mathcal{L}^{J}_{\text{SO(3)}} := \mathbb{E}_{\substack{s<t \\ q(R_0,R_1)}}\Big[\big\|\omega_\theta - \omega_t\big\|_2^2\Big], \tag{11}$$
$$\omega_\theta = \text{sg}\big(J((t-s)A_\theta^{\text{avg}})\big)\Big(A_\theta^{\text{avg}} + (t-s)\,\text{sg}(\tfrac{d}{dt}A_\theta^{\text{avg}})\Big),$$
$$\mathcal{L}^{J^{-1}}_{\text{SO(3)}} := \mathbb{E}\left[\|A_\theta^{\text{avg}} - \text{sg}(A_\theta^{tgt})\|^2\right] \tag{12}$$
$$A_\theta^{tgt} = J^{-1}((t-s)A_\theta^{\text{avg}})\omega_t - (t-s)\frac{d}{dt}A_\theta^{\text{avg}}$$

where all model arguments are (s, t, R_t, x_t) and $\frac{d}{dt}A_\theta^{\text{avg}}$ is given by (9). Loss (12) is well-defined since J is invertible on the principal branch $\|(t-s)A_\theta^{\text{avg}}\|_2 \le \pi$. We refer to Appendix F.1 for details.

Manifold derivatives via Euclidean JVPs. Although $R_t \in \text{SO}(3)$ is manifold-valued, $\frac{d}{dt}A_\theta^{\text{avg}}$ is a directional derivative along the interpolation trajectory (R_t, x_t), which we evaluate with a standard Euclidean Jacobian–vector product (e.g. `torch.jvp`) through the chosen matrix representation of R_t. A formal equivalence is given in Appendix F.1, Proposition 6.

Proposition 2 (Correctness of the MeanFlow objective, informal)**.** *If $\mathcal{L}_{SO(3)}$ in (11) or (12) is zero, the model recovers the correct relative rotations along the path:* $\exp\big(((t-s)A_\theta^{avg}(s,t,R_t,x_t))^\wedge\big) = R_s^\top R_t$ *for all* $0 \le s < t \le 1$.

A formal statement and proof are in Proposition 5 (Appendix).

Inference. We step backwards using the learned average flow,

$$R_s \leftarrow R_t \exp\big(-(t-s)\,\Omega_\theta^{\text{avg}}(s,t,R_t,x_t)\big), \qquad t = 1, \dots, \tfrac{1}{T},$$

and analogously $x_s \leftarrow x_t - (t-s)\, v_\theta^{\text{avg}}$.See Figure 1. Reported results use a variant of this update, the exponential rotation schedule of Appendix I.

3.2 Practical implementation: $\text{SE}(3)^N$ adaptation

Each protein sample is $R^N \in \mathbb{R}^{N\times 3\times 3}$, $x^N \in \mathbb{R}^{N\times 3}$, with N the number of residues. The model is $f_\theta(s,t,R_t^N,x_t^N) = [A_\theta^{\text{avg},N}; v_\theta^{\text{avg},N}] \in \mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times 3}$, where the i-th block $A_{\theta,i}^{\text{avg},N} \in \mathbb{R}^3$ parametrizes the $\mathfrak{so}(3)$ component via $\Omega_{\theta,i}^{\text{avg}} = (A_{\theta,i}^{\text{avg},N})^\wedge$. The loss (11) (or (12)) is summed over residues.

3.3 $\text{SE}(3)$ α-Flow: rotation formulation and MeanFlow limit

The differential target (8) requires the trajectory derivative $\frac{d}{dt}A_\theta^{\text{avg}}$ via a JVP, which could be fragile on the rotation branch. We therefore also adopt an α-Flow formulation Zhang et al. [2025] that replaces this derivative by a two-segment construction using only forward evaluations of A_θ^{avg}.

Given $0 \le s < t \le 1$ and a ratio $\alpha \in (0, 1]$, insert an intermediate time $m = \alpha s + (1-\alpha)t$ with step $\delta := t - m = \alpha(t-s)$, splitting $[s, t]$ into a *far* segment $[s, m]$ and a *near* segment $[m, t]$. With $D(a, b) := R_a^\top R_b \in \mathrm{SO}(3)$ the accumulated relative rotation, the segments compose *multiplicatively* (rather than additively as in Euclidean space),

$$D(s,t) = D(s,m)\, D(m,t) \qquad \text{(Appendix J, Prop. 7).}$$

We anchor the near factor to data and bootstrap the far factor from the model. Stepping back from R_t along the data angular velocity $\omega_t := \Omega(t, R_t, x_t)^\vee$ gives the intermediate state $R_m = R_t \exp(-\delta\, \omega_t^\wedge)$ and the near factor $D(m,t) = \exp(\delta\, \omega_t^\wedge)$; a stop-gradient model query at (s, m, R_m, x_m) returns $A_m := A_\theta^{\mathrm{avg}}(s, m, R_m, x_m)$ and the far factor $D(s,m) = \exp((m-s)A_m^\wedge)$. Composing and mapping back to the Lie algebra gives the target average generator over $[s, t]$:

$$A_{\mathrm{tgt}}^{\mathrm{avg}}(s,t) = \frac{1}{t-s} \log\Big(\underbrace{\exp\big((m-s)A_m^\wedge\big)}_{D(s,m)\ \text{(model)}}\ \underbrace{\exp\big(\delta\, \omega_t^\wedge\big)}_{D(m,t)\ \text{(data)}} \Big)^\vee. \tag{13}$$

The scalar $\frac{1}{t-s}$ must stay outside the $\log(\exp \cdot \exp)$: since A_m and ω_t do not commute, folding it in would corrupt the Baker–Campbell–Hausdorff cross term and no longer give the generator of $D(s,t)$. We regress

$$\mathcal{L}_{\mathrm{rot}}^{\alpha} = \frac{1}{\alpha} \sum_{i=1}^{N} \big\| A_{\theta,i}^{\mathrm{avg}}(s,t,R_t,x_t) - \mathrm{sg}\big(A_{\mathrm{tgt},i}^{\mathrm{avg}}\big) \big\|_2^2, \tag{14}$$

which uses only forward evaluations (one log, two exps; no JVP). The numerically stabilized form and the abelian translation branch — where composition reduces to a convex combination of velocities as in the Euclidean α-Flow Zhang et al. [2025] — are given in Appendix J.3.

The ratio α interpolates between flow matching and MeanFlow: at $\alpha = 1$ the far factor vanishes ($m = s$) and $A_{\mathrm{tgt}}^{\mathrm{avg}} = \omega_t$, while as $\alpha \to 0$ a first-order BCH expansion recovers the differential MeanFlow objective (8) in gradient (Appendix J.5).

4 Experiments

We evaluate our method on unconditional protein backbone generation and compare it against recent diffusion- and flow-based baselines. Our experiments are designed to assess both generation quality and sampling efficiency, with a particular focus on few-step generation. To this end, we report performance under different numbers of sampling steps, allowing us to examine how well each method maintains designability, diversity, and novelty as the computational budget is reduced.

4.1 Datasets

Following prior works Yue et al. [2025], Woo et al. [2026], we conduct experiments on the SCOPe dataset Chandonia et al. [2022], Yim et al. [2023a], which consists of 3,673 preprocessed protein backbones with residue lengths between 60 and 128. We refer to Figure 9 in the Appendix for a visualization of the backbone length-frequency distribution.

4.2 Baselines

We compare recent flow-based backbone generators: FrameFlow Yim et al. [2023a], QFlow and ReQFlow Yue et al. [2025], and Riemannian MeanFlow (RMF) Woo et al. [2026]. Among these,

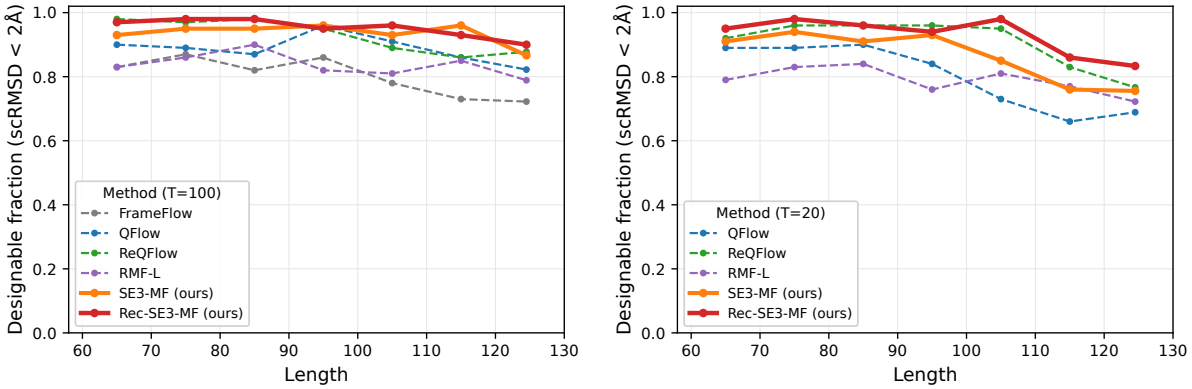

Figure 2: Designable fraction versus length (top: T=100, bottom: T=20).

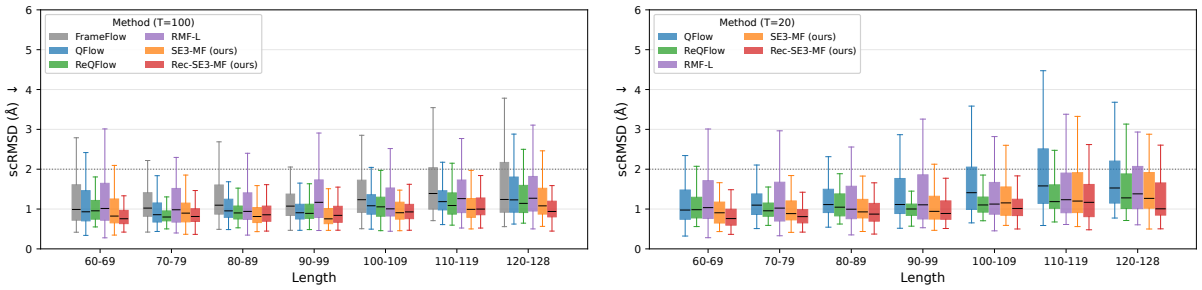

Figure 3: scRMSD distributions (top: T=100, bottom: T=20).

ReQFlow and RMF are most closely related to our approach, as they also target efficient few-step or accelerated generation. As a representative of earlier (pre-2024) diffusion- and flow-matching methods we include FrameFlow, which demonstrated strongest generation capacity on SCOPe; we discuss the earlier baselines (e.g. FrameDiff, Genie) in Appendix E.1.

4.3 Implementation Details

For experiments on SCOPe, we use publicly available checkpoints when possible to reproduce baseline results under a consistent evaluation protocol. All experiments are conducted using 4 H-100 GPUs. Unless otherwise stated, we follow the evaluation settings used in prior work Yue et al. [2025], including the same datasets, sampling protocols, and evaluation metrics, to ensure a fair comparison across methods.

4.4 Training details

Model and trainer. We implement our method within the public QFlow/ReQFlow codebase Yue et al. [2025] and inherit its overall training pipeline, adapting it in the following respects to fit the MeanFlow model.

(i) Representation and interpolation. QFlow/ReQFlow parametrize rotations by unit quaternions; we instead work directly with rotation matrices under the decoupled $\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$ representation of Yim et al. [2023b]. Accordingly, we build the data–noise interpolation and the mini-batch coupling with our own SE(3) geodesic interpolant and optimal-transport (OT) coupling in rotation-matrix form (Appendix D; global-OT coupling in Eq. (95)), rather than the quaternion interpolation of

Steps	Method	Designability			Diversity TM ↓	Novelty TM ↓
		Fraction ↑	scRMSD ↓	scTM ↑		
500 / 100	FrameFlow (500)	0.849	1.439 ± 1.137	0.879 ± 0.084	0.369	<u>0.654</u>
	FrameFlow (100)	0.803	1.576 ± 1.367	0.872 ± 0.090	<u>0.360</u>	**0.638**
	QFlow (500)	<u>0.900</u>	<u>1.271 ± 1.113</u>	<u>0.897 ± 0.078</u>	0.399	0.720
	QFlow (100)	0.885	1.319 ± 1.018	0.889 ± 0.077	0.393	0.699
	RMF (100)	0.832	1.425 ± 1.145	0.885 ± 0.081	**0.343**	0.719
	SE3MF (100)	**0.936**	**1.106** ± 0.870	**0.909** ± 0.067	0.415	0.736
50	QFlow	<u>0.870</u>	<u>1.432 ± 1.218</u>	0.882 ± 0.080	<u>0.379</u>	**0.684**
	RMF	0.825	1.446 ± 1.189	<u>0.885 ± 0.081</u>	**0.344**	<u>0.717</u>
	SE3MF	**0.906**	**1.246** ± 1.190	**0.902** ± 0.078	0.413	0.722
20	QFlow	0.778	1.703 ± 1.658	0.853 ± 0.098	<u>0.377</u>	**0.648**
	RMF	<u>0.806</u>	<u>1.554 ± 1.438</u>	<u>0.877 ± 0.091</u>	**0.345**	0.722
	SE3MF	**0.867**	**1.369** ± 1.144	**0.883** ± 0.087	0.408	<u>0.703</u>
10	QFlow	0.559	2.691 ± 2.263	0.773 ± 0.144	<u>0.379</u>	**0.615**
	RMF	**0.778**	**1.641** ± 1.432	**0.870** ± 0.088	**0.346**	0.713
	SE3MF	<u>0.728</u>	<u>1.940 ± 1.741</u>	<u>0.832 ± 0.118</u>	0.402	<u>0.659</u>

Table 1: Unconditional protein backbone generation on SCOPe, grouped by sampling budget. Baselines: FrameFlow Yim et al. [2023a], QFlow Yue et al. [2025], Riemannian MeanFlow (RMF) Woo et al. [2026]. Within each budget group, best in **bold** and second-best <u>underlined</u>.

QFlow.

(ii) Model. We keep the ReQFlow IPA trunk but make it consume a rotation-matrix state $(R_t, x_t) \in \mathrm{SE}(3)^N$ and condition on *two* times (s, t)—via a shared two-time embedding and a per-block AdaLN-Zero gate—and predict endpoints $(\hat{R}_0^\theta, \hat{x}_0^\theta)$ (Appendix N).

(iii) Objective. In place of the (V-)QFlow flow-matching loss, we train with our SE(3) MeanFlow objective (Section 3) and its α-Flow variant (Section 3.3); derivations and the stable implementation are in Appendix F.1 and M. The time sampler is the two-time extension ($s \le t$) of the QFlow sampler, with the marginal schedule of t left unchanged, and $s \mid t \sim \mathcal{U}[t_{\min}, t]$ with $t_{\min} = 10^{-6}$ (Table 7). All remaining pipeline components are inherited from the QFlow/ReQFlow codebase.

Stable training. The differential MeanFlow target requires a time derivative obtained via a Jacobian–vector product (JVP), which can be numerically brittle through the backbone network. We use two remedies: the JVP-free α-Flow objective (Section 3.3) as a warm-up, and a small-t stabilized form of the JVP-based loss (Appendix I, Algorithm 3).

Pre-training (two stages). Stage 1 is the α-Flow warm-up on $\mathrm{SE}(3)^N$ (Section 3.3; Appendix M.1); Stage 2 switches to the endpoint+MeanFlow objective (Appendix M.2).

Post-training. We further apply a rectification (self-reflow) stage following ReQFlow's rectified-flow strategy, keeping the Stage-2 MeanFlow objective (Appendix M.3). We denote the resulting model RecSE3MF.

Steps	Method	Designability			Diversity TM ↓	Novelty TM ↓
		Fraction ↑	scRMSD ↓	scTM ↑		
100	ReQFlow	0.946	1.120 ± 0.635	0.901 ± 0.062	**0.420**	**0.695**
	RecSE3MF	**0.968**	**0.974** ± 0.620	**0.920** ± 0.057	0.448	0.734
50	ReQFlow	0.946	1.131 ± 0.633	0.900 ± 0.062	**0.418**	**0.689**
	RecSE3MF	**0.970**	**0.941** ± 0.483	**0.923** ± 0.048	0.452	0.728
20	ReQFlow	0.910	1.270 ± 0.736	0.883 ± 0.074	**0.416**	**0.678**
	RecSE3MF	**0.929**	**1.133** ± 1.314	**0.903** ± 0.073	0.454	0.726
10	ReQFlow	0.837	1.552 ± 1.113	0.846 ± 0.092	**0.414**	**0.662**
	RecSE3MF	**0.894**	**1.269** ± 1.040	**0.891** ± 0.084	0.453	0.710

Table 2: **Post-train comparison on SCOPe.** Within each budget, best in **bold**. Training budgets are in Table 4.

4.5 Evaluation metrics and settings

We evaluate generated protein backbones with four metrics, following prior work [Yue et al., 2025]: designability, diversity, novelty, and efficiency. For each chain length N (from 60 to 128) we generate 10 backbones. Designability is the primary measure of sample quality and assesses whether a generated backbone admits an amino-acid sequence that folds back into a consistent structure. For each backbone we design 8 sequences with ProteinMPNN [Dauparas et al., 2022], predict the folded structure of each with ESMFold [Lin et al., 2023], and take the minimum self-consistency RMSD (scRMSD) over the 8 designs. A backbone is deemed *designable* when this scRMSD is at most 2 Å. We report the fraction of designable backbones (per length, averaged over lengths), denoted Fraction, the mean scRMSD, and the mean self-consistency TM-score (scTM), where higher scTM and Fraction and lower scRMSD are better.

To assess distributional properties, we measure structural diversity and novelty over the designable subset. Diversity is the pairwise TM-score [Zhang and Skolnick, 2004] among designable structures of the same length, averaged across lengths, where lower values indicate a less redundant set. Novelty compares each designable sample against the Protein Data Bank (PDB) with Foldseek [van Kempen et al., 2022] and records the maximum TM-score to the retrieved structures; the average of these maxima summarizes similarity to known proteins, with lower values indicating greater novelty. Finally, we quantify efficiency by the number of sampling steps used to generate the backbones.

4.6 Results and Discussion

SE(3)-MeanFlow (SE3MF) consistently improves few-step protein backbone generation. Across the 20–100-step regime, it achieves the strongest designability among pretrained baselines, ranking first in designable fraction, scRMSD, and scTM (Table 1). The gains are preserved across protein lengths (Figures 2 and 3), indicating that the improvement is not restricted to short or structurally simple backbones. At 20 steps, SE(3)-MeanFlow retains a designable fraction of 0.867, compared with 0.806 for RMF and 0.778 for QFlow, demonstrating a strong designability–efficiency trade-off under substantial step reduction; Figure 4 shows designable backbones generated at this budget.

These results support the effectiveness of specializing MeanFlow to the Lie-group structure of protein frames. Empirically, the complete formulation maintains strong designability as the sampling budget is reduced while preserving stable local backbone geometry, with Cα-validity remaining near 0.97 from 100 to 10 steps (Table 3). The secondary-structure statistics are likewise stable across

Method	Inf Steps	α-Helix	β-Strand	Cα-valid ↑
FrameFlow	500	0.450 ± 0.323	0.248 ± 0.218	0.962 ± 0.041
	100	0.465 ± 0.308	0.234 ± 0.212	0.987 ± 0.014
QFlow	100	0.502 ± 0.302	0.208 ± 0.206	0.993 ± 0.012
	50	0.469 ± 0.291	0.223 ± 0.204	0.996 ± 0.008
	20	0.476 ± 0.273	0.201 ± 0.184	0.894 ± 0.046
	10	0.461 ± 0.246	0.168 ± 0.160	0.831 ± 0.061
RMF	100	0.346 ± 0.232	0.278 ± 0.179	0.996 ± 0.007
	50	0.342 ± 0.232	0.279 ± 0.178	0.996 ± 0.008
	20	0.340 ± 0.230	0.277 ± 0.178	0.996 ± 0.008
	10	0.341 ± 0.229	0.276 ± 0.173	0.996 ± 0.008
SE3MF	100	0.536 ± 0.339	0.200 ± 0.238	0.971 ± 0.022
	50	0.517 ± 0.336	0.213 ± 0.236	0.971 ± 0.022
	20	0.505 ± 0.324	0.210 ± 0.224	0.970 ± 0.023
	10	0.497 ± 0.299	0.193 ± 0.203	0.975 ± 0.020
ReQFlow	100	0.511 ± 0.312	0.224 ± 0.214	0.993 ± 0.009
	50	0.509 ± 0.311	0.223 ± 0.212	0.986 ± 0.014
	20	0.507 ± 0.307	0.214 ± 0.212	0.860 ± 0.066
	10	0.502 ± 0.310	0.206 ± 0.209	0.712 ± 0.105
RecSE3MF	100	0.559 ± 0.357	0.207 ± 0.249	0.979 ± 0.017
	50	0.559 ± 0.360	0.206 ± 0.249	0.979 ± 0.017
	20	0.571 ± 0.354	0.197 ± 0.246	0.978 ± 0.017
	10	0.558 ± 0.359	0.198 ± 0.245	0.979 ± 0.017

Table 3: Evaluation scores at different sampling steps. Methods follow Table 1 and Table 2.

sampling budgets (Figure 10), suggesting that accelerated generation does not substantially alter the structural composition of the samples.

Rectification further strengthens the aggressive few-step regime. RecSE3MF outperforms ReQFlow on the three designability metrics at nearly all sampling budgets and reaches a designable fraction of 0.894 at 10 steps (Table 2). This suggests that rectification and MeanFlow play complementary roles: rectification simplifies the transport paths, while MeanFlow learns accurate finite-interval motion along those paths.

The improved designability comes at a cost in coverage. Both SE(3)-MeanFlow and RecSE3MF have higher diversity TM-scores than the coverage-oriented baselines (lower is better), and their novelty is weakest at large budgets (0.736 at 100 steps, against 0.719 for RMF); the gap narrows as the budget falls, with novelty second-best in its group at 20 and 10 steps. Diversity is nearly flat in the number of steps (0.415 to 0.402), so the concentration reflects the learned model rather than step reduction. Overall, SE(3)-MeanFlow advances the designability–efficiency frontier while preserving stable geometric and structural statistics under substantially reduced sampling budgets.

Training budget and model size. Table 4 compares model size, training budget and per-step sampling cost. Our network is the ReQFlow/QFlow trunk with a small (0.33%) AdaLN addition, hence comparable in size to QFlow and checkpoint-compatible with it (Appendix N), and $\sim$26$\times$ smaller than RMF; its total budget is on the same order as the flow-matching baselines from the same codebase, and far below RMF's $\sim$598k steps (their cap is 1000k). We conjecture that part of this gap reflects RMF's semigroup consistency objective rather than its model size alone:

Method	Params	Train Steps	Time/step (ms) ↓
Pre-trained			
FrameFlow	16.8M	177k	57
QFlow	16.8M	90k	43
RMF	437M	598k	454
SE3MF (ours)	16.8M	106k	43
Post-trained (rectification)			
ReQFlow	16.8M	90k + 4.6k	43
RecSE3MF (ours)	16.8M	106k + 6.5k	43

Table 4: Model size, training budget, and sampling cost. Post-training steps are given as pre-training + rectification. Time/step is wall-clock per sampling step on a single NVIDIA H100 80GB (single process, batch 10, length 128, fp32); each method runs one network forward per step, so total sampling time ≈ steps × time/step.

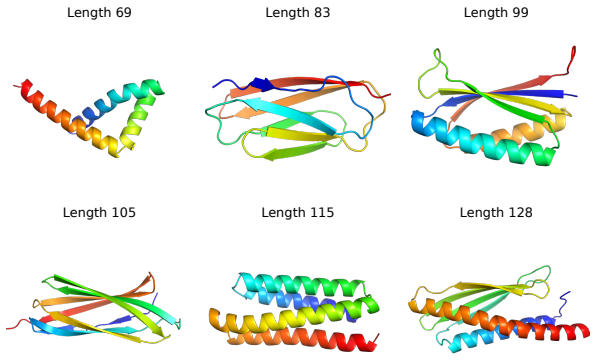

Figure 4: Qualitative samples on SCOPe generated at different sampling budgets. See Appendix for additional diagnostics.

straightening the transport paths enough for few-step sampling appears to demand a large budget, and this cost grows with the number of residues. Appendix O.2 tests this by fixing the model, the initialization and the training budget, and swapping only the objective; under this control the semigroup target does not reach the few-step regime while ours does.

On the low-dimensional $\mathrm{SO}(3)^2$ benchmark of Appendix H, a second controlled ablation in which only the loss varies, every average-velocity method reaches the few-step regime under a much smaller budget.

Remark 1. Epoch counts are not comparable across methods, as the data loaders differ, whereas each optimizer step processes the same amount of data. We therefore report budgets in optimizer steps; see Appendix O for details.

5 Conclusion

We introduced **SE(3)-MeanFlow**, a few-step generative framework on $\mathrm{SE}(3)^N$. Exploiting the decoupled $\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$ structure, we derive simulation-free average-velocity objectives directly in the Lie algebra, avoiding parallel transport, and stabilize training with a JVP-free α-Flow

warm-up and a small-t MeanFlow scheme supporting both training from scratch and rectification-based post-training. On SCOPe, SE(3)-MeanFlow advances the designability–efficiency frontier while using fewer sampling steps and a smaller model. Its main limitation is a modest reduction in diversity; future work will seek objectives that better balance coverage and designability, extend to conditional and motif-scaffolding tasks, and push toward one-step generation.

Acknowledgments

Y.B. was supported in part by the Purdue Institute for Physical AI (IPAI) Postdoctoral Fellows Program. G.L. would like to thank the support of National Science Foundation (DMS-2533878, DMS-2053746, DMS-2134209, ECCS-2328241, CBET-2347401 and OAC-2311848), and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program, under the "Uncertainty Quantification for Multifidelity Operator Learning (MOLUcQ)" project (Project No. 81739), DE-SC0023161, the SciDAC LEADS Institute, and DOE–Fusion Energy Science, under grant number: DE-SC0024583.

References

Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2024.

Longxing Cao, Inna Goreshnik, Brian Coventry, James Brett Case, Lauren Miller, Lisa Kozodoy, Rita E Chen, Lauren Carter, Alexandra C Walls, Young-Jun Park, et al. De novo design of picomolar sars-cov-2 miniprotein inhibitors. *Science*, 370(6515):426–431, 2020.

John-Marc Chandonia, Lindsey Guan, Shiangyi Lin, Changhua Yu, Naomi K Fox, and Steven E Brenner. 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.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 30467–30476, 2026b.

Andrew J Hanson. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, pages 1–es. 2005.

Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Cheng-Hao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, et al. Sequence-augmented se (3)-flow matching for conditional protein generation. *Advances in neural information processing systems*, 37:33007–33036, 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with genie 2. *arXiv preprint arXiv:2405.15489*, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

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

Linus Pauling, Robert B Corey, and Herman R Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.

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

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

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Daniel-Adriano Silva, Shawn Yu, Umut Y Ulge, Jamie B Spangler, Kevin M Jude, Carlos Labão-Almeida, Lestat R Ali, Alfredo Quijano-Rubio, Mikel Ruterbusch, Isabel Leung, et al. De novo design of potent and selective mimics of il-2 and il-15. *Nature*, 565(7738):186–191, 2019.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, volume 2024, pages 15078–15097, 2024.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Eva-Maria Strauch, Steffen M Bernard, David La, Alan J Bohn, Peter S Lee, Caitlin E Anderson, Travis Nieusma, Carly A Holstein, Natalie K Garcia, Kathryn A Hooper, et al. Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nature biotechnology*, 35(7):667–671, 2017.

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

Michel van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Foldseek: fast and accurate protein structure search. *Biorxiv*, pages 2022–02, 2022.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Dongyeop Woo, Marta Skreta, Seonghyun Park, Kirill Neklyudov, and Sungsoo Ahn. Riemannian meanflow. *arXiv preprint arXiv:2602.07744*, 2026.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023a.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023b.

Angxiao Yue, Zichong Wang, and Hongteng Xu. Reqflow: Rectified quaternion flow for efficient and high-quality protein backbone generation. *arXiv preprint arXiv:2502.14637*, 2025.

Olga Zaghen, Floor Eijkelboom, Alison Pouplin, Cong Liu, Max Welling, Jan-Willem van de Meent, and Erik J Bekkers. Riemannian variational flow matching for material and protein design. *arXiv preprint arXiv:2502.12981*, 2025.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. *arXiv preprint arXiv:2510.20771*, 2025.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Yuan Zhang and Celeste Sagui. Secondary structure assignment for conformationally irregular peptides: Comparison between dssp, stride and kaksi. *Journal of Molecular Graphics and Modelling*, 55:72–84, 2015.

Zichen Zhong, Haoliang Sun, Yukun Zhao, Yongshun Gong, and Yilong Yin. Riemannian meanflow for one-step generation on manifolds. *arXiv preprint arXiv:2603.10718*, 2026.

Appendix Contents

A Background: MeanFlow in Euclidean Space

Consider a probability path $\{p_t : t \in [0,1]\}$ with $p_0 = p_{data}$ and $p_1 = p_{prior}$, we assume p_t is generated by the following ODE system:

$$\begin{cases} X_0 \sim p_0 & \text{initial distribution} \\ dX_t = v(t, X_t)dt & \text{ODE} \\ p_t = \text{Law}(X_t) & \text{Marginal distribution in time} \end{cases} \tag{15}$$

Equivalently, we write (v_t, x_t) satisfies the continuity equation:

$$\frac{d}{dt}p_t = -\nabla \cdot (v_t p_t) \tag{16}$$

with the same initial distribution condition.

Geng et al. [2026a] proposed the mean velocity field $u(s,t,x_t)$ defined as

$$u(s,t,x_t) := \frac{1}{t-s}\int_s^t v(\tau,x_\tau)d\tau \in \mathbb{R}^d. \tag{17}$$

Multiplying $(t-s)$ on both sides and differentiating both sides with respect to t yields the *MeanFlow identity*:

$$\begin{cases} \text{L.H.S.} = \frac{d}{dt}(t-s)u(s,t,x_t) = u(s,t,x_t) + (t-s)\frac{d}{dt}u(s,t,x_t) \\ \text{R.H.S.} = v(t,x_t) = x_1 - x_0 \end{cases}$$

We compute R.H.S. from the data and L.H.S. from the model, the training loss is derived:

$$\mathcal{L} = \begin{cases} \mathbb{E}_{t,(x_0,x_1)\sim\pi_0}\left[\left\|u_\theta + \text{sg}((t-s)(\frac{d}{dt}u_t) - v(t,x_t))\right\|^2\right] \text{ Geng et al. [2026a]} \\ \mathbb{E}_{t,(x_0,x_1)\sim\pi_0}\left[\left\|u_\theta + (t-s)\text{sg}(\frac{d}{dt}u_t) - v(t,x_t)\right\|^2\right] \text{ Geng et al. [2026b]} \end{cases} \tag{18}$$

where $v(t,x_t) = x_1 - x_0$

$$\frac{d}{dt}u_\theta(r,t,x_t) = \frac{\partial u_\theta}{\partial x_t}v(t,x_t) + \frac{\partial u_\theta}{\partial t}$$

and sg denotes the stop-gradient operation, which is used to prevent backpropagation through the target.

One-step training objective. MeanFlow learns a parametric field $u_\theta(s,t,x_t)$ via regression, using the standard linear interpolation between a data sample $x_0 \sim p_{\text{data}}$ and a prior sample $x_1 \sim p_{\text{prior}}$:

$$x_t = (1-t)x_0 + tx_1, \qquad v = x_1 - x_0. \tag{19}$$

Using the MeanFlow identity, the target mean flow can be written as

$$u_{\text{tgt}}(s,t,x_t) = v - (t-s)\frac{d}{dt}u_\theta(s,t,x_t), \tag{20}$$

where $\frac{d}{dt}u_\theta$ is the total derivative (implemented as a Jacobian-vector product), and u_{tgt} is treated as a stop-gradient target. The MeanFlow training loss is

$$\mathcal{L}_{\text{MF}}(\theta) = \mathbb{E}_{x_0\sim p_{\text{data}},\ x_1\sim p_{\text{prior}},\ s<t}\left[\left\|u_\theta(s,t,x) - \text{sg}\big(u_{\text{tgt}}(s,t,x_t)\big)\right\|_2^2\right]. \tag{21}$$

B Background: SO(3) Space

B.1 Basic concepts in SO(3) space

The special orthogonal group in three dimensions, denoted as SO(3), is defined as

$$\text{SO}(3) := \{R \in \mathbb{R}^{3\times 3} : RR^\top = R^\top R = I_3, \det(R) = 1\}.$$

The constraint $RR^\top = I_3$ consists of smooth polynomial equations, which implies that SO(3) is a smooth embedded submanifold of $\mathbb{R}^9$.

Equipped with matrix multiplication $\text{SO}(3)^{\times 2} \ni (S_1, S_2) \mapsto S_1S_2 \in \text{SO}(3)$, SO(3) forms a Lie group. The identity element is I_3, and the inverse of any element $R \in \text{SO}(3)$ is given by $R^{-1} = R^\top$.

Tangent space. Let $R \in \mathrm{SO}(3)$ be fixed. Consider a smooth curve $R(t) \in \mathrm{SO}(3)$ such that

$$R(0) = R.$$

Since $R(t)R(t)^\top = I_3$ holds for all t, differentiating at $t = 0$ yields

$$\dot{R}(0)R^\top + R\dot{R}(0)^\top = 0.$$

This implies that $R^\top \dot{R}(0)$ is skew-symmetric. Therefore, the tangent space at R is given by

$$T_R(\mathrm{SO}(3)) = \{R\Omega : \Omega \in \mathfrak{so}(3)\},$$

where the Lie algebra $\mathfrak{so}(3)$ is defined as

$$\mathfrak{so}(3) := \{\Omega \in \mathbb{R}^{3\times 3} : \Omega^\top = -\Omega\}.$$

This representation provides a global linear parameterization of tangent vectors on $\mathrm{SO}(3)$. We also use the standard hat/vee identification between $\mathbb{R}^3$ and $\mathfrak{so}(3)$: for $\omega = (\omega_1, \omega_2, \omega_3) \in \mathbb{R}^3$,

$$\omega^\wedge := \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3), \qquad (\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3 \text{ is its inverse.}$$

Lie algebra and Lie bracket. The vector space $\mathfrak{so}(3)$ is the Lie algebra associated with the Lie group $\mathrm{SO}(3)$. The Lie bracket on $\mathfrak{so}(3)$ is defined as the matrix commutator

$$[\Omega_1, \Omega_2] = \Omega_1\Omega_2 - \Omega_2\Omega_1, \quad \Omega_1, \Omega_2 \in \mathfrak{so}(3).$$

It is straightforward to verify that this operation preserves skew-symmetry, hence $\mathfrak{so}(3)$ is closed under the Lie bracket.

B.2 Curves and flows in SO(3)

In Euclidean space, a curve connecting x_0 and x_1 can be defined by the ODE

$$\begin{cases} X_0 = x_0, \\ \dot{X}_t = v(t, X_t). \end{cases} \tag{22}$$

and thus $X_1 = \int_0^1 v(t, x_t)dt$. We now extend this construction to the Lie group $\mathrm{SO}(3)$.

ODE-defined curves on $\mathrm{SO}(3)$**.** Let $R_t : [0, 1] \to \mathrm{SO}(3)$ be a time-dependent curve. A natural intrinsic definition of its evolution is given by

$$\begin{cases} R_0 = r_0 \\ \dot{R}_t = R_t\Omega_t, \quad \Omega_t \in \mathfrak{so}(3). \end{cases}$$

This ODE guarantees that R_t remains in $\mathrm{SO}(3)$ for all t, since

$$\frac{d}{dt}(R_tR_t^\top) = R_t\Omega_tR_t^\top + R_t\Omega_t^\top R_t^\top = 0.$$

Given an initial condition $R_0 \in \mathrm{SO}(3)$, the above equation defines a smooth curve on the manifold.

Exponential map and geodesics. When the angular velocity is constant, namely $\Omega_t = \Omega$, the ODE admits a closed-form solution

$$R_t = R_0 \exp(t\Omega),$$

where exp denotes the matrix exponential. Under the canonical bi-invariant Riemannian metric on SO(3), curves of this form are geodesics.

Given two points $R_0, R_1 \in \mathrm{SO}(3)$, the geodesic connecting them is given by

$$R_t = R_0 \exp\big(t\mathrm{Log}(R_0^\top R_1)\big), \quad t \in [0,1],$$

where Log denotes the matrix logarithm mapping SO(3) to $\mathfrak{so}(3)$. This curve minimizes the path length among all smooth curves on SO(3) connecting R_0 and R_1.

Inner product and geodesic distance in SO(3)**.** Under the canonical bi-invariant Riemannian metric, we identify $T_R\mathrm{SO}(3) = \{R\Omega : \Omega \in \mathfrak{so}(3)\}$ and define the inner product

$$\langle R\Omega_1, R\Omega_2\rangle_R = \tfrac{1}{2}\operatorname{tr}(\Omega_1^\top \Omega_2) = \tfrac{1}{2}\operatorname{tr}\big((R^\top \dot{R}_1)^\top (R^\top \dot{R}_2)\big),$$

which is invariant under left and right multiplication. The induced geodesic distance between $R_0, R_1 \in \mathrm{SO}(3)$ is

$$\mathrm{dist}(R_0, R_1) = \frac{1}{\sqrt{2}} \big\|\mathrm{Log}(R_0^\top R_1)\big\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Integration in SO(3). In Euclidean space, $x_1 = \int_0^1 v(t, x_t)dt + x_0$.

Due to the non-commutative group structure, the endpoint in SO(3) cannot be written as a simple integral. Instead, the solution at $t = 1$ admits the group-valued representation

$$R(1) = R(0)\mathcal{T}\exp\Big(\int_0^1 \Omega_t dt\Big),$$

where $\mathcal{T}$ denotes the time-ordering operator:

$$\mathcal{T}\exp\Big(\int_0^1 A(t)dt\Big) = I_3 + \sum_{k=1}^{\infty} \int_{0\le t_1 \le \cdots \le t_k \le 1} A(t_1)\cdots A(t_k)dt_1\cdots dt_k,$$

where the time-ordering operator enforces the chronological ordering of the matrix products.

C Background: SE(3) Space

C.1 Basic concepts in the SE(3) space

The special Euclidean group in three dimensions is defined as

$$\mathrm{SE}(3) = \{(R, x) : R \in \mathrm{SO}(3),\ x \in \mathbb{R}^3\},$$

representing orientation-preserving rigid motions in $\mathbb{R}^3$. Algebraically, SE(3) is the semidirect product

$$\mathrm{SE}(3) = \mathrm{SO}(3) \ltimes (\mathbb{R}^3, +),$$

where rotations act on translations. The group operation is given by

$$(R_1, x_1)(R_2, x_2) = (R_1 R_2,\ x_1 + R_1 x_2), \tag{23}$$

and the inverse by

$$(R, x)^{-1} = (R^\top,\ -R^\top x).$$

As a smooth manifold, SE(3) is six-dimensional and diffeomorphic to $\mathrm{SO}(3) \times \mathbb{R}^3$.

Lie algebra and tangent space. The Lie algebra $\mathfrak{se}(3)$ consists of pairs (Ω, v) with $\Omega \in \mathfrak{so}(3)$ and $v \in \mathbb{R}^3$, i.e.,

$$\mathfrak{se}(3) = \{(\Omega, v) : \Omega^\top = -\Omega,\ v \in \mathbb{R}^3\}.$$

The Lie bracket is given by

$$[(\Omega_1, v_1), (\Omega_2, v_2)] = ([\Omega_1, \Omega_2],\ \Omega_1 v_2 - \Omega_2 v_1).$$

For any $(R, x) \in \mathrm{SE}(3)$, the tangent space is obtained by left translation:

$$T_{(R,x)}\mathrm{SE}(3) = \{(R\Omega,\ Rv) : \Omega \in \mathfrak{so}(3),\ v \in \mathbb{R}^3\}.$$

C.2 Curves and flows in SE(3)

A time-dependent rigid motion is represented by a curve $(R_t, s_t) \in \mathrm{SE}(3)$. A natural way to define its intrinsic evolution is through the left-trivialized velocity:

$$\begin{cases} \dot{R}_t = R_t \Omega_t, \\ \dot{x}_t = R_t v_t, \end{cases} \qquad (\Omega_t, v_t) \in \mathfrak{se}(3),$$

where Ω_t governs angular motion and v_t determines translational motion in the body frame. Note that the translational velocity is expressed in the body frame and is therefore rotated by R_t: in the full semidirect-product geometry the two components are coupled.

C.3 The decoupled product geometry used in this work

Following the setting in Section 3 of Bose et al. [2024], we instead work with the product manifold

$$\mathrm{SO}(3) \times \mathbb{R}^3$$

equipped with a product Riemannian metric, under which the composition simplifies to

$$(R_1, s_1)(R_2, s_2) = (R_1 R_2,\ s_1 + s_2). \tag{24}$$

Comparing with (23), this replaces $s_1 + R_1 s_2$ by $s_1 + s_2$, i.e. it drops the action of R_1 on s_2, so that the rotational and translational coordinates evolve independently. The left-trivialized flow correspondingly reduces to $\dot{R}_t = R_t \Omega_t$ and $\dot{s}_t = v_t$.

Remark 2 (Scope of this section). This section is included as background only. Our method does *not* use the semidirect-product structure (23) or any property specific to it. Following FoldFlow Bose et al. [2024] and ReQFlow Yue et al. [2025], we work throughout with the decoupled product geometry (24), in which rotation and translation are independent. Accordingly, all constructions in this paper—the interpolation path, the MeanFlow identity, the α-Flow target, the semigroup loss, and the training objectives—are derived *separately* on $\mathrm{SO}(3)$ and on $\mathbb{R}^3$, and the $\mathrm{SE}(3)$ ($\mathrm{SE}(3)^N$) losses are simply the sum of the two branches. The translational velocity is likewise taken in the ambient frame rather than the body frame. Whenever we write $\mathrm{SE}(3) \cong \mathrm{SO}(3) \times \mathbb{R}^3$ in the main text, it refers to this decoupled convention rather than to a group isomorphism.

D Background: SE(3) Flow Matching

Our method builds on flow matching, which we briefly review here—first on a general Riemannian manifold, and then in the decoupled SE(3) geometry used throughout the paper.

D.1 Riemannian flow matching

Flow matching [Lipman et al., 2022, Tong et al., 2023] learns a time-dependent velocity field whose flow transports a prior density p_1 to the data density p_0. On a Riemannian manifold $\mathcal{M}$, one fixes a conditional coupling $(z_0, z_1) \sim q$ and connects the endpoints by the minimizing geodesic

$$z_t = \exp_{z_0}\big(t \, \log_{z_0}(z_1)\big), \qquad t \in [0,1],$$

whose time derivative is the conditional (target) velocity $\dot{z}_t \in \mathcal{T}_{z_t}\mathcal{M}$. Regressing a model field $u_\theta(t, z_t)$ onto this target,

$$\mathbb{E}_{t,\, q(z_0,z_1)}\big[\, \|u_\theta(t, z_t) - \dot{z}_t\|_g^2 \,\big],$$

recovers at its minimizer the marginal velocity that generates the interpolating path p_t. Sampling then integrates the learned field, e.g. backward from a prior draw z_1 to a data sample z_0.

D.2 SE(3) flow matching

Following Bose et al. [2024], we adopt the decoupled product geometry $\mathrm{SE}(3) \cong \mathrm{SO}(3) \times \mathbb{R}^3$, so that the geodesic and its velocity split into independent rotational and translational parts. For a data frame (R_0, x_0) and a prior frame (R_1, x_1), the conditional path is the SO(3) geodesic paired with the Euclidean straight line,

$$R_t = R_0 \exp\big(t \, \log(R_0^\top R_1)\big), \qquad x_t = (1-t)\, x_0 + t\, x_1.$$

Differentiating gives the closed-form conditional velocities, which are constant along each path:

$$\omega_t := \log(R_0^\top R_1)^\vee \in \mathbb{R}^3, \qquad v_t := x_1 - x_0 \in \mathbb{R}^3,$$

so that $\dot{R}_t = R_t\, \omega_t^\wedge$ and $\dot{x}_t = v_t$. The model predicts the body angular and translational velocities $A_\theta(t, R_t, x_t)$ and $v_\theta(t, x_t)$—equivalently the instantaneous ($s{=}t$) evaluation of the two-time head, $A_\theta(t, R_t, x_t) = A_\theta^{\mathrm{avg}}(t, t, R_t, x_t)$—and the SE(3) flow-matching objective regresses them onto these targets,

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\, q(z_0,z_1)} \sum_{i=1}^{N} \Big[\big\| A_{\theta,i}(t, R_{t,i}, x_{t,i}) - \omega_{t,i} \big\|_2^2 + \big\| v_{\theta,i}(t, x_{t,i}) - v_{t,i} \big\|_2^2 \Big], \tag{25}$$

summed over the N residues. Equation (25) is the data-anchored objective to which our MeanFlow and α-Flow targets reduce in the appropriate limit—for instance at $\alpha = 1$, where $A_{\mathrm{tgt}}^{\mathrm{avg}} = \omega_t$—and it supplies the boundary condition that keeps the consistency objective from collapsing.

D.3 Mini-batch optimal-transport coupling

The flow-matching objective (25) is defined for any coupling $q(z_0, z_1)$ of the data and prior marginals. The simplest choice is the independent coupling $q = p_0 \otimes p_1$, under which trajectories from different pairs cross and the marginal velocity field is far from constant along each path. Following Tong et al. [2023], Pooladian et al. [2023] and FoldFlow Bose et al. [2024], we instead re-pair each batch

by discrete optimal transport. Given M data frames $\{z_0^k\}$ and M prior frames $\{z_1^k\}$ with uniform weights, the Optimal transport problem Villani et al. [2009], Flamary et al. [2021]

$$\min_{\Pi \in \mathcal{U}_M} \sum_{k,l} \Pi_{kl}\, c(z_0^k, z_1^l), \qquad \mathcal{U}_M = \{\Pi \geq 0 : \Pi \mathbf{1} = \Pi^\top \mathbf{1} = \tfrac{1}{M}\mathbf{1}\},$$

attains its optimum at a vertex of $\mathcal{U}_M$, i.e. (Birkhoff) at a permutation $\pi^\star \in \mathcal{S}_M$, so the plan reduces to a linear assignment solvable exactly in $O(M^3)$ by the Hungarian algorithm. On the decoupled product geometry (24) we take the squared $\mathrm{SE}(3)^N$ geodesic cost,

$$c\big((R_0, x_0), (R_1, x_1)\big) = \sum_{i=1}^{N} \Big[\lambda_R\, d^2_{\mathrm{SO}(3)}(R_0^i, R_1^i) + \lambda_x\, \|x_0^i - x_1^i\|_2^2\Big], \quad d_{\mathrm{SO}(3)}(R, R') = \tfrac{1}{\sqrt{2}}\|\log(R^\top R')\|_F,$$

with $(\lambda_R, \lambda_x) = (0.5, 0.5)$. Re-pairing shortens the average transport distance and straightens the induced probability path; the resulting velocity field is closer to constant along each trajectory, which is exactly the regime in which few-step average-velocity sampling is accurate. This is also why the controlled benchmark of Appendix H disables OT (Section H.2): with OT the instantaneous- and average-velocity parameterisations nearly coincide, and the comparison would no longer be informative.

E Related Work

E.1 Diffusion and early flow-matching baselines

The first wave of $\mathrm{SE}(3)^N$ backbone generators are diffusion models over residue frames. FrameDiff Yim et al. [2023b] formulates SE(3)-invariant score-based diffusion on multiple frames and generates designable monomers up to 500 residues without a pretrained structure-prediction network (17.4M parameters). Genie Lin et al. [2024] instead diffuses oriented C_α residue clouds with SE(3)-equivariant triangle updates; it is parameter-light (4.1M) and attains high diversity and novelty, but its designability is comparatively low and degrades sharply as the number of sampling steps is reduced. FrameFlow Yim et al. [2023a] keeps the frame representation but replaces diffusion with SE(3) flow matching, reporting roughly $2\times$ higher designability than FrameDiff at about $5\times$ fewer sampling steps, and a $\sim 23\times$ sampling speed-up over Genie at markedly higher designability. Because FrameFlow dominates both earlier models on the designability–efficiency axis that is our focus, we take it as the representative of this pre-2024 generation in the main-text comparison (Table 1) and do not separately tabulate FrameDiff or Genie. The flow-matching methods most closely related to ours—FoldFlow, ReQFlow, and Riemannian MeanFlow—are discussed next.

E.2 FoldFlow Bose et al. [2024]

Bose et al. [2024] introduces SE(3) stochastic flow matching for protein backbone generation. Their training objective is a flow-matching regression that fits a time-dependent vector field (v_θ^R, v_θ^x) to the drift of a chosen probability path (an SE(3) bridge) between data (R_0, x_0) and noise (R_1, x_1):

$$\begin{aligned}
\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,(R_0,x_0),(R_1,x_1)} \Big[& \|v_\theta^R(R_t, t) - v^R(R_t, t \mid R_0, R_1)\|^2_{\mathrm{SO}(3)} \\
& + \|v_\theta^x(x_t, t) - v_t^x(x_t, t | x_0, x_1)\|^2 \Big] \\
\text{where } v_t^R(R_t | R_0, R_1) := \dot{R}_t \mid_{R_0,R_1} & := R_t \Omega_t, \Omega_t = \log(R_0^\top R_1), R_t = R_0 e^{t\Omega_t} \\
v_t^x(x_t | x_0, x_1) & = x_1 - x_0, x_t = (1-t)x_0 + t x_1.
\end{aligned}$$

In this work, the authors introduce three variants:

- In the base setting, q is the independent coupling between $(p_0 = p_{data}, p_1 = p_{prior})$.
- In addition, the authors introduce the (mini-batch) optimal transport coupling, obtained by batch-size sample $p_0^B \sim$ i.i.d. $p_0, p_1^B \sim$ i.i.d. p_1 and the method is denoted as FoldFlow-OT.
- Furthermore, they introduce perturbation in the rotation interpretation R_t by the IGSO-distribution:
$$\tilde{R}_t \mid_{R_0,R_1} \sim \mathcal{IG}_{SO(3)}(R_t, \gamma^2(t)t(1-t)).$$
where $\gamma^2(t) > 0$ is a predefined function.

Adaptation to protein backbone space: $\mathrm{SE}(3)^N$ As discussed in the main text, a protein backbone with N residues can be represented as a product space $\mathrm{SE}(3)^N$, i.e., $x = (g_1, \ldots, g_N)$ with $g_i = (R_i, x_i) \in \mathrm{SE}(3)$. The SE(3) flow-matching loss then extends by summing (or averaging) the per-residue SE(3) losses, which is equivalent to *concatenating* all residue-wise tangent vectors into a single $6N$-dimensional stacked vector:

$$\begin{aligned}\mathcal{L}_{\mathrm{FM}}^{\mathrm{SE}(3)^N}(\theta) =& \mathbb{E}\frac{1}{N}\Big[\sum_{i=1}^{N} \|v_{\theta,i}^R(R_{t,i}, t) - v_{t,i}^R(R_{t,i} \mid R_{0,i}, R_{1,i})\|_{\mathrm{SO}(3)}^2 \\ &+ \sum_{i=1}^{N} \|v_{\theta,i}^x(x_{t,i}, t) - v_{t,i}^x(x_{t,i} \mid x_{0,i}, x_{1,i})\|^2\Big]\end{aligned}$$

In addition, the auxiliary loss (see section G.1) is included in the final training loss.

E.3 FoldFlow-2 Huguet et al. [2024]

Huguet et al. [2024] extends FoldFlow to a sequence-conditioned generative model. The underlying generative task remains $\mathrm{SE}(3)^N$ flow matching as in FoldFlow, and the loss structure is identical. The key novelty is that the vector field is now conditioned on a (possibly masked) amino acid sequence $\bar{a}$:

$$\begin{aligned}\mathcal{L}_{\mathrm{FM}}^{\mathrm{FF2}}(\theta) = \mathbb{E}_{t,\,\pi(x_0,x_1),\,\bar{a}}\frac{1}{N}\Big[&\sum_{i=1}^{N} \|v_{\theta,i}^R(R_{t,i}, t|\bar{a}) - v_{t,i}^R(R_{t,i} \mid R_{0,i}, R_{1,i}, \bar{a})\|_{\mathrm{SO}(3)}^2 \\ &+ \sum_{i=1}^{N} \|v_{\theta,i}^x(x_{t,i}, t|\bar{a}) - v_{t,i}^x(x_{t,i} \mid x_{0,i}, x_{1,i}, \bar{a})\|^2\Big],\end{aligned} \tag{26}$$

where $\pi(x_0, x_1)$ is the minibatch OT coupling (as in FoldFlow-OT), and $\bar{a} = a \odot m$ is the masked sequence with mask $m \sim \mathrm{Bern}(0.5)$ applied uniformly across all residues. This stochastic masking enables a single model to handle both conditional and unconditional generation:

- $\bar{a} = [\varnothing]^N$ (fully masked, probability 0.5): unconditional backbone generation, equivalent to FoldFlow-OT.
- $\bar{a} = a$ (unmasked, probability 0.5): sequence-conditioned generation, i.e. protein folding.
- $\bar{a} = a \odot m$ (partially masked): structure in-painting and motif scaffolding.

Reinforced Fine-Tuning (ReFT). FoldFlow-2 further introduces a fine-tuning objective to align generations towards an auxiliary reward r_{aux}. Given a preferential dataset $\mathcal{D}_{\text{pref}}$ filtered by r_{aux}, the ReFT objective is:

$$\max_{p_\theta} \mathcal{L}_{\text{ReFT}}(\theta) = \mathbb{E}_{(x,a)\sim\mathcal{D}_{\text{pref}}} \left[r_{\text{aux}}(x) \log p_\theta(x \mid a) \right]. \tag{27}$$

This is applied, for example, to improve secondary structure diversity by upweighting samples rich in β-sheets and coils.

E.4 ReQFlow Yue et al. [2025]

Yue et al. [2025] propose a quaternion-based flow model for protein backbone generation. Similar to FrameFlow and FoldFlow, the protein backbone is represented as a collection of residue-wise rigid frames in $\text{SE}(3)^N$. However, instead of representing rotations by matrices in SO(3), ReQFlow parameterizes each residue frame as

$$g_i = (x_i, q_i), \qquad x_i \in \mathbb{R}^3, \;\; q_i \in \mathbb{S}^3,$$

where x_i is the local translation and q_i is a unit quaternion representing the 3D rotation.

In particular, given a rotation matrix $R \in \text{SO}(3)$, let $\omega = \phi u = (\log(R))^\vee \in \mathbb{R}^3$ be its axis-angle vector, where $\phi = \|\omega\|, u = \frac{\omega}{\|\omega\|}$ the corresponding unit quaternion is given by

$$q = \exp(\frac{1}{2}\omega) = [\cos\frac{\phi}{2}, \sin\frac{\phi}{2}u^\top]^\top \in \mathbb{S}^3.$$

Conversely, given a unit quaternion $q = (w, x, y, z)^\top \in \mathbb{S}^3$, the corresponding rotation matrix is

$$R = \exp((2\log(q))^\wedge).$$

Moreover, given 3D rotations R_1, R_2 with corresponding quaternion representations $q_1 := (s_1, u_1), q_2 := (s_2, u_2)$, their group action (matrix multiplication) can be expressed by quaternion multiplication:

$$q_1 \otimes q_2 = \begin{bmatrix} s_1 s_2 - u_1^\top u_2 \\ s_1 u_2 + s_2 u_1 + u_1 \otimes u_2 \end{bmatrix}.$$

This quaternion algebra leads to a more numerically stable and efficient treatment of rotations, especially when the rotation angle is very small or close to π.

Quaternion Flow Matching (QFlow). In ReQFlow, the authors denote by $\mathcal{T}_0$ the prior distribution

$$\mathcal{N}(0, I_3) \times \mathcal{IG}_{SO(3)},$$

where, with a slight abuse of notation, $\mathcal{IG}_{SO(3)}$ denotes the isotropic Gaussian distribution in rotation space under the quaternion representation. The target distribution $\mathcal{T}_1$ corresponds to the real protein data distribution.

ReQFlow parameterizes the model as

$$T_{\theta,1}(x_t, q_t) = (x_{\theta,1}, q_{\theta,1}),$$

where $x_{\theta,1}$ and $q_{\theta,1}$ denote the predicted translation and rotation at terminal time $t = 1$, conditioned on the current state (x_t, q_t).

At time $t \in [0,1]$, the translation path follows the standard linear interpolation

$$x_t = (1-t)x_0 + tx_1,$$

with translation velocity

$$v_t^x = x_1 - x_0 = \frac{x_1 - x_t}{1-t}, \qquad v_{\theta,t}^x = \frac{x_{\theta,1} - x_t}{1-t}.$$

For the rotational component, ReQFlow adopts quaternion geodesic interpolation:

$$q_t = q_0 \otimes \exp\big(t \log(q_0^{-1} \otimes q_1)\big).$$

The corresponding angular velocity and its model-based estimate are

$$v_t^q = 2\log(q_0^{-1} \otimes q_1) = \frac{2\log(q_t^{-1} \otimes q_1)}{1-t}, v_{\theta,t}^q = \frac{2\log(q_t^{-1} \otimes q_{\theta,1})}{1-t}.$$

The inference dynamics are then approximated by

$$\begin{cases} x_{t+\Delta t} = x_t + \Delta t\, v_{\theta,t}^x, \\ q_{t+\Delta t} = q_t \otimes \exp\left(\frac{1}{2}\Delta t\, v_{\theta,t}^q\right). \end{cases}$$

Accordingly, the flow-matching loss is defined as

$$\mathcal{L}_{\text{QFlow}} = \mathbb{E}_{t,(x_0,q_0),(x_1,q_1)}\big[\|v_t^x - v_{\theta,t}^x\|^2\big] + \mathbb{E}_{t,(x_0,q_0),(x_1,q_1)}\big[\|v_t^q - v_{\theta,t}^q\|^2\big].$$

In addition, the auxiliary loss (see section G.1) is incorporated into the training process.

E.5 Riemannian Gaussian Variational Flow Matching (RG-VFM) Zaghen et al. [2025]

Zaghen et al. [2025] approach manifold generation from the *variational* rather than the velocity-matching side. Building on Variational Flow Matching, they replace the instantaneous-velocity regression used by CFM/RFM—and, in our setting, by FrameFlow, FoldFlow, and ReQFlow—with an *endpoint* objective: a network predicts a terminal mean $\mu_\theta(x_t) \in \mathcal{M}$, and the posterior $q_\theta(x_1 \mid x_t)$ is modeled as a Riemannian Gaussian

$$\mathcal{N}_{\text{Riem}}\big(x_1 \mid \mu_\theta(x_t), \sigma\big) \;\propto\; \exp\Big(-\frac{\text{dist}_g\big(x_1, \mu_\theta(x_t)\big)^2}{2\sigma^2}\Big).$$

On a homogeneous manifold with closed-form geodesics, the normalizing constant is independent of μ, and the objective collapses to a squared geodesic (endpoint) distance:

$$\mathcal{L}_{\text{RG-VFM}}(\theta) = \mathbb{E}_{t,x_1,x_t}\big[\|\log_{x_1}(\mu_\theta(x_t))\|_g^2\big] = \mathbb{E}_{t,x_1,x_t}\big[\text{dist}_g\big(x_1, \mu_\theta(x_t)\big)^2\big],$$

which recovers the Euclidean VFM/MSE loss $\|\mu_\theta(x_t) - x_1\|^2$ when $\mathcal{M} = \mathbb{R}^d$. Unlike vanilla RFM, whose vector field lives in $T_{x_t}\mathcal{M}$ and requires $\text{supp}(p_0)$ to lie on $\mathcal{M}$, the variational objective compares endpoints in the single tangent space $T_{x_1}\mathcal{M}$ and only needs the local geometry around p_1.

Adaptation to the backbone setting ("variational ReQFlow"). Instantiating this objective in ReQFlow's frame representation $g_i = (x_i, q_i) \in \mathbb{R}^3 \times \mathbb{S}^3$ yields an endpoint-matching counterpart of QFlow. Rather than regressing the translation/quaternion velocities as in $\mathcal{L}_{\text{QFlow}}$,

the model predicts the terminal frame $(x_{\theta,1}, q_{\theta,1})$ and minimizes the endpoint geodesic distance directly:

$$\mathcal{L}_{\text{v-ReQFlow}}(\theta) = \mathbb{E}_t\Big[\sum_{i=1}^{N}\|x_{1,i}-x_{\theta,1,i}\|^2+\lambda\sum_{i=1}^{N}\text{dist}_{\text{SO(3)}}\big(q_{1,i}, q_{\theta,1,i}\big)^2\Big], \qquad \text{dist}_{\text{SO(3)}}(q_1, q_{\theta,1}) = 2\big\|\log(q_{\theta,1}^{-1}\otimes q_1)\big\|,$$

i.e. the flow-matching term is replaced by a squared endpoint distance on the product space $\text{SE(3)}^N = (\text{SO(3)} \times \mathbb{R}^3)^N$.

Relation to our work. At the loss level, RG-VFM can be read as a variant of ReQFlow in which the velocity-matching FM objective is replaced by an endpoint geodesic distance: the network still predicts a terminal mean $\mu_\theta(x_t)$ (an x_1-prediction parameterization), so sampling remains an ODE integration and inherits the same limited few-step generation behavior as standard flow matching. Our method is instead a MeanFlow-type model that directly learns the average-velocity flow map $\Phi_{s,t}$ via the time-ordered exponential (Eq 30), which is what enables few-step generation. The two are nonetheless complementary rather than opposed: in the second phase of our training (Section M), we combine this endpoint geodesic loss with the MeanFlow objective, using the former to stabilize the JVP (total-derivative) term in the MeanFlow loss. Finally, RG-VFM is validated only on a synthetic spherical dataset and releases no SE(3)^N protein checkpoint, so we do not include it as a standalone baseline (see below).

E.6 Riemannian MeanFlow (RMF) Woo et al. [2026]

Concurrent with our work, Woo et al. [2026] generalize MeanFlow from Euclidean space to Riemannian manifolds, with applications to scientific generative modeling on manifolds such as the simplex and SE(3)^N. Instead of learning an instantaneous velocity field and numerically integrating it during sampling, RMF directly learns the flow map

$$\Phi_{s,t} : \mathcal{M} \to \mathcal{M},$$

which transports a point $x_s \sim p_s$ at time s to its corresponding point $x_t \sim p_t$ at time t along the same integral curve.

The key geometric quantity in RMF is the *average velocity*, defined by

$$u_{s,t}(x_s) = \begin{cases} \dfrac{1}{t-s}\log_{x_s}(x_t), & t \neq s, \\ v_s(x_s), & t = s, \end{cases} \tag{28}$$

where $\log_{x_s}(x_t) \in T_{x_s}\mathcal{M}$ denotes the Riemannian logarithmic map. Geometrically, $u_{s,t}(x_s)$ is the constant tangent velocity that transports x_s to x_t along a geodesic over time interval $t-s$. By differentiating both sides with respect to t, we obtain:

$$u_{s,t}(x_s) = d(\log_{x_s})_{x_t}[v_t] - (t-s)\partial_t u_{s,t}(x_s). \tag{29}$$

and it induces the training loss:

$$\mathcal{L}_{\text{RMF}}(\theta) = \mathbb{E}_{x_s,s,t}[\|u^\theta_{s,t}(x_s) - \text{sg}(\hat{u}_{tgt})\|^2],$$

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

They further include a cycle-consistency regularizer

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

Training objective in the protein experiments. In the $\mathrm{SE}(3)^N$ protein setting, RMF is trained with a flow-matching loss together with a semigroup (flow-map composition) consistency loss. The semigroup term enforces $\Phi_{r,t} = \Phi_{s,t} \circ \Phi_{r,s}$, but measures the residual as an *endpoint geodesic distance*—the SE(3) distance between the one-step map $\Phi_{r,t}(x_r)$ and the composed two-step map $\Phi_{s,t}(\Phi_{r,s}(x_r))$—rather than regressing log-displacements in the Lie algebra, as in our finite semigroup loss (Appendix K). The two agree at the minimizer, but differ in conditioning: the endpoint-distance form couples the rotation and translation branches through a single SE(3) metric, whereas our log-displacement form keeps them separated and, on the rotation branch, makes the BCH/right-Jacobian structure explicit. We compare both average-velocity formulations on a controlled $\mathrm{SO}(3)^2$ benchmark in Appendix H.

E.7 Riemannian MeanFlow via Parallel Transport (RMF-PT)

Also concurrent with our work, Zhong et al. [2026] extend MeanFlow to general Riemannian manifolds by defining the average velocity as a parallel-transport integral: the instantaneous velocities along the trajectory are transported to a common tangent space before being averaged (Eq. 5 therein). To distinguish it from the Riemannian MeanFlow of Woo et al. [2026] discussed above, we refer to this method as **RMF-PT**. Our approach differs in three respects.

(i) Definition. RMF-PT defines the mean velocity through a path-dependent parallel-transport integral, which requires the full trajectory $\{x_\tau\}_{\tau\in[s,t]}$. We instead define it via the time-ordered exponential (30), which depends only on the endpoints R_s and R_t and reduces to the logarithmic map $\Omega^{\mathrm{avg}} = \frac{1}{t-s}\log(R_s^\top R_t)$. The two definitions coincide when the velocity field is constant along the trajectory (the geodesic case); for general fields they differ by BCH correction terms arising from the non-commutativity of SO(3).

(ii) Identity and loss. Differentiating the two definitions yields structurally different identities. That of RMF-PT involves the covariant derivative $\nabla_{\dot\gamma} u$ with Christoffel corrections (Appendix A of Zhong et al. 2026), which their implementation drops via a log-map approximation. Ours involves the *exact* right Jacobian J of SO(3) (Proposition 3), which admits a closed form and requires no approximation.

(iii) Application. RMF-PT targets general manifolds and is evaluated on synthetic data (spheres, tori, and SO(3) rotations). Our method is built for $\mathrm{SE}(3)^N$ protein backbone generation: we exploit the decoupled product geometry $\mathrm{SO}(3)\times\mathbb{R}^3$ (Appendix C) to obtain separate, simulation-free objectives for rotations and translations, and validate them on *de novo* backbone design.

E.8 Baseline selection

Since FoldFlow/FoldFlow2 and v-QFlow/v-ReQFlow do not have public checkpoints for the SCOPe dataset, these methods are not included in the experiments. In addition, FoldFlow and QFlow are essentially the same in nature—both are flow-matching models—and differ only in the mathematical representation of rotations (rotation matrices vs. quaternions) and in the backbone architecture. Since QFlow outperforms FoldFlow on the PDB dataset, we consider QFlow already sufficiently representative of this family. v-QFlow, in turn, can be viewed as a variant of QFlow whose performance largely matches QFlow's—strong at a moderate number of sampling steps (e.g. 100/500 steps) but degrading as the step count is reduced. We therefore select QFlow alone as the representative method.

F MeanFlow on SE(3): Derivation, Theoretical Properties, and N-component Implementation

F.1 Model and loss function

We first introduce some fundamental results in Riemannian/SO3 manifold:

Remark 3.

- **The hat map does not contribute to the derivative.** Let $A : [0,1] \to \mathbb{R}^3$ be a smooth curve and define $\Omega(t) = A(t)^\wedge$, where $(\cdot)^\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$ is the (linear) hat operator, and $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is its inverse. Since $(\cdot)^\wedge$ is linear and independent of t, the chain rule gives
$$\frac{d}{dt}\Omega(t) = \Big(\frac{dA}{dt}\Big)^\wedge \in \mathbb{R}^{3\times 3}.$$

- **Frechet derivative of the matrix exponential.** Let $R(t) : [0,1] \to \mathbb{R}^{3\times 3}$ be a smooth matrix curve and define $F(t) = \exp(R(t))$. By the chain rule on Banach spaces,
$$\dot{F}(t) = d(\exp)_{R(t)}[\dot{R}(t)],$$
where $d(\exp)_R : \mathbb{R}^{3\times 3} \to \mathbb{R}^{3\times 3}$ denotes the *Fréchet derivative of the matrix exponential at* R in the direction $H = \dot{R}(t)$. The derivative admits the integral (Wilcox) representation:
$$d(\exp)_R(H) = \int_0^1 e^{(1-\alpha)R} H e^{\alpha R} d\alpha.$$
Equivalently, the derivative of the map $R \mapsto \exp(R)$ is the linear operator $d(\exp)_R(\cdot)$ defined for all $H \in \mathbb{R}^{3\times 3}$.

- **Derivative with respect to a point on a manifold.** Consider a smooth curve $t \mapsto R(t)$ on the manifold SO(3) and a smooth function $f : \mathrm{SO}(3) \to \mathbb{R}$. Let $F(t) = f(R(t))$. By the chain rule, we have
$$\frac{d}{dt}F(t) = df_{R(t)}[\dot{R}(t)] = \langle \nabla^{\mathcal{M}}_R f(R(t))\dot{R}(t)\rangle,$$
where df_R denotes the Frechet differential of f at R, and $\nabla^{\mathcal{M}}_R f(R)$ denotes the Riemannian gradient at R. Equivalently, the differential df_R is a linear operator defined on the tangent space $\mathcal{T}_R\mathrm{SO}(3)$ for every $R \in \mathrm{SO}(3)$.

- **On SO(3), the Riemannian gradient coincides with the Euclidean gradient.**
On the Lie group SO(3), we use the canonical inner product $\langle\cdot,\cdot\rangle_R : \mathcal{T}_R(\mathrm{SO}(3))^{\otimes 2} \to \mathbb{R}$ defined by
$$\langle A, B\rangle = \frac{1}{2}\mathrm{Tr}(A^\top B).$$
Then, for any smooth function $f : \mathrm{SO}(3) \to \mathbb{R}$ and its smooth extension $\bar{f} : \mathbb{R}^{3\times 3} \to \mathbb{R}$, we have
$$\nabla^{\mathcal{M}}_R f = \nabla_R \bar{f}, \qquad \forall R \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}.$$

With these preliminaries in place, we now define the average angular velocity on SO(3) and derive the corresponding MeanFlow identity.

We define the **average velocity** as

$$\exp(\underbrace{(t-s)\Omega^{\mathrm{avg}}(s,t,R_t,x_t)}_{\Omega^{s\to t}}) := \mathcal{T}\exp\Big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\, d\tau\Big) \tag{30}$$

Remark 4. SO(3) is not commutative. Thus, in general, $\mathcal{T}\exp\big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau\big) \neq \exp\big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau\big)$ unless Ω_τ is constant. Therefore, we cannot directly define Ω^{avg} via $\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau$.

Note that since each $\Omega(\tau, R_\tau, x_\tau) \in \mathfrak{so}(3)$, we have $\Omega^{\text{avg}}(s, t, R_t, x_t) \in \mathfrak{so}(3)$.

In the extreme case, given $s = 0, t = 1$, the above mean velocity recovers R_0 from R_1:

$$\begin{cases} R_1 = R_0 \exp(\Omega^{\text{avg}}(0, 1, R_1, x_1)) & \text{Forward} \\ R_0 = R_1 \exp(-\Omega^{\text{avg}}(0, 1, R_1, x_1)) & \text{Backward} \end{cases}$$

We apply this convention to the ground truth as well:

$$\Omega^{\text{avg}}(s, t, R_t, x_t) = A(s, t, R_t, x_t)^\wedge.$$

Remark 5. With this model, it is clearer to define the mean velocity via (30), since $\int_s^t \Omega(\tau, R_\tau)\,d\tau$ is identified with an element of $\mathbb{R}^3$, and the above definition is a natural average in Euclidean space.

With this definition, we can now derive a differential identity relating the average angular velocity to the instantaneous velocity, which will form the basis of our training objective.

Proposition 3 (Derivative of the averaged angular velocity)**.** *Under the assumption s, t are independent and using the above notations, set $A^{s\to t} = (t-s)A^{avg}(s, t, R_t, x_t)$. Taking the derivative of both sides of* (30) *with respect to t yields*

$$\begin{cases} L.H.S. & = R_s^\top R_t \left(J(A^{s\to t}) \frac{d}{dt} A^{s\to t}\right)^\wedge \\ R.H.S. & = R_s^\top R_t \Omega(t, R_t, x_t) \end{cases}. \tag{31}$$

Since $R_s^\top R_t$ is invertible, equating the two sides and applying $(\cdot)^\vee$ gives

$$J(A^{s\to t})\, \tfrac{d}{dt} A^{s\to t} = \omega_t \qquad \Longleftrightarrow \qquad \tfrac{d}{dt} A^{s\to t} = J^{-1}(A^{s\to t})\, \omega_t, \tag{32}$$

the two forms underlying (11) *and* (12) *respectively; the equivalence uses invertibility of J (Remark 6). Expanding $\frac{d}{dt} A^{s\to t} = A^{avg} + (t-s)\frac{d}{dt} A^{avg}$ by* (38) *recovers* (8).

Proof. Differentiate both sides of (30) with respect to t.

For the right-hand side, the standard derivative rule for the time-ordered exponential gives

$$\begin{aligned} \frac{d}{dt}\,\mathcal{T}\exp\Big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau\Big) &= \Big(\mathcal{T}\exp\big(\int_s^t \Omega(\tau, R_\tau, x_\tau)\,d\tau\big)\Big)\,\Omega(t, R_t, x_t) \\ &= \exp(\Omega^{s\to t})\,\Omega(t, R_t, x_t), \end{aligned} \tag{33}$$

where $\exp(\Omega^{s\to t}) = R_s^\top R_t$ by definition.

For the left-hand side, the Wilcox formula for the Fréchet derivative of the matrix exponential gives

$$\begin{aligned} \frac{d}{dt}\exp(\Omega^{s\to t}) &= \exp(\Omega^{s\to t}) \int_0^1 \exp\big(-\alpha\Omega^{s\to t}\big)\Big(\frac{d}{dt}\Omega^{s\to t}\Big)\exp\big(\alpha\Omega^{s\to t}\big)\,d\alpha \qquad (34) \\ &= \exp(\Omega^{s\to t}) \int_0^1 \exp\big(-\alpha\Omega^{s\to t}\big)\Big(\frac{d}{dt}A^{s\to t}\Big)^\wedge \exp\big(\alpha\Omega^{s\to t}\big)\,d\alpha \\ &= \exp(\Omega^{s\to t}) \int_0^1 \left(\exp\big(-\alpha(A^{s\to t})^\wedge\big)\,\frac{d}{dt}A^{s\to t}\right)^\wedge d\alpha \\ &= \exp(\Omega^{s\to t}) \left(\int_0^1 \exp\big(-\alpha(A^{s\to t})^\wedge\big)\,d\alpha\,\frac{d}{dt}A^{s\to t}\right)^\wedge. \qquad (35) \end{aligned}$$

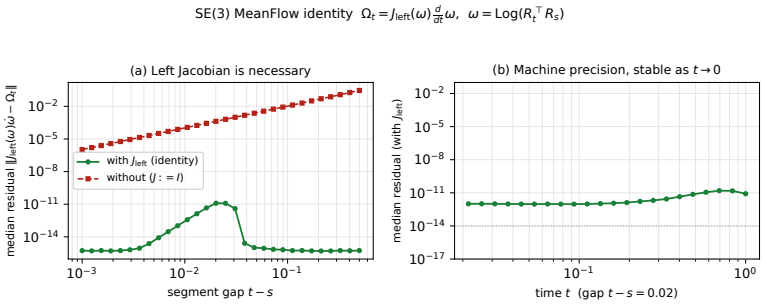

Figure 5: Numerical verification of Proposition 3. On an analytic, non-geodesic SO(3) path we evaluate both sides of (31) and plot the median residual $\|J(A^{s\to t})\,\frac{d}{dt}A^{s\to t} - \omega(t)\|$ against the segment gap $t - s$. With the Jacobian term $J(A^{s\to t})$ (green) the residual stays at float64 machine precision ($\sim 10^{-15}$, at worst 10^{-11}), uniformly in the gap; dropping it ($J := I$, red) breaks the identity by $\mathcal{O}(t - s)$, confirming both the proposition and the necessity of the Jacobian term.

Here the second line uses $\Omega^{s\to t} = (A^{s\to t})^\wedge$ and the linearity of $(\cdot)^\wedge$. The third line uses the conjugation identity $Qx^\wedge Q^\top = (Qx)^\wedge$ for $Q \in \text{SO}(3)$ and $x \in \mathbb{R}^3$. The last line follows because $(\cdot)^\wedge$ is linear and independent of α. We complete the proof by the identity of right Jacobian defined in (10)

$$J(A^{s\to t}) = \int_0^1 \exp(-\alpha(A^{s\to t})^\wedge)d\alpha.$$

□

Remark 6 (Inverse of the right Jacobian). Since $A^\wedge$ is skew with eigenvalues $0, \pm i\phi$ where $\phi := \|A\|_2$, the matrix $J(A) = \int_0^1 e^{-\alpha A^\wedge} d\alpha$ has eigenvalues 1 and $(1 - e^{\mp i\phi})/(\pm i\phi)$, of modulus $2\sin(\phi/2)/\phi$, all nonzero for $\phi \le \pi$. Hence J is invertible on the principal branch, with

$$J(A)^{-1} = I + \tfrac{1}{2}A^\wedge + \Big(\frac{1}{\phi^2} - \frac{1 + \cos\phi}{2\phi\sin\phi}\Big)(A^\wedge)^2,$$

and $J(A)^{-1} = I + \frac{1}{2}A^\wedge + \frac{1}{12}(A^\wedge)^2 + O(\phi^4)$ for small ϕ, which we use below the numerical threshold. This covers the model output: with the endpoint parameterization $A_\theta^{\text{avg}} = \frac{1}{t}\log(\cdot)^\vee$ of Section 3.1 the principal branch gives $\|A_\theta^{s\to t}\|_2 = \frac{t-s}{t}\,\|\log(\cdot)^\vee\|_2 \le \pi$, so (12) is well-defined.

Remark 7. If the angular velocity is constant, i.e. $\Omega_\tau = \omega^\wedge$ for all τ, then $A^{s\to t} = (t - s)\omega$ and $\frac{d}{dt}A^{s\to t} = \omega$. Since $\omega^\wedge\omega = \omega \times \omega = 0$, every term of the series for J beyond the constant one annihilates ω, so

$$J(A^{s\to t})\,\tfrac{d}{dt}A^{s\to t} = J\big((t - s)\omega\big)\,\omega = \omega,$$

and likewise $J^{-1}(A^{s\to t})\,\omega = \omega$: the Jacobian does not distort the velocity when the axis is fixed, and the two forms of (32) coincide.

We verify Proposition 3 numerically. We sample a smooth, non-geodesic curve $R(\tau) = \exp\big((a + b\tau + c\tau^2)^\wedge\big)$ on SO(3) with b, c non-parallel, so the averaged generator $A^{s\to t} = \text{Log}(R_s^\top R_t)$ genuinely differs from the instantaneous one and $J(A^{s\to t})$ acts non-trivially. For a batch of times we compute the instantaneous body angular velocity $\omega(t) := \big(\Omega(t)\big)^\vee = (R_t^\top \dot{R}_t)^\vee$ and the derivative $\frac{d}{dt}A^{s\to t}$ by forward-mode automatic differentiation, and evaluate $J(A^{s\to t})$ from its closed form, all using the same SO(3) exp/log/Jacobian routines as the training loss in double precision. Figure 5 reports the median residual of (31) over the batch versus $t - s$.

The identity in Proposition 3 involves the total time derivative $\frac{d}{dt}A^{s\to t}$, which we now compute explicitly.

Proposition 4. *In the above notation, we have the total derivative*

$$\frac{d}{dt}A(s,t,R_t,x_t) = \partial_t A(s,t,R_t,x_t) + \langle \nabla_R A(s,t,R_t,x_t), \dot{R}_t\rangle + \langle \nabla_x A(s,t,R_t,x_t), \dot{x}_t\rangle.$$

Proof. By the above remark, for $v \in T_{R_t}\mathrm{SO}(3)$,

$$d_R A(s,t,R_t,x_t)[v] = \nabla_R A(s,t,R_t,x_t)[v].$$

Similarly, in the Euclidean component, $d_x A(s,t,R_t,x_t)[w] = \langle \nabla_x A(s,t,R_t,x_t), w\rangle$ for $w \in \mathbb{R}^3$. Applying the chain rule along the curve $t \mapsto (R_t, x_t)$ gives the claimed identity. □

Substituting $A_\theta^{\mathrm{avg}}(s,t,R_t,x_t)$ into the two identities of (32) and defining $A_\theta^{s\to t} = (t-s)A_\theta^{\mathrm{avg}}(s,t,R_t,x_t)$, we obtain the two training losses of Section 3.1:

$$\mathcal{L}^{J}_{\mathrm{SO}(3)} := \mathbb{E}_{s<t,\,((R_0,x_0),(R_1,x_1))\sim q_{0,1}}\Big[\big\|\,\mathrm{sg}\big(J(A_\theta^{s\to t})\big)\big(A_\theta^{\mathrm{avg}} + (t-s)\,\mathrm{sg}(\tfrac{d}{dt}A_\theta^{\mathrm{avg}})\big) - \omega_t\,\big\|_2^2\Big], \tag{36}$$

$$\mathcal{L}^{J^{-1}}_{\mathrm{SO}(3)} := \mathbb{E}_{s<t,\,((R_0,x_0),(R_1,x_1))\sim q_{0,1}}\Big[\big\|\,A_\theta^{\mathrm{avg}} - \mathrm{sg}\big(J^{-1}(A_\theta^{s\to t})\,\omega_t - (t-s)\,\tfrac{d}{dt}A_\theta^{\mathrm{avg}}\big)\,\big\|_2^2\Big], \tag{37}$$

where $q_{0,1}$ is a coupling between $p_0 = p_{data}$ and $p_1 = p_{prior} = p_{noise}$ (e.g. product measure or optimal transport coupling), and the stop-gradient is placed so that the gradient flows through A_θ^{avg} only. The two residuals are related by $J(A_\theta^{s\to t})$ and hence share their zero set; we train with (37). In addition, the interpolation $R(t)$ (and its velocity $\dot{R}(t)$) is obtained from the geodesic interpolation:

$$\Omega(t) = \mathrm{Log}(R_0^\top R_1), \quad R(t) = R(0)\exp(t\Omega(t)), \quad \dot{R}_t = R_t\Omega(t).$$

The following proposition confirms that minimizing this loss recovers the correct relative rotation between any two points along the path.

Proposition 5. *If $\mathcal{L}^{J}_{\mathrm{SO}(3)} = 0$ or $\mathcal{L}^{J^{-1}}_{\mathrm{SO}(3)} = 0$, i.e.*

$$\big(J(A_\theta^{s\to t})\tfrac{d}{dt}A_\theta^{s\to t}\big)^\wedge = \Omega(t,R_t,x_t) \qquad \forall s<t\in[0,1],$$

then

$$\exp\big((A_\theta^{s\to t})^\wedge\big) = R_s^\top R_t.$$

Proof. For simplicity we take $s=0, t=1$. Define the candidate reconstruction

$$\tilde{R}_t := R_0\exp((A_\theta^{0\to t})^\wedge).$$

By the closed-form Fréchet derivative of the exponential on SO(3),

$$\frac{d}{dt}\exp((A_\theta^{0\to t})^\wedge) = \exp((A_\theta^{0\to t})^\wedge)\big(J(A_\theta^{0\to t})\tfrac{d}{dt}A_\theta^{0\to t}\big)^\wedge.$$

Using the condition $\mathcal{L}^{J}_{\mathrm{SO}(3)} = 0$ (or $\mathcal{L}^{J^{-1}}_{\mathrm{SO}(3)} = 0$) we have

$$\big(J(A_\theta^{0\to t})\tfrac{d}{dt}A_\theta^{0\to t}\big)^\wedge = \Omega(t,R_t,x_t).$$

Hence

$$\dot{\tilde{R}}_t = \tilde{R}_t\Omega(t,R_t,x_t).$$

On the other hand the geodesic interpolation satisfies

$$\dot{R}_t = R_t \Omega(t, R_t, x_t), \qquad R_0 \text{ given.}$$

Thus R_t and $\tilde{R}_t$ solve the same ODE with the same initial condition. By uniqueness of solutions on SO(3) we obtain

$$\tilde{R}_t = R_t, \qquad \forall t \in [0, 1].$$

Taking $t = 1$ yields

$$R_0 \exp((A_\theta^{0\to 1})^\wedge) = R_1 \quad \Longleftrightarrow \quad \exp((A_\theta^{0\to 1})^\wedge) = R_0^\top R_1.$$

For general $s < t$ the same argument gives

$$\exp((A_\theta^{s\to t})^\wedge) = R_s^\top R_t,$$

which completes the proof. □

To implement the loss in (36), it remains to compute the total derivative $\frac{d}{dt} A_\theta^{s\to t}$.

Proposition 6. *From the product rule and chain rule, we have*

$$\frac{d}{dt}(t-s)A_\theta(s,t,R_t,x_t) = A_\theta(s,t,R_t,x_t) + (t-s)\Big(\partial_t A_\theta(s,t,R_t,x_t) + \langle \nabla_R A_\theta(s,t,R_t,x_t), \dot{R}_t\rangle + \langle \nabla_x A_\theta(s,t,R_t,x_t), \tag{38}$$

Proof. Fix s, t and (R_t, x_t). For convenience, we denote

$$A_\theta := A_\theta(s, t, R_t, x_t).$$

By definition, $(t-s)A_\theta$ is a product of the scalar function $t - s$ and the vector-valued function $A_\theta(s, t, R_t, x_t)$. Hence, by the product rule,

$$\begin{aligned} \frac{d}{dt}(t-s)A_\theta &= \frac{d}{dt}(t-s)\, A_\theta + (t-s)\frac{d}{dt}A_\theta \\ &= A_\theta + (t-s)\frac{d}{dt}A_\theta. \end{aligned} \tag{39}$$

Next, we compute the total derivative of $A_\theta(s, t, R_t, x_t)$ with respect to t. Since A_θ depends on t explicitly and implicitly through both R_t and x_t, by the chain rule,

$$\frac{d}{dt}A_\theta(s,t,R_t,x_t) = \partial_t A_\theta(s,t,R_t,x_t) + d_R A_\theta(s,t,R_t,x_t)[\dot{R}_t] + d_x A_\theta(s,t,R_t,x_t)[\dot{x}_t]. \tag{40}$$

Substituting (40) into (39) yields

$$\frac{d}{dt}(t-s)A_\theta(s,t,R_t,x_t) = A_\theta + (t-s)\Big(\partial_t A_\theta + \langle \nabla_R A_\theta, \dot{R}_t\rangle + \langle \nabla_x A_\theta, \dot{x}_t\rangle\Big).$$

This proves (38). □

F.2 Practical implementation: $\mathrm{SE}(3)^N$ adaptation

In practice, each sample of the protein data is in shape $R^N \in \mathbb{R}^{N\times 3\times 3}, x^N \in \mathbb{R}^{N\times 3}$ and N can be treated as the dimension of the data. The model is defined as

$$f_\theta : (R_t^N, x_t^N, t, s) = [A_\theta^{avg,N}; v_\theta^{avg,N}] \in \mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times 3}.$$

Here $A_{\theta,i}^{avg,N} \in \mathbb{R}^3$ parametrizes the i-th $\mathfrak{so}(3)$ component via $\Omega_{\theta,i}(s,t,R_{t,i}) = (A_{\theta,i}^{avg,N}(s,t,R_{t,i}))^\wedge$.

Independent $\mathfrak{so}(3)^N$ loss. Extending (36) to $\mathfrak{so}(3)^N$ by treating each component independently gives

$$\mathcal{L}_{\mathfrak{so}(3)^N} := \mathbb{E}_{\substack{s<t,\\ q(R_0^N, R_1^N)}} \sum_{i=1}^{N} \left\| \mathrm{sg}\Big(J\big((t-s)A_{\theta,i}^{\mathrm{avg}}\big)\Big)\Big(A_{\theta,i}^{\mathrm{avg}} + (t-s)\,\mathrm{sg}\big(\tfrac{d}{dt}\hat{A}_{\theta,i}^{\mathrm{avg}}\big)\Big) - \omega_{t,i} \right\|_2^2, \tag{41}$$

where $\omega_{t,i} := \Omega_i(t, R_{t,i}, x_{t,i})^\vee$ is the data instantaneous angular velocity (cf. Eq. (7)) and $J(\cdot)$ is the SO(3) right Jacobian applied per component. The J^{-1} form (37) extends in the same way, with the i-th summand replaced by $\big\|A_{\theta,i}^{\mathrm{avg}} - \mathrm{sg}\big(J^{-1}\big((t-s)A_{\theta,i}^{\mathrm{avg}}\big)\omega_{t,i} - (t-s)\tfrac{d}{dt}\hat{A}_{\theta,i}^{\mathrm{avg}}\big)\big\|_2^2$.

Translation $\mathbb{R}^{3\times N}$ loss. For the translation component, the loss in Euclidean space (3) extends directly to N independent particles. With the model prediction $v_{\theta,i}^{\mathrm{avg}} \in \mathbb{R}^3$ for residue i, the translation loss is

$$\mathcal{L}_{\mathbb{R}^{3\times N}} := \mathbb{E}_{\substack{s<t,\\ q(x_0^N, x_1^N)}} \sum_{i=1}^{N} \left\| v_{\theta,i}^{\mathrm{avg}} + (t-s)\,\mathrm{sg}\left(\frac{d}{dt}\hat{v}_{\theta,i}^{\mathrm{avg}}\right) - v_i(t, x_{t,i}) \right\|_2^2, \tag{42}$$

where $v_i(t, x_{t,i}) = x_{1,i} - x_{0,i}$ is the constant ground-truth translation velocity along the linear interpolation path $x_{t,i} = (1-t)x_{0,i} + tx_{1,i}$. The combined MeanFlow loss on $\mathrm{SE}(3)^N$ is therefore $\mathcal{L}_{MF}^{\mathrm{SE}(3)^N} = \mathcal{L}_{\mathfrak{so}(3)^N} + \mathcal{L}_{\mathbb{R}^{3\times N}}$.

G Common Settings: Auxiliary Loss, Prior and Inference.

G.1 Auxiliary Loss

We include auxiliary losses from Yim et al. [2023b] to enforce geometric consistency at the atomic level. Specifically, let $\mathcal{A}_0 \in \mathbb{R}^{N\times 4\times 3}$ denote the ground-truth backbone atom coordinates (in Å) of the four heavy atoms $(\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{O})$ per residue, and let $\hat{\mathcal{A}}_0$ be the corresponding coordinates predicted by the model. We define a direct regression loss on backbone atom positions and a pairwise distance loss in a local neighbourhood,

$$\mathcal{L}_{\mathrm{bb}} = \frac{1}{4N}\sum \|\mathcal{A}_0 - \hat{\mathcal{A}}_0\|^2, \qquad \mathcal{L}_{\mathrm{2D}} = \frac{\|\mathbf{1}\{D < 6\text{Å}\}(D - \hat{D})\|^2}{\sum \mathbf{1}_{D<6\text{Å}} - N}, \tag{43}$$

where $D \in \mathbb{R}^{N\times N\times 4\times 4}$ is the tensor of pairwise distances between heavy atoms, i.e. $D_{ijab} = \|\mathcal{A}_{ia} - \mathcal{A}_{jb}\|$, and $\hat{D}$ is defined analogously from $\hat{\mathcal{A}}_0$. The auxiliary loss is then

$$\mathcal{L}_{\mathrm{aux}} = \mathbb{E}_{\mathcal{Q}}[\mathcal{L}_{\mathrm{bb}} + \mathcal{L}_{\mathrm{2D}}], \tag{44}$$

where $\mathcal{Q}(t, x_0, x_1, \tilde{x}_t) := \mathcal{U}(0,1) \otimes \bar{\pi}(x_0, x_1) \otimes \rho_t(\tilde{x}_t \mid x_0, x_1)$ is the factorized joint distribution. And following the settings in Bose et al. [2024], Yim et al. [2023b], we only apply $\mathcal{L}_{\mathrm{aux}}$ for $t < 0.75$, scaling it by λ_{aux}.

G.2 Prior Distribution

Following the notation in Geng et al. [2026a,b], we denote protein backbones by (R_0^N, x_0^N) and sample $(R_1^N, x_1^N) \in \mathrm{SO}(3)^N \times \mathbb{R}^{N\times 3}$ from a simple prior.

Algorithm 1: MeanFlow training for $\mathrm{SE}(3)^N$ (idealized form). This is the plain MeanFlow objective, stated to make the structure of the method explicit; it is *not* the recipe used for our reported results. The trained objective is the Stage-2 loss (97)—which adds the endpoint anchor and the auxiliary term—with the rotation and translation losses computed in the small-t stabilized form of Algorithm 3, after the α-Flow warm-up of Algorithm 5. See Appendix M.

Require: Data distribution p_{data} over $\mathrm{SE}(3)^N$; MeanFlow network f_θ; OT-coupling flag

for each training iteration **do**
  Sample a mini-batch $\{(R_0^N, x_0^N)\}_{b=1}^B \sim p_{\text{data}}$.
  $R_0^N \in \mathbb{R}^{B\times N\times 3\times 3}$, $x_0^N \in \mathbb{R}^{B\times N\times 3}$.
  Sample (R_1^N, x_1^N) from the prior; see Section G.2.
  if OT-coupling **then**
    Solve $OT((R_0, x_0), (R_1, x_1))$ and couple (R_0, x_0) and (R_1, x_1) using the optimal transport plan.
  end if
  Sample times $0 \le s < t \le 1$.
  Forward-interpolate to time t to obtain (R_t^N, Ω_t^N) and (x_t^N, v_t^N).
  Evaluate the network: $[A_\theta^{avg,N}, v_\theta^{avg,N}] \leftarrow f_\theta(s, t, R_t^N, x_t^N)$.
  Compute $\mathcal{L}_{\mathfrak{so}(3)^N}$ via Eq. (41).
  Compute $\mathcal{L}_{\mathbb{R}^{3\times N}}$ via Eq. (42).
  Compute the auxiliary loss $\mathcal{L}_{\text{aux}}$; see Section G.1.
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathfrak{so}(3)^N} + \mathcal{L}_{\mathbb{R}^{3\times N}} + \lambda_{\text{aux}} \mathbb{1}[t < 0.75]\, \mathcal{L}_{\text{aux}}$.
  Update θ using a gradient step (or any optimizer).
end for

Translation prior. In Euclidean space, the natural analogue of a "standard" prior is an isotropic Gaussian,

$$x_1^N \sim \mathcal{N}\big(0, \sigma_x^2 I_{N\times 3}\big), \tag{45}$$

where we identify $x_1^N \in \mathbb{R}^{N\times 3}$ with a vector in $\mathbb{R}^{3N}$ and typically set $\sigma_x = 1$.

Rotation prior: Gaussian analogue on $\mathrm{SO}(3)$**.** On the rotation group, the closest analogue of an isotropic Gaussian is an *isotropic (heat-kernel) Gaussian* on $\mathrm{SO}(3)$, denoted $\mathrm{IGSO}(3)(\sigma_R)$. It is the transition density of Brownian motion on $\mathrm{SO}(3)$ at time σ_R^2, and is isotropic in the sense that its density depends only on the geodesic rotation angle

$$\theta(R) := \|\mathrm{Log}(R)\|_2 \in [0, \pi], \qquad \mathrm{Log} : \mathrm{SO}(3) \to \mathfrak{so}(3) \cong \mathbb{R}^3, \tag{46}$$

with respect to the Haar measure $\mathrm{d}R$.

Let $IGSO(3)^N$ denote the product measure of $IGSO(3)$. Concretely, we use the factorized prior

$$p(R_1^N, x_1^N) = IGSO(3)(\sigma_R)^N \times \mathcal{N}(0, I_{N\times 3}). \tag{47}$$

The typical choice of σ_R is 1.5. In addition, as $\sigma_R \to \infty$, $\mathrm{IGSO}(3)(\sigma_R)$ approaches the uniform (Haar) distribution on $\mathrm{SO}(3)$, which is another typical choice of the prior.

G.3 Inference process

Similar to the original MeanFlow method in Euclidean space, we can adapt multi-step inference for the $\mathrm{SE}(3)^N$ MeanFlow framework.

Algorithm 2: $\mathrm{SE}(3)^N$ MeanFlow inference

Require: Batch size B; pretrained network f_θ; number of steps T
Ensure: $(\hat{R}_0^N, \hat{x}_0^N)$

$\Delta t = 1/T$
Sample (R_1^N, x_1^N) from the prior over $\mathrm{SE}(3)^N$.
$(R_t^N, x_t^N) \leftarrow (R_1^N, x_1^N)$.
for $t = 1, 1 - \Delta t, \ldots, \Delta t$ **do**
    $s \leftarrow t - \Delta t$.
    $[A_\theta^{avg,N}, v_\theta^{avg,N}] \leftarrow f_\theta(s, t, R_t^N, x_t^N)$.
    $R_t^N \leftarrow R_t^N \exp(-\Delta t\, (A_\theta^{avg,N})^\wedge)$.
    $x_t^N \leftarrow x_t^N - \Delta t\, v_\theta^{avg,N}$.
end for
$\hat{R}_0^N = R_t^N, \hat{x}_0^N = x_t^N$.

In particular, let $T \in \mathbb{N}$ denote the number of inference steps and let $\Delta t = \frac{1}{T}$ be the step size. We apply the following updates on $\mathrm{SO}(3)$ and $\mathbb{R}^3$, respectively:

$$R_{t-1} = R_t e^{-\Delta t A_\theta^\wedge(s,t,R_t,x_t)},$$
$$x_{t-1} = x_t - \Delta t v_\theta^{\mathrm{avg}}(s, t, R_t, x_t).$$

where t decreases from 1 to $1/T$ and $s = t - \Delta t$. We summarize the procedure in Algorithm 2.

H Ablation Study: A Controlled $\mathrm{SO}(3)$ Benchmark for Few-Step Generation

Designability on SCOPe confounds the generative objective with the protein-specific IPA trunk, the auxiliary losses, the self-conditioning recipe, and the folding oracle, so a few-step win there is hard to attribute. This appendix therefore tests the central claim of the paper — that it is the *average-velocity parameterisation* that enables few-step generation, rather than the network, coupling, or protein-specific machinery — on a controlled $\mathrm{SO}(3)^2$ benchmark, where we hold all confounders fixed and vary only the training loss.

A full ablation on real proteins is prohibitively expensive: the design space includes many hyperparameter combinations (e.g., OT variants in sampling such as per-GPU OT, global OT, or no OT; loss normalizations and clipping; different loss mixtures; architectural toggles such as enabling/disabling AdaLN; and optimizer / learning-rate choices). Each setting requires substantial compute (4×80GB GPUs), long GPU-hour budgets, and careful checkpoint selection to reliably judge performance. Under our limited training budget we therefore do *not* run exhaustive real-dataset ablations; instead, on the synthetic $\mathrm{SO}(3)^2$ benchmark we fix the model, learning rate, SE(3) interpolator, and training steps so that performance differences are attributable to the losses themselves.

H.1 Data

A sample is a pair of rotations $(R^{(0)}, R^{(1)}) \in \mathrm{SO}(3)^2$: the rotational part of a two-residue backbone, with the translation branch switched off. Throughout we visualise a rotation at its axis-angle

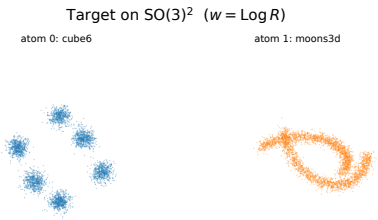

Figure 6: The $\mathrm{SO}(3)^2$ target, drawn in axis-angle coordinates $w = \mathrm{Log}(R)$, $\|w\| \le \pi$. **Left**, atom 0 (CUBE6): six sharp modes at the cube-face rotations — a mode-seeking task. **Right**, atom 1 (MOONS3D): two interlocking crescents — a curved-manifold task. Both must be solved by the same network under the same objective.

coordinate

$$w = \mathrm{Log}(R) \in \mathbb{R}^3, \qquad \|w\| \le \pi, \tag{48}$$

so a distribution on SO(3) becomes a point cloud inside the π-ball (Figure 6).

The two atoms carry deliberately different geometry.

Atom 0 (cube6): a mode-seeking task. A mixture of six isotropic components centred at the six cube-face rotations (rotations by $\pi/2$ about $\pm e_x, \pm e_y, \pm e_z$, hence $\|w\| = \pi/2$). The modes are sharp and well separated, so a sampler must *commit* to one basin; hedging between two modes is immediately visible as mass in the empty region between them.

Atom 1 (moons3d): a curved-manifold task. The classical two-moons distribution, lifted into the Lie algebra so that its support is a pair of interlocking crescents. Here the model must trace a thin, curved, one-dimensional structure rather than collapse onto a few points.

Prior. The prior is the Haar (uniform) measure on SO(3), independently per atom. A Haar draw sits a mean geodesic distance of 126.5° from the identity, so the transport distance is large and the prior carries no information about either target. We draw 10,000 training and 2,000 held-out samples; the noise is resampled every epoch, from a seeded stream (below).

H.2 Experiment setting

All methods share the following. Only the training loss differs.

1. **Network.** The same 1.07M-parameter SE(3) MLP for every method, mapping $(R_t, t, s) \mapsto \omega \in \mathbb{R}^{2\times 3}$, a body-frame angular velocity per atom. Single-time methods (FoldFlow) are fed $s{=}t$; endpoint methods (QFlow, V-QFlow) use the quaternion head of the same trunk, at matched rotation-branch capacity.

2. **Time sampler.** $t \sim \mathcal{U}(0,1)$ with *no* lower cut-off ($t_\varepsilon = 0$; the regular-t losses never divide by t), and $s \sim \mathcal{U}(0,t)$ for the two-time methods, with a 50% $s{=}t$ anchor branch. The three-time method (RMF semigroup) additionally draws an interior $m \in (s,t)$ and gets no $s{=}t$ branch, since a degenerate interval makes the semigroup identity vacuous.

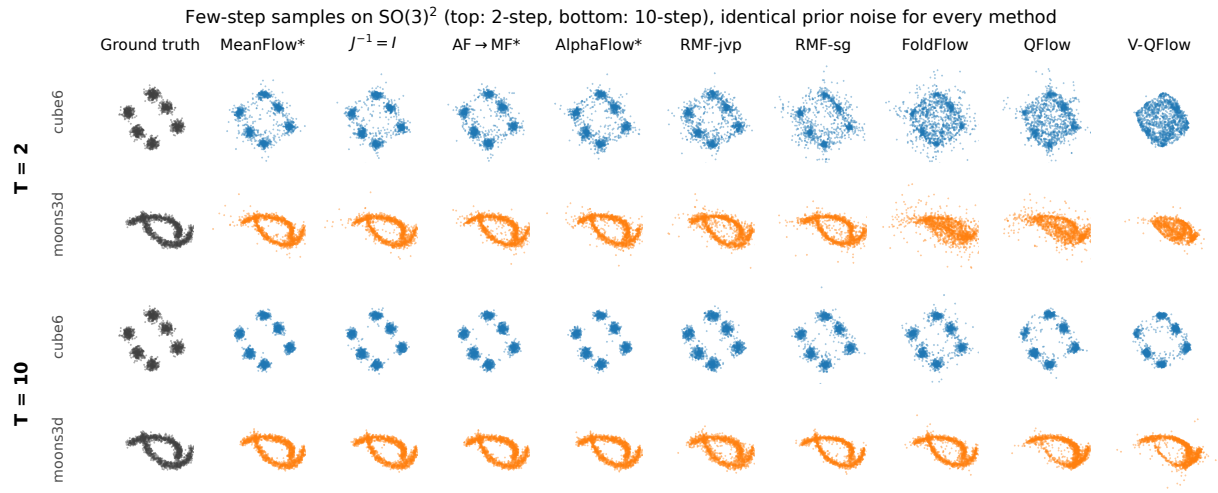

Figure 7: Two-step (T=2) samples, in the same axis-angle coordinates as Figure 6, from the *identical* frozen prior noise for every method. Top row: atom 0 (CUBE6); bottom row: atom 1 (MOONS3D). The average-velocity methods (ours and both RMF variants) already resolve the six modes and the crescent geometry after two steps. FoldFlow and QFlow smear across the ball, and V-QFlow contracts onto a shell.

3. **Budget.** 20k steps, batch 500, the same optimiser and learning rate, and EMA weights at evaluation.

4. **Data and noise stream.** *Bit-identical* across methods. This is enforced by three independent RNG streams — batch indices, coupling, and times — so that methods which consume different numbers of random draws (OT draws a multinomial; a three-time method draws three uniforms where a single-time method draws one) nevertheless see the same (R_0, R_1) pair at every step. With a single shared stream the sequences desynchronise after the first step, silently, and the comparison is no longer controlled.

No optimal-transport coupling. No method uses mini-batch OT. This is deliberate and it is the crux of the protocol: OT is a flow-*straightening* device. Under an OT coupling the learned probability path is close to a straight geodesic, in which case the average velocity and the instantaneous velocity nearly coincide — so a plain flow-matching model gets few-step generation for free, and the experiment can no longer distinguish the two parameterisations. Enabling OT would hand the instantaneous-velocity baselines exactly the property under test. We therefore run every method *without* OT, so that few-step behaviour is attributable to the objective alone.

H.3 Evaluation Metric

We report the Wasserstein-2 distance between 2,000 generated and 2,000 held-out samples, under the squared geodesic cost summed over the two atoms:

$$c\big((R^{(0)}, R^{(1)}), (S^{(0)}, S^{(1)})\big) = \sum_{a\in\{0,1\}} d^2_{\mathrm{SO}(3)}\big(R^{(a)}, S^{(a)}\big), \qquad d_{\mathrm{SO}(3)}(R, S) = \big\|\mathrm{Log}(R^\top S)\big\|, \tag{49}$$

solved as an exact assignment and reported in degrees.

Method	$\mathcal{W}_2$ (°) ↓ at T steps				
	T=1	T=2	T=5	T=10	T=20
Average-velocity parameterisation					
SE(3)-MeanFlow (ours)	**30.74**	**26.29**	25.83	25.77	25.95
ablation: $J^{-1} := I$ (log-map)	31.15	26.37	25.91	25.90	25.91
α-Flow → MeanFlow (ours)	31.13	26.67	25.30	25.31	25.35
α-Flow (ours)	32.16	26.81	25.63	25.51	25.63
RMF, JVP form Woo et al. [2026]	32.44	27.65	**25.26**	25.21	25.18
RMF, semigroup+FM Woo et al. [2026]	31.52	29.87	25.36	**24.75**	24.83
Instantaneous-velocity / endpoint parameterisation					
FoldFlow Bose et al. [2024]	80.52	43.74	26.82	24.94	**24.77**
QFlow Yue et al. [2025]	77.99	42.68	29.44	31.28	31.96
V-QFlow Yue et al. [2025]	90.64	51.35	31.75	32.93	32.24
FLOOR	21.70	21.70	21.70	21.70	21.70

Table 5: Few-step generation on the $\mathrm{SO}(3)^2$ toy benchmark. $\mathcal{W}_2$ (degrees; lower is better) between 2,000 generated and 2,000 held-out samples, as the number of Euler steps T falls from 20 to 1. All methods share one network, one data/noise stream, one time sampler and one budget (Section H.2); *none* uses an OT coupling. FLOOR is the $\mathcal{W}_2$ between two independent draws of the target and is the lowest attainable value. Best per column in bold. The separation is by *parameterisation*, not by method family: average-velocity methods survive T=1; instantaneous-velocity and endpoint-prediction methods collapse. Removing the SO(3) Jacobian ($J^{-1} := I$) leaves the average-velocity model unchanged within noise for $T \geq 2$ and costs 0.4° at T=1, where the sampler queries the largest interval.

The floor. At finite sample size, $\mathcal{W}_2$ between two *independent* draws of the target is not zero. We measure this floor with the identical estimator and the identical sample size, and report it in Table 5: 21.70°. No model can go below it, and a model at the floor is indistinguishable from the target at this sample size.

Sampling. Models are integrated on a uniform Euler grid $t \in \mathrm{linspace}(1, 0, T)$ for $T \in \{1, 2, 5, 10, 20\}$, with no final-step special case and no $t_{\min}$ floor. Every method is initialised from the same frozen prior-noise batch, so the sample sets — and hence the panels of Figure 7 — are directly comparable.

H.4 Results

Table 5 shows a clean separation by *parameterisation*, not by method family or by method quality.

20 steps: the fine-grid regime. At T=20 the two parameterisations are indistinguishable in quality: every average-velocity method, together with FoldFlow, sits within a few degrees of the 21.70° floor, and the single best number is in fact FoldFlow's (24.77°). With a fine enough Euler grid the instantaneous-velocity parameterisation is entirely adequate and our objective buys nothing. The exceptions are the quaternion-parameterised methods, QFlow and V-QFlow, which plateau roughly 10° above the floor. Since QFlow optimises a flow-matching loss and V-QFlow an endpoint loss, both should in principle track FoldFlow in this regime; we attribute the gap to the rotation-matrix backbone used throughout this ablation, which forces repeated quaternion–matrix conversions inside the model and the loss. Appendix L.4 reports the same effect at protein scale.

Few steps: MeanFlow methods outperform flow matching. At T=2 (and T=5), the MeanFlow-style average-velocity methods — ours and both RMF variants — already approach the 21.70° floor, with our model being slightly better overall. In contrast, the flow-matching baselines FoldFlow, QFlow and V-QFlow exhibit substantially higher errors at these low step counts. Figure 7 corroborates this qualitatively: after two steps our samples already show six distinct modes and a recognisable pair of crescents, whereas FoldFlow and QFlow do not.

At T=1, SE(3)-MeanFlow remains the best method, but all approaches are relatively far from the floor. This is expected: in the extreme case where two rotations differ by an angle of π, the shortest geodesic (and hence the induced velocity) is not well-defined due to non-uniqueness (analogous to the ambiguity of shortest paths between the north and south poles on a sphere). This suggests that SO(3) is intrinsically ill-suited for one-step generation. However, few-step generation methods can still be applied.

Effect of the right Jacobian. Replacing J^{-1} by the identity — the log-map approximation of Zhong et al. [2026] — costs 0.41° at T=1 and is within 0.13° of our model elsewhere (Table 5). This is the expected pattern: since $J^{-1}(A) = I + \frac{1}{2}A^{\wedge} + \cdots$ acts on $A^{s\to t} = (t-s)A^{\text{avg}}$, the correction scales with the interval, and a T-step sampler queries only $t - s = 1/T$. The same scaling holds on real training data (Table 9). The Jacobian is closed-form and free, so we retain it; its benefit is concentrated in the aggressive few-step regime.

The α-Flow ablation isolates the mechanism. Our α-Flow objective (Section J), trained alone with α annealed $1 \to 0.1$, performs competitively, but is consistently slightly worse than SE(3)-MeanFlow at low step counts (Table 5). This is expected: α-Flow is a JVP-free surrogate that relaxes the full average-velocity consistency enforced by MeanFlow. Appending a MeanFlow phase to the last 40% of the same schedule — same network, same budget, same data — improves the one-step error by about 1°, with smaller gains at T=2, indicating that few-step capability is primarily supplied by the MeanFlow objective. Training with MeanFlow throughout gives the better one- and two-step error, while the warm-up variant is marginally better for $T \geq 5$; neither ordering is large relative to the within-block spread. The few-step advantage reported in the main text is therefore not an artefact of a short fine-tuning phase.

Concurrent work, Riemannian Meanflow. Both RMF variants Woo et al. [2026] sit in the same block as our method and perform comparably (RMF semigroup+FM reaches 31.52° at T=1, statistically indistinguishable from our 30.74° at this sample size). We regard this as supporting the paper's thesis rather than undercutting it: the thesis is about the average-velocity parameterisation, and RMF is an average-velocity method. The SO(3)-specific contribution of our work — the exact right-Jacobian identity of Proposition 1 and the resulting $\mathrm{SE}(3)^N$ objective — is evaluated on protein backbones in the main text, where the two methods do separate.

Caveat. The numbers in Table 5 are from a single seed. The $\sim$50° gap between the two blocks is far beyond any plausible seed variation, but small differences *within* a block (e.g. our 30.74° versus RMF's 31.52° at T=1) should not be read as a ranking.

I Stable Training Framework, Part 1: MeanFlow Loss (Vanilla and Small-t Stabilized)

Vanilla (no small-t reparameterisation). The MeanFlow identity on SO(3) (Proposition 1, with full derivation in Appendix F.1) yields a consistency target involving the trajectory derivative $\frac{d}{dt}A_\theta^{\text{avg}}$. Our practical deep learning model is *diff-frame*: it takes the current state (R_t, x_t) and a time embedding (constructed from (t, s)) as input, and directly predicts the endpoints $(\hat{R}_0^\theta, \hat{x}_0^\theta)$. We then define the average velocities using the endpoint ("x-prediction") parameterisation

$$A_\theta^{\text{avg}}(R_t, x_t, t, s) = \frac{1}{t}\,\log\big((\hat{R}_0^\theta)^\top R_t\big)^\vee, \tag{50}$$

$$v_\theta^{\text{avg}}(R_t, x_t, t, s) = \frac{1}{t}\,(x_t - \hat{x}_0^\theta), \tag{51}$$

which is the SE(3) analogue of pixel-space MeanFlow. This is the "normal" objective; if one samples times bounded away from 0 and computes $\frac{d}{dt}A_\theta^{\text{avg}}$ via a JVP, it can be used directly.

Motivation: small-t numerical stability. Both branches contain explicit $1/t$ factors and the rotation target further requires differentiating through $\log((\hat{R}_0^\theta)^\top R_t)$, so naive autodiff becomes numerically fragile as $t \to 0$. Below we describe our implementation that preserves the same objective but avoids explicit $1/t$ during target construction.

Positive $t_{\min}$ and last-step endpoint prediction in baselines

Following prior work we enforce a strictly positive $t_{\min} > 0$ and only sample training times from $[t_{\min}, 1]$. In our method we extend the lower bound to $t_{\min} = 10^{-6}$.

At inference time, we integrate on a fixed grid $t \in \text{linspace}(1, t_{\min}, T)$. In the final iteration, we set $(t, s) = (t_{\min}, 0)$ and directly use the model's endpoint prediction $(\hat{R}_0^\theta, \hat{x}_0^\theta)$ as the generated sample rather than performing another Euler update. For the small-t consistency losses we use a denominator clamp t_ε when dividing by t^2 (only applied at the final loss normalization step).

I.1 Auxiliary B-variables for stable time derivatives

The MeanFlow consistency loss involves a time derivative term of the form $(t - s)\,\frac{d}{dt}A_\theta^{\text{avg}}$ (and similarly for v_θ^{avg}). Directly differentiating $A_\theta^{\text{avg}} = \frac{1}{t}\log((\hat{R}_0^\theta)^\top R_t)^\vee$ can lead to large gradients when t is small.

We therefore introduce auxiliary variables that absorb the problematic factor t,

$$B_\theta^A(R_t, x_t, t, s) := t\,A_\theta^{\text{avg}}(R_t, x_t, t, s), \tag{52}$$

$$B_\theta^v(R_t, x_t, t, s) := t\,v_\theta^{\text{avg}}(R_t, x_t, t, s), \tag{53}$$

and compute time-derivative information via $\frac{d}{dt}B_\theta^A$ and $\frac{d}{dt}B_\theta^v$. In implementation we avoid dividing by t throughout the derivative-target construction; all $1/t$ factors are deferred, and we only apply the t^{-2} normalization after forming the squared residuals (with t clamped below by t_ε). Using $B_\theta^A = tA_\theta^{\text{avg}}$ and $h = t - s$, one can rewrite

$$\begin{aligned} A_\theta^{s\to t} &= h\,A_\theta^{\text{avg}} = \frac{h}{t}\,B_\theta^A, \\ t\,\frac{d}{dt}A_\theta^{s\to t} &= B_\theta^A + h\Big(\frac{d}{dt}B_\theta^A - A_\theta^{\text{avg}}\Big) = B_\theta^A + \Big(h\,\frac{d}{dt}B_\theta^A - \frac{h}{t}B_\theta^A\Big). \end{aligned} \tag{54}$$

where $\frac{h}{t} \leq 1$. In implementation, we absorb the factor h into the JVP tangents, so the computed JVP output corresponds to $h\,\frac{d}{dt}B_\theta^A$ directly (and analogously for B_θ^v). Here $\frac{d}{dt}B_\theta^A$ is the *total* time derivative of $B_\theta^A(R_t, x_t, t, s)$ (with s held fixed). By the chain rule,

$$h\frac{d}{dt}B_\theta^A = \frac{\partial B_\theta^A}{\partial t} + \Big\langle \frac{\partial B_\theta^A}{\partial R_t},\, h\frac{dR_t}{dt} \Big\rangle + \Big\langle \frac{\partial B_\theta^A}{\partial x_t},\, h\frac{dx_t}{dt} \Big\rangle, \tag{55}$$

where $\frac{\partial B_\theta^A}{\partial R_t}$ and $\frac{\partial B_\theta^A}{\partial x_t}$ include the implicit dependence through the network prediction $(\hat{R}_0^\theta, \hat{x}_0^\theta) = f_\theta(R_t, x_t, t, s)$. In practice we compute $\frac{d}{dt}B_\theta^A$ (and $\frac{d}{dt}B_\theta^v$) with a JVP given the instantaneous velocities $(\frac{dR_t}{dt}, \frac{dx_t}{dt})$.

which avoids explicitly forming $\frac{d}{dt}A_\theta^{\text{avg}}$. We apply the same construction to translation with $B_\theta^v = t\, v_\theta^{\text{avg}}$.

Small-t consistency losses (implementation form). To match the implementation, we form the residuals in a scaled way and defer all divisions by t to the final normalization step. Let $(t-s)$ denote the step size and define

$$A_\theta^{s\to t} := (t-s)A_\theta = \frac{t-s}{t}B_\theta^A, \qquad v_\theta^{s\to t} := (t-s)v_\theta = \frac{t-s}{t}B_\theta^v, \tag{56}$$

where $B_\theta^A = tA_\theta^{\text{avg}}$ and $B_\theta^v = tv_\theta^{\text{avg}}$. With the $(t-s)$-absorbed JVP outputs $(t-s)\,\frac{d}{dt}B_\theta^A$ and $(t-s)\,\frac{d}{dt}B_\theta^v$, we construct the scaled derivative targets

$$\begin{aligned} t\,\frac{d}{dt}A_\theta^{s\to t} &:= B_\theta^A + \text{sg}\left((t-s)\,\frac{d}{dt}B_\theta^A - \frac{t-s}{t}B_\theta^A\right), \\ t\,\frac{d}{dt}x_\theta^{s\to t} &:= B_\theta^v + \text{sg}\left((t-s)\,\frac{d}{dt}B_\theta^v - \frac{t-s}{t}B_\theta^v\right). \end{aligned} \tag{57}$$

where the JVP-derived directional derivatives $(t-s)\,\frac{d}{dt}B_\theta^A$ and $(t-s)\,\frac{d}{dt}B_\theta^v$ are computed as a single Jacobian–vector product of $f_\theta(R_t, x_t, t, s)$:

$$\begin{aligned} &(B_\theta^A, B_\theta^v, \ldots),\ \big((t-s)\,\tfrac{d}{dt}B_\theta^A,\ (t-s)\,\tfrac{d}{dt}B_\theta^v, \ldots\big) \\ &\qquad := \ \text{JVP}_{(\dot{R}_t, \dot{x}_t, \dot{t}, \dot{s})}\big[f_\theta(R_t, x_t, t, s)\big]. \end{aligned} \tag{58}$$

with the (step-size absorbed) input tangent

$$\begin{cases} \dot{R}_t = R_t\,\big((t-s)\,\Omega_t\big), \\ \dot{x}_t = (t-s)\,v_t \end{cases} \qquad \text{(MF)} \tag{59}$$

or, for IMF where the instantaneous velocities are taken from a single shared forward at (t,t),

$$\begin{cases} \dot{R}_t = R_t\,\big(\frac{t-s}{t}\,(B_\theta^{\text{A}}(t,t,R_t,x_t))^\wedge\big), \\ \dot{x}_t = \dfrac{t-s}{t}B_\theta^v(t,t,R_t,x_t) \end{cases} \qquad \text{(IMF)}. \tag{60}$$

The time tangents are $\dot{t} = (t-s)$ in the small-t parameterization (otherwise $\dot{t} = 1$), and $\dot{s} = 0$. The per-sample small-t losses (summing over residues) are computed from B_θ^A, B_θ^v and the JVP-derived

Algorithm 3: Stable X-prediction style Mean-flow training loss

Require: State (R_t, x_t); times $s < t$; instantaneous velocities (Ω_t, v_t) (MF) or $(B_\theta^A(t,t), B_\theta^v(t,t))$ (IMF); network f_θ; clamp t_ε; Jacobian placement $\texttt{jmode} \in \{J, J^{-1}\}$

$\Delta t \leftarrow t - s$

Construct the (step-size absorbed) input tangent:

(MF) $\dot{R}_t \leftarrow R_t(\Delta t\, \Omega_t)$, $\dot{x}_t \leftarrow \Delta t\, v_t$

(IMF) $\dot{R}_t \leftarrow R_t(\frac{\Delta t}{t}\,(B_\theta^A(t,t))^\wedge)$, $\dot{x}_t \leftarrow \frac{\Delta t}{t} B_\theta^v(t,t)$ See (Eq. (54)).

Set $\dot{t} \leftarrow \Delta t$ in the small-t parameterisation (otherwise $\dot{t} \leftarrow 1$), and $\dot{s} \leftarrow 0$.

Compute a single JVP of $f_\theta(R_t, x_t, t, s)$ along $(\dot{R}_t, \dot{x}_t, \dot{t}, \dot{s})$ to obtain primals (B_θ^A, B_θ^v) and directional derivatives $\big(\Delta t\, \frac{d}{dt} B_\theta^A, \Delta t\, \frac{d}{dt} B_\theta^v\big)$ (Eq. (58))

Form the scaled derivative targets via Eq. (57).

$\mathcal{J} \leftarrow \mathrm{sg}\big(J(A_\theta^{s\to t})\big)$

if $\texttt{jmode} = J$ **then**

  $\mathrm{res}_{\mathrm{rot}} \leftarrow \mathcal{J}\big(t\, \frac{d}{dt} A_\theta^{s\to t}\big) - t\, \omega_t$

else

  $\mathrm{res}_{\mathrm{rot}} \leftarrow t\, \frac{d}{dt} A_\theta^{s\to t} - \mathcal{J}^{-1}\big(t\, \omega_t\big)$ ▷ Remark 6

end if

$\mathrm{res}_{\mathrm{trans}} \leftarrow t\, \frac{d}{dt} v_\theta^{s\to t} - t\, v_t$ ▷ $J \equiv I$ on the abelian branch

Normalise at the end by $\max(t, t_\varepsilon)^{-2}$ (Eq. (62))

directional derivatives $(t-s)\, \frac{d}{dt} B_\theta^A$ and $(t-s)\, \frac{d}{dt} B_\theta^v$ as

$$\mathcal{L}_{\mathrm{rot}} := \frac{1}{\max(t, t_\varepsilon)^2} \sum_{i=1}^{N} \begin{cases} \Big\| \mathrm{sg}\Big(J\Big(A_{\theta,i}^{s\to t}\Big)\Big)\, \Big(t\, \frac{d}{dt} A_{\theta,i}^{s\to t}\Big) - t\, \omega_{t,i} \Big\|_2^2, & \texttt{jmode} = J, \\ \Big\| t\, \frac{d}{dt} A_{\theta,i}^{s\to t} - \mathrm{sg}\Big(J^{-1}\Big(A_{\theta,i}^{s\to t}\Big)\Big)\, t\, \omega_{t,i} \Big\|_2^2, & \texttt{jmode} = J^{-1}, \end{cases} \tag{61}$$

$$\mathcal{L}_{\mathrm{trans}} := \frac{1}{\max(t, t_\varepsilon)^2} \sum_{i=1}^{N} \big\| t\, v_{t,i} - t\, \tfrac{d}{dt} v_{\theta,i}^{s\to t} \big\|_2^2 . \tag{62}$$

This is algebraically equivalent to the unscaled MeanFlow objectives but avoids explicit $1/t$ factors during target construction; the only division by t is the final $\max(t, t_\varepsilon)^{-2}$ normalization applied after squaring the residuals. The two `jmode` branches share the same JVP (58) and scaled targets (57), differing only in the final residual.

I.2 Rescaled MeanFlow: training and inference pseudocode

The rescaled MeanFlow training objective and the corresponding stable inference procedure are summarised in Algorithm 3 and Algorithm 4, respectively. In inference, in addition to the standard linear rotation schedule, we also use an exponential (EXP) rotation schedule inspired by ReQFlow Yue et al. [2025]. The main idea is to redefine the effective rotation step sizes so that steps are larger on the noise side ($t \approx 1$) and become smaller and more fine-grained near the data side ($t \approx 0$), which empirically improves few-step generation stability.

Algorithm 4: Stable inference with $t_{\min} > 0$ and endpoint prediction (linear and exponential rotation schedules)

Require: Prior sample (R_1^N, x_1^N); network f_θ; steps T; $t_{\min} > 0$; rotation schedule $\in \{\text{LINEAR}, \text{EXP}\}$, exp rate $c > 0$
Ensure: $(\hat{R}_0^N, \hat{x}_0^N)$

$\{t_i\}_{i=0}^{T-1} \leftarrow \text{linspace}(1, t_{\min}, T)$
$(R, x) \leftarrow (R_1^N, x_1^N)$
for $i = 0, \dots, T-2$ **do**
  $t \leftarrow t_i, \quad s \leftarrow t_{i+1}, \quad \Delta t \leftarrow t - s$
  $(\hat{R}_0^\theta, \hat{x}_0^\theta) \leftarrow f_\theta(s, t, R, x)$
  $u_\theta \leftarrow \log\big((\hat{R}_0^\theta)^\top R\big)^\vee$ ▷ log-map from R toward endpoint $\hat{R}_0^\theta$
  if schedule = LINEAR **then**
    $A_\theta^{\text{avg}} \leftarrow \frac{1}{t}\, u_\theta$ ▷ rate $1/t$ (remaining time)
  else ▷ EXP
    $A_\theta^{\text{avg}} \leftarrow c\, u_\theta$ ▷ constant rate c
  end if
  $v_\theta^{\text{avg}} \leftarrow \frac{1}{t}(x - \hat{x}_0^\theta)$ ▷ translation: linear in both schedules
  $R \leftarrow R \exp\big(-\Delta t\, (A_\theta^{\text{avg}})^\wedge\big)$
  $x \leftarrow x - \Delta t\, v_\theta^{\text{avg}}$
end for
$(\hat{R}_0^\theta, \hat{x}_0^\theta) \leftarrow f_\theta(0, t_{min}, R, x)$
$(\hat{R}_0^N, \hat{x}_0^N) \leftarrow (\hat{R}_0^\theta, \hat{x}_0^\theta)$

J Stable Training Framework, Part 2: JVP-Free Training via SE(3) α-Flow

The stabilisation in Section I mitigates, but does not remove, the core difficulty of the *differential* consistency target: it still requires the trajectory derivative $\frac{d}{dt} A_\theta^{\text{avg}}$ via a Jacobian–vector product (JVP) of f_θ. Since $A_\theta^{\text{avg}} = \frac{1}{t}\log((\hat{R}_0^\theta)^\top R_t)^\vee$ already carries a $1/t$ factor, differentiating it injects a second $1/t$, so a head error ε on the rotation output propagates to the target at order ε/t^2. This is benign for the flat translation branch but dominates the instability of the rotation branch, where it compounds with SO(3) curvature and fragile forward-mode autodiff through the IPA trunk. We adopt the α-Flow framework of Zhang et al. [2025], which replaces the differential target by a two-evaluation, JVP-free consistency target, reducing the amplification to $\mathcal{O}(\varepsilon/t)$.

J.1 Model parameterization: endpoint and average-velocity heads

The network f_θ, evaluated at input (s, t, R_t, x_t), may be read either as an *endpoint* (x-)predictor returning $(\hat{R}_0^\theta, \hat{x}_0^\theta)$, or as an *average-velocity* (u-)predictor returning $(A_\theta^{\text{avg}}, v_\theta^{\text{avg}})$ — the mean body angular velocity and mean translational velocity of the predicted geodesic from the endpoint to (R_t, x_t). The two views are equivalent and can be transferred to each other:

$$\text{(rotation)} \quad A_\theta^{\text{avg}} = \tfrac{1}{t}\log\big((\hat{R}_0^\theta)^\top R_t\big)^\vee \iff \hat{R}_0^\theta = R_t \exp\big(-t\, A_\theta^{\text{avg}\wedge}\big), \tag{63}$$

$$\text{(translation)} \quad v_\theta^{\text{avg}} = \tfrac{1}{t}\big(x_t - \hat{x}_0^\theta\big) \iff \hat{x}_0^\theta = x_t - t\, v_\theta^{\text{avg}}. \tag{64}$$

Both maps depend on (s, t, R_t, x_t); we suppress the arguments where clear, and write $A_\theta^{\text{avg}}(s, t, R_t, x_t)$, $\hat{R}_0^\theta(R_t, x_t, s, t)$ etc. when needed. We develop the α-Flow targets and losses in the average-velocity

variables $(A_\theta^{\text{avg}}, v_\theta^{\text{avg}})$, where the construction is most transparent (the *regular-t* form), and switch to the endpoint variables only to expose and cancel the small-denominator factors, giving the numerically stabilised *small-t* form. Throughout, the data-side instantaneous quantities are the body angular velocity $\omega_t = \Omega(t, R_t)^\vee$ and the translational velocity $v_t = v(t, x_t) = x_1 - x_0$.

J.2 Time grid

Let $\alpha \in [\alpha_{\min}, 1]$ be the consistency-step ratio, floored by a small $\alpha_{\min} > 0$. With $0 \le s \le t \le 1$, define the intermediate time and step

$$m = \alpha s + (1-\alpha)t, \qquad \delta = t - m = \alpha(t-s), \qquad m - s = (1-\alpha)(t-s), \qquad s \le m \le t,$$

so that $(m-s) + \delta = t - s$. The stepped-back (intermediate) state, obtained by integrating the shift velocities backward over $[m, t]$, is

$$x_m = x_t - \delta\, \tilde{v}_t, \qquad R_m = R_t \exp(-\delta\, \tilde{\omega}_t^\wedge),$$

with the shift velocities $\tilde{v}_t, \tilde{\omega}_t$ fixed below.

J.3 Translation target

Following Zhang et al. [2025], the average velocity over $[s, t]$ decomposes across the split point m into a *near* segment $[m, t]$ and a *far* segment $[s, m]$. The near segment uses the shift velocity

$$\tilde{v}_t = \begin{cases} v_t, & \text{(data velocity; flow-matching / MeanFlow choice),} \\ u_\theta^x(m, t, x_t) = v_\theta^{\text{avg}}(m, t, x_t), & \text{(model prediction; Shortcut choice),} \end{cases}$$

and we adopt the data velocity $\tilde{v}_t = v_t$. The far segment uses the stop-gradient model average velocity at the stepped-back state $x_m = x_t - \delta \tilde{v}_t$,

$$v_\theta^{\text{avg}}(s, m, x_m) = \tfrac{1}{m}\big(x_m - \hat{x}_0^\theta(R_m, x_m, s, m)\big),$$

and the α-Flow target is the convex combination

$$v_{\text{tgt}}^{\text{avg}} = \alpha\, \tilde{v}_t + (1-\alpha)\, v_\theta^{\text{avg}}(s, m, x_m). \tag{65}$$

Regular-t loss. Regressing the model average velocity directly,

$$\mathcal{L}_{\text{trans}}^{\text{reg}} = \frac{1}{\alpha} \sum_{i=1}^{N} \big\| v_{\theta,i}^{\text{avg}}(s, t, x_t) - \text{sg}\big(v_{\text{tgt},i}^{\text{avg}}\big) \big\|_2^2. \tag{66}$$

The $1/\alpha$ prefactor matches the $\alpha \to 0$ scaling of the residual (Prop. 8); no further s, t-dependent weight is needed because the abelian target is a plain convex combination.

Small-t (stabilised) form. The average-velocity variables carry an explicit $1/t$ (via $v^{\text{avg}} = \frac{1}{t}(x_t - \hat{x}_0)$) and $1/m$ (in the far term), which amplify head error as $t, m \to 0$. Deferring the $1/t$ into the displacement variable $B_\theta^x = t\, v_\theta^{\text{avg}} = x_t - \hat{x}_0^\theta$ removes the model-side division, and the corresponding displacement target is

$$B_{\text{tgt}}^x = t\, v_{\text{tgt}}^{\text{avg}} = \alpha t\, \tilde{v}_t + \frac{(1-\alpha)t}{m}\big(x_m - \hat{x}_0^\theta(R_m, x_m, s, m)\big),$$

where $0 < \frac{(1-\alpha)t}{m} \leq 1$ is numerically bounded. Regressing on displacements with a floored normaliser gives

$$\mathcal{L}_{\text{trans}} = \frac{1}{\alpha \, \max(t, t_\varepsilon)^2} \sum_{i=1}^{N} \left\| B^x_{\theta,i} - \text{sg}(B^x_{\text{tgt},i}) \right\|_2^2. \tag{67}$$

Since $\|v^{\text{avg}}_\theta - v^{\text{avg}}_{\text{tgt}}\|_2^2 = t^{-2}\|B^x_\theta - B^x_{\text{tgt}}\|_2^2$, (67) coincides with (66) for $t \geq t_\varepsilon$; the floor only regularises the vanishing-t limit.

J.4 Rotation target

We mirror the translation branch on SO(3), replacing vector addition by group composition. For $0 \leq a \leq b \leq 1$, the accumulated relative rotation is

$$D(a,b) := \mathcal{T} \exp\Big(\int_a^b \Omega(\tau)\, d\tau \Big) = \exp\big((b-a)\, A^{\text{avg}}(a,b)^\wedge\big) = R_a^\top R_b \in \text{SO}(3), \tag{68}$$

where the middle equality collapses the time-ordered exponential to a single generator and is exact under the geodesic (constant-body-velocity) assumption, which holds for the data path and for the x-prediction geodesic to $\hat{R}^\theta_0$.

Proposition 7 (Interval additivity)**.** *For any $s \leq m \leq t$, $D(s,t) = D(s,m)\, D(m,t)$.*

Proof. $D(s,t) = R_s^\top R_t = (R_s^\top R_m)(R_m^\top R_t) = D(s,m)D(m,t)$. The regrouping is exact (independent of the geodesic assumption) and is the non-commutative SO(3) analogue of Euclidean displacement additivity: addition becomes group multiplication, ordered far ($[s,m]$) before near ($[m,t]$). □

Mirroring (65), the near segment $[m,t]$ uses the shift angular velocity

$$\tilde{\omega}_t = \begin{cases} \omega_t = \Omega(t, R_t)^\vee, & \text{(data angular velocity)}, \\ A^{\text{avg}}_\theta(m, t, R_t, x_t), & \text{(model prediction; Shortcut choice)}, \end{cases}$$

and we adopt the data angular velocity $\tilde{\omega}_t = \omega_t$. The far segment $[s,m]$ uses the stop-gradient model average angular velocity at the stepped-back state (R_m, x_m), $R_m = R_t \exp(-\delta\, \tilde{\omega}_t^\wedge)$:

$$A_m := A^{\text{avg}}_\theta(s, m, R_m, x_m) = \tfrac{1}{m} \log\big((\hat{R}^\theta_0(R_m, x_m, s, m))^\top R_m\big)^\vee.$$

Then $D(s,m) = \exp\big((m-s)A_m^\wedge\big)$ and $D(m,t) = \exp\big(\delta\, \tilde{\omega}_t^\wedge\big) = R_m^\top R_t$, with $x_m = x_t - \delta \tilde{v}_t$ as in Section J.3.

Regular-t target and loss. By Proposition 7 and (68), $\log D(s,t)^\vee = (t-s)\, A^{\text{avg}}_{\text{tgt}}$, so the average-velocity target is the group-composition analogue of (65):

$$A^{\text{avg}}_{\text{tgt}} = \frac{1}{t-s} \log\Big(\underbrace{\exp\big((m-s)A_m^\wedge\big)}_{D(s,m)} \; \underbrace{\exp\big(\delta\, \tilde{\omega}_t^\wedge\big)}_{D(m,t)} \Big)^\vee. \tag{69}$$

The loss mirrors (66):

$$\mathcal{L}^{\text{reg}}_{\text{rot}} = \frac{1}{\alpha} \sum_{i=1}^{N} \left\| A^{\text{avg}}_{\theta,i}(s,t) - \text{sg}\big(A^{\text{avg}}_{\text{tgt},i}\big) \right\|_2^2. \tag{70}$$

The scalar $\frac{1}{t-s}$ must remain *outside* $\log(\exp\cdot\exp)$: because A_m and $\tilde{\omega}_t$ do not commute, folding it into the two exponentials would rescale each segment angle and alter the Baker–Campbell–Hausdorff (BCH) cross term, no longer yielding $\log D(s,t)$. This is the non-commutative counterpart of the translation branch, where the same normalization is instead absorbed into the linear convex weights α, $1-\alpha$.

Small-t (stabilised) form. Two small-denominator factors appear: the overall $1/t$ shared with translation, and the rotation-specific $1/(t-s)$ together with $\log(\cdot)$ near I. Deferring the $1/t$ into the displacement $B^A_\theta = t\,A^{\text{avg}}_\theta = \log((\hat{R}^\theta_0)^\top R_t)^\vee$ gives the displacement target $B^A_{\text{tgt}} = t\,A^{\text{avg}}_{\text{tgt}}$,

$$\boxed{B^A_{\text{tgt}} = \frac{t}{t-s}\,\log\Big(\underbrace{\exp\big((\tfrac{m-s}{m}B^A_m)^\wedge\big)}_{D(s,m)}\;\underbrace{\exp\big(\delta\,\tilde{\omega}_t^\wedge\big)}_{D(m,t)}\Big)^\vee, \qquad B^A_m = \log\big((\hat{R}^\theta_0(R_m,x_m,s,m))^\top R_m\big)^\vee,} \tag{71}$$

using $(m-s)A_m = \frac{m-s}{m}B^A_m$. Both segment angles scale as $O(t-s)$ and are bounded: $\frac{m-s}{m}B^A_m$ has $\frac{m-s}{m} \le 1$ and main $\log \le \pi$, while $\delta\,\tilde{\omega}_t = \alpha(t-s)\,\omega_t$ with $\|\omega_t\| \le \pi$. Hence $\|\log D(s,t)^\vee\| = O(t-s)$, the $\frac{1}{t-s}$ factor is cancelled at the same rate, and

$$\big\|B^A_{\text{tgt}}\big\| \le \pi\Big(\tfrac{(1-\alpha)t}{m} + \alpha t\Big) \le 2\pi,$$

mirroring the bounded weights $\frac{(1-\alpha)t}{m}$, $\alpha t \le 1$ of the translation target: the target is analytically free of small-denominator blow-up. For numerical stability when $t-s$ is below machine tolerance t_ε (where $\log(\cdot)$ near I loses precision), we avoid forming $\frac{1}{t-s}$ and use the first-order BCH limit, whose $O(t-s)$ correction is then negligible:

$$B^A_{\text{tgt}} = \begin{cases} \dfrac{t}{t-s}\,\log\big(D(s,m)\,D(m,t)\big)^\vee, & t-s \ge t_\varepsilon, \\ t\big[(1-\alpha)A_m + \alpha\,\tilde{\omega}_t\big], & t-s < t_\varepsilon. \end{cases}$$

The rotation loss is then identical in form to (67),

$$\mathcal{L}_{\text{rot}} = \frac{1}{\alpha\,\max(t,t_\varepsilon)^2}\sum_{i=1}^{N}\big\|B^A_{\theta,i} - \text{sg}(B^A_{\text{tgt},i})\big\|_2^2, \tag{72}$$

and, via $\|A^{\text{avg}}_\theta - A^{\text{avg}}_{\text{tgt}}\|_2^2 = t^{-2}\|B^A_\theta - B^A_{\text{tgt}}\|_2^2$, coincides with (70) whenever $t \ge t_\varepsilon$ and $t-s \ge t_\varepsilon$. The target requires one logarithm and two exponentials and no differentiation of f_θ.

Remark 8 (Fallback at $\alpha = 1$ and choice of normalization). At $\alpha = 1$ the intermediate time hits the far end, $m = s$ and $\delta = t-s$, so the far segment vanishes ($D(s,m) = I$) and $B^A_{\text{tgt}} = \frac{t}{t-s}\log D(m,t)^\vee = t\,\tilde{\omega}_t$: the objective reduces to flow matching (25), anchoring the branch to the data velocity and supplying the boundary condition that prevents collapse. The opposite limit $\alpha \to 0$ recovers the differential MeanFlow objective and is analysed in Section J.5.

The group composition $\log(\exp\cdot\exp)$ is essential throughout: the naive Lie sum $(m-s)A_m + \delta\,\tilde{\omega}_t$ drops the BCH/Jacobian correction and is not equivalent.

J.5 The $\alpha \to 0$ Limit Recovers MeanFlow

In this subsection we show that the α-Flow loss converges to the differential MeanFlow loss as $\alpha \to 0$. We treat the rotation branch; the translation branch is the abelian case ($J \equiv I$, no BCH correction) and reduces to Euclidean MeanFlow (3) verbatim.

Remark 9 (Alphaflow normalization). If α-Flow is to interpolate exactly between flow matching and MeanFlow, the prefactor should be $1/\alpha^2$ rather than $1/\alpha$, since the residual regressed in (70) is $O(\alpha)$ (Proposition 8). We therefore analyse

$$\widetilde{\mathcal{L}}_{\text{rot}} := \frac{1}{\alpha^2} \sum_{i=1}^{N} \left\| A^{\text{avg}}_{\theta,i}(s,t,R_t,x_t) - \text{sg}\big(A^{\text{avg}}_{\text{tgt},i}\big) \right\|_2^2, \tag{73}$$

with $A^{\text{avg}}_{\text{tgt}}$ from (69); the $1/\alpha$ prefactor of (70) follows the style of Zhang et al. [2025] and is what we train with. We also use only the first case of (71), the second being a small-$(t-s)$ fallback introduced for numerical convenience.

Proposition 8 ($\alpha \to 0$ recovers the MeanFlow loss)**.** *Let $g(\tau) := (\tau - s)A^{\text{avg}}_\theta(s,\tau,R_\tau,x_\tau)$ denote the model log-displacement along the interpolation path, so that $g(t) = (t-s)A^{\text{avg}}_\theta(s,t)$ and, by the product rule, $\dot g(t) = A^{\text{avg}}_\theta + (t-s)\frac{d}{dt}A^{\text{avg}}_\theta$ is the bracketed quantity of* (8)*. If g is C^2 on t, then the target* (69) *expands as*

$$A^{\text{avg}}_{\text{tgt}} = A^{\text{avg}}_\theta(s,t) + \alpha\Big(J\big(g(t)\big)^{-1}\omega_t - \dot g(t)\Big) + O(\alpha^2) \tag{74}$$

and consequently, per residue,

$$\widetilde{\mathcal{L}}_{\text{rot}} = \big\|\dot g(t) - J^{-1}\omega_t\big\|_2^2 + O(\alpha), \tag{75}$$

i.e. exactly the residual of the J^{-1} form (12) *of the MeanFlow identity.*

Proof. Since (73) decouples across residues, we take $N = 1$ and drop the index i. Write $m = t - \delta$ with $\delta = \alpha(t-s)$, and $J := J(g(t))$.

Since the data velocities ω_t and v_t are constant along the interpolation path (Appendix D), the stepped-back state satisfies $R_t \exp(-\delta\,\omega_t^\wedge) = R_m$ and $x_t - \delta v_t = x_m$ *exactly*. The two factors of (69) are therefore $D(s,m) = \exp(g(m)^\wedge)$ and $D(m,t) = \exp(\delta\,\omega_t^\wedge)$, with g the same function appearing in the statement; this is what lets the finite difference below capture the *total* derivative (9) rather than ∂_t alone.

The first-order BCH formula gives $\log(e^X e^Y) = X + \frac{\text{ad}_X}{1-e^{-\text{ad}_X}}Y + O(\|Y\|^2)$. On $\mathfrak{so}(3)$ one has $\text{ad}_{\phi^\wedge}\psi^\wedge = (\phi^\wedge\psi)^\wedge$, so in vector coordinates $\big(\frac{1-e^{-\text{ad}_X}}{\text{ad}_X}\big)^\vee = \int_0^1 e^{-u\phi^\wedge}du = J(\phi)$ by (10). This is invertible: $J(\phi)$ is a polynomial in the skew matrix $\phi^\wedge$, so its eigenvalues are 1 and $\frac{1-e^{\mp i\theta}}{\pm i\theta}$ with $\theta = \|\phi\| \le \pi$, all nonzero. Taking $X = g(m)^\wedge$ and $Y = \delta\,\omega_t^\wedge = O(\alpha)$,

$$\log\big(D(s,m)D(m,t)\big)^\vee = g(m) + \delta\, J\big(g(m)\big)^{-1}\omega_t + O(\delta^2).$$

Substituting $g(m) = g(t) - \delta\dot{g}(t) + O(\delta^2)$ and $J(g(m))^{-1} = J^{-1} + O(\delta)$, we have:

$$\begin{aligned}
A_{\text{tgt}}^{\text{avg}} &= \frac{1}{t-s}\log\big(D(s,m)\,D(m,t)\big)^{\vee} \\
&= \frac{1}{t-s}\Big(g(m) + \delta\, J\big(g(m)\big)^{-1}\omega_t + O(\delta^2)\Big) \\
&= \frac{1}{t-s}\Big(g(t) - \delta\,\dot{g}(t) + \delta\, J^{-1}\omega_t + O(\delta^2)\Big) \\
&= \frac{g(t)}{t-s} + \frac{\alpha(t-s)}{t-s}\Big(J^{-1}\omega_t - \dot{g}(t)\Big) + \frac{O\big(\alpha^2(t-s)^2\big)}{t-s} \\
&= A_{\theta}^{\text{avg}}(s,t) - \alpha\Big(\dot{g}(t) - J^{-1}\omega_t\Big) + O(\alpha^2),
\end{aligned}$$

which is (74). Substituting into (73),

$$\begin{aligned}
\widetilde{\mathcal{L}}_{\text{rot}} &= \frac{1}{\alpha^2}\Big\|A_{\theta}^{\text{avg}}(s,t) - \text{sg}\big(A_{\text{tgt}}^{\text{avg}}\big)\Big\|_2^2 \\
&= \frac{1}{\alpha^2}\Big\|A_{\theta}^{\text{avg}}(s,t) - \Big(A_{\theta}^{\text{avg}}(s,t) - \alpha\left(\dot{g}(t) - J^{-1}\omega_t\right) + O(\alpha^2)\Big)\Big\|_2^2 \\
&= \frac{1}{\alpha^2}\Big\|\alpha\left(\dot{g}(t) - J^{-1}\omega_t\right) + O(\alpha^2)\Big\|_2^2 \\
&= \big\|\big(\dot{g}(t) - J^{-1}\omega_t\big)\big\|_2^2 + O(\alpha),
\end{aligned}$$

which is (75). □

Remark 10 (Relation to the two loss forms). The limit (75) is the J^{-1} form (12), which is exactly the objective used in Stage 2 (Appendix M): the α-Flow warm-up and the MeanFlow phase therefore optimise the same target up to $O(\alpha)$, rather than switching objectives mid-training. The J form (11) shares its zero set but differs by the reweighting $J^{\top}J$, which is the identity on the translation branch and on the diagonal $s = t$.

We verify Proposition 8 numerically. We take a smooth generator field $g(\tau)$ and a data velocity ω_t drawn *independently* of g, so that $J(g(t))^{-1}\omega_t \neq \dot{g}(t)$ in general and the first-order term of (74) is non-degenerate. For a range of α we form the α-Flow rotation target $A_{\text{tgt}}^{\text{avg}}(\alpha) = \frac{1}{t-s}\text{Log}\big(e^{g(m)^{\wedge}}e^{\delta\,\omega_t^{\wedge}}\big)^{\vee}$ with $m = \alpha s + (1-\alpha)t$ and $\delta = \alpha(t-s)$, and compare it to the zeroth- and first-order predictions of (74), computing $\dot{g}$ by forward-mode AD and $J(g(t))$ from its closed form. The $\mathcal{O}(\alpha)/\mathcal{O}(\alpha^2)$ scalings (Figure 8) confirm the expansion and its coefficient; consistently, $\widetilde{\mathcal{L}}_{\text{rot}}$ converges to $\|\dot{g} - J(g(t))^{-1}\omega_t\|_2^2$.

K Stable Training Framework, Part 3: Semigroup Loss of SE(3)-MeanFlow

The MeanFlow identity (8) is the *differential* form of our average velocity: it is obtained by differentiating (30) in t, and its training target carries both a Jacobian–vector product (for $\frac{d}{dt}A_{\theta}^{\text{avg}}$) and the right Jacobian J. In this section, we show that the *same* definition (30) also admits a *finite*, derivative-free characterization: the average-velocity flow map is a genuine two-parameter semigroup on SE(3), and the consistency with this semigroup yields a JVP-free, J-free training objective. Throughout, let $s \le m \le t \in [0,1]$. We write

$$\omega_t := \Omega(t, R_t, x_t)^{\vee} = \text{Log}(R_0^{\top}R_1)^{\vee}, \qquad v_t := v(t, x_t) = x_1 - x_0$$

for the data-side instantaneous (constant body-frame) velocities used in flow matching (25).

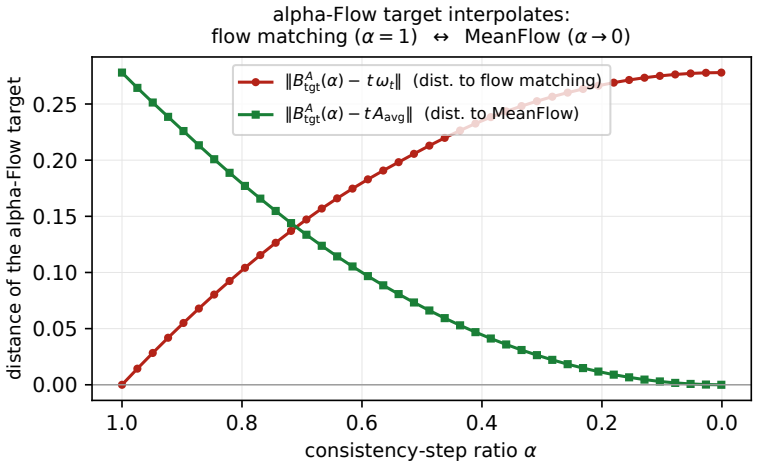

Figure 8: The α-Flow target interpolates between flow matching and MeanFlow. On a smooth generator field $g(\tau)$ we form the rotation target $B^A_{\text{tgt}}(\alpha) = tA^{\text{avg}}_{\text{tgt}}$ of (71) over a range of α and plot its distance to the two endpoints: to the flow-matching target $t\,\omega_t$ (red), which vanishes at $\alpha = 1$ (Remark 8), and to the MeanFlow target $t\,A^{\text{avg}}$ (green), which vanishes as $\alpha \to 0$ (Proposition 8). The plot is drawn in the displacement variable $B^A = t\,A^{\text{avg}}$ of Section J.4; the common factor t affects both curves equally.

K.1 The average-velocity flow map

Recall from (30) that the average angular velocity over $[s, t]$ is defined by

$$\exp\big((t-s)\,\Omega^{\text{avg}}(s,t,R_t,x_t)\big) = \mathcal{T}\exp\Big(\int_s^t \Omega(\tau,R_\tau,x_\tau)\,d\tau\Big) =: D(s,t) \in \text{SO}(3), \tag{76}$$

where the last equality $D(s,t) = R_s^\top R_t$ holds by integrating the left-trivialized ODE $\dot{R}_\tau = R_\tau\Omega(\tau, R_\tau)$; this is exactly (68). We write $A^{\text{avg}} = (\Omega^{\text{avg}})^\vee$, and define the rotation log-displacement by

$$A^{s\to t} := (t-s)A^{\text{avg}}(s,t,R_t,x_t) \in \mathbb{R}^3, \quad \text{thus} \quad \exp\big((A^{s\to t})^\wedge\big) = D(s,t).$$

For the translation branch on $\mathbb{R}^3$, the decoupled product metric (24) gives the displacement $p(s,t) := x_t - x_s = (t-s)\,v^{\text{avg}}(s,t,x_t)$. Based on the two notations of displacement above, we give the following definition.

Definition 1 (Average-velocity flow map)**.** For $s \le t$, we define $\Psi_{s\to t} : \text{SE}(3) \to \text{SE}(3)$ by

$$\Psi_{s\to t}(R,x) := \Big(R\,D(s,t),\quad x + p(s,t)\Big) = \Big(R\,\exp\big((A^{s\to t})^\wedge\big),\quad x + (t-s)\,v^{\text{avg}}(s,t)\Big).$$

By construction, $\Psi_{s\to t}(R_s,x_s) = (R_t,x_t)$ along the interpolation path.

K.2 The semigroup property

Having defined the average-velocity flow map, we next show that it forms a semigroup under time composition.

Algorithm 5: JVP-free SE(3) α-Flow training step

Require: (R_t^N, x_t^N); times $s \le t$; data (Ω_t^N, v_t^N); head f_θ; $\alpha \in [\alpha_{\min}, 1]$; clamp t_ε

$m \leftarrow \alpha s + (1-\alpha)t, \quad \delta \leftarrow t - m, \quad \omega_t := \omega_t^N \leftarrow (\Omega_t^N)^\vee$

$R_m^N \leftarrow R_t^N \exp(-\delta \omega_t^\wedge), \quad x_m^N \leftarrow x_t^N - \delta\, v_t^N$

$(\hat{R}_0^m, \hat{x}_0^m) \leftarrow \mathrm{sg}\, f_\theta(R_m^N, x_m^N, s, m)$

$mA_m \leftarrow \log((\hat{R}_0^m)^\top R_m^N)^\vee, \quad mu_m \leftarrow x_m^N - \hat{x}_0^m$ ▷ displacements

$B_{\mathrm{tgt}}^A \leftarrow \frac{t}{t-s} \log\big(\exp((\frac{m-s}{m}\, mA_m)^\wedge) \exp(\delta\, \omega_t^\wedge)\big)^\vee$

$B_{\mathrm{tgt}}^x \leftarrow \alpha t\, v_t^N + \frac{(1-\alpha)t}{m}\, mu_m$

$(\hat{R}_0^\theta, \hat{x}_0^\theta) \leftarrow f_\theta(R_t^N, x_t^N, s, t)$ ▷ only graph path

$B_\theta^A \leftarrow \log((\hat{R}_0^\theta)^\top R_t^N)^\vee, \quad B_\theta^x \leftarrow x_t^N - \hat{x}_0^\theta$

Form $\mathcal{L}_{\mathrm{rot}}, \mathcal{L}_{\mathrm{trans}}$ (Eqs. (72),(67));

Proposition 9 (Semigroup property of the average-velocity flow)**.** *For all $s \le m \le t$, the family $\{\Psi_{s\to t}\}_{0\le s\le t\le 1}$ defined in Definition 1 satisfies*

$$\Psi_{m\to t} \circ \Psi_{s\to m} = \Psi_{s\to t}, \qquad \Psi_{t\to t} = \mathrm{id}. \tag{77}$$

Equivalently, in terms of the average velocities,

$$\textbf{\textit{(rotation)}} \quad \exp\big((A^{s\to m})^\wedge\big)\, \exp\big((A^{m\to t})^\wedge\big) = \exp\big((A^{s\to t})^\wedge\big), \tag{78}$$

$$\textbf{\textit{(translation)}} \quad (m-s)\, v^{\mathrm{avg}}(s,m) + (t-m)\, v^{\mathrm{avg}}(m,t) = (t-s)\, v^{\mathrm{avg}}(s,t). \tag{79}$$

Proof. On SO(3), the transition operators compose by inserting $R_m R_m^\top = I$:

$$D(s,t) = R_s^\top R_t = (R_s^\top R_m)(R_m^\top R_t) = D(s,m)\, D(m,t),$$

which is Proposition 7. (Equivalently, this is the multiplicativity of the time-ordered exponential $\mathcal{T}\exp(\int_s^t) = \mathcal{T}\exp(\int_s^m)\, \mathcal{T}\exp(\int_m^t)$, so the statement holds for an arbitrary velocity field, not only for the geodesic path.) Substituting $D(a,b) = \exp((A^{a\to b})^\wedge)$ gives (78). On the flat translation factor, displacements add, $x_t - x_s = (x_t - x_m) + (x_m - x_s)$, which is (79). Combining the two factors and using the product law (24) gives $\Psi_{m\to t}(\Psi_{s\to m}(R,x)) = (R\, D(s,m)D(m,t),\, x + p(s,m) + p(m,t)) = (R\, D(s,t),\, x + p(s,t)) = \Psi_{s\to t}(R,x)$, i.e. (77). □

Remark 11 (Where SO(3) curvature enters). The composition (78) is *exact* and independent of curvature: it is the associativity of group multiplication. Curvature appears only if one tries to collapse the product of exponentials into a *single* Lie-algebra sum. By the Baker–Campbell–Hausdorff (BCH) formula,

$$A^{s\to t} = \log\Big(\exp\big((A^{s\to m})^\wedge\big) \exp\big((A^{m\to t})^\wedge\big)\Big)^\vee = A^{s\to m} + A^{m\to t} + \tfrac{1}{2}\, A^{s\to m} \times A^{m\to t} + \cdots,$$

so the naive additive law $A^{s\to t} = A^{s\to m} + A^{m\to t}$ holds only when the two axes are collinear ($A^{s\to m} \times A^{m\to t} = 0$); in general it drops the BCH cross terms. The translation identity (79) has no such correction because $\mathbb{R}^3$ is abelian. This is the single place where the rotation branch departs from the Euclidean MeanFlow semigroup.

K.3 The semigroup-consistency loss

Proposition 9 characterizes the correct average-velocity field *without* any time derivative: the field is consistent iff its one-step prediction on $[s,t]$ equals the two-step composition through any intermediate $m \in [s,t]$. We turn this into a regression objective. Let f_θ output $A_\theta^{\mathrm{avg}}(\cdot)$ and $v_\theta^{\mathrm{avg}}(\cdot)$ (in the endpoint parameterization of Section I, $A_\theta^{\mathrm{avg}}(s,t,R_t) = \frac{1}{t}\log((\hat{R}_0^\theta)^\top R_t)^\vee$). Given a triple $s < m < t$, form the intermediate state by stepping back the near segment $[m,t]$,

$$R_m = R_t \exp\big(-(t-m)A_\theta^{\mathrm{avg}}(m,t,R_t,x_t)^\wedge\big), \qquad x_m = x_t - (t-m)\, v_\theta^{\mathrm{avg}}(m,t,x_t), \tag{80}$$

and define the composed (two-step) targets by (78)–(79):

$$A_{\mathrm{tgt}}^{s\to t} := \log\Big(\underbrace{\exp\big((m-s)A_\theta^{\mathrm{avg}}(s,m,R_m,x_m)^\wedge\big)}_{D(s,m)}\ \underbrace{\exp\big((t-m)A_\theta^{\mathrm{avg}}(m,t,R_t,x_t)^\wedge\big)}_{D(m,t)}\Big)^\vee, \tag{81}$$

$$p_{\mathrm{tgt}}^{s\to t} := (m-s)\, v_\theta^{\mathrm{avg}}(s,m,x_m) + (t-m)\, v_\theta^{\mathrm{avg}}(m,t,x_t). \tag{82}$$

The model regresses its *direct* one-step prediction on $[s,t]$ onto these stop-gradient targets:

$$\mathcal{L}_{\mathrm{rot}}^{\mathrm{sg}} = \mathbb{E}_{s<m<t}\Big\|(t-s)A_\theta^{\mathrm{avg}}(s,t,R_t,x_t) - \mathrm{sg}\big(A_{\mathrm{tgt}}^{s\to t}\big)\Big\|_2^2, \tag{83}$$

$$\mathcal{L}_{\mathrm{trans}}^{\mathrm{sg}} = \mathbb{E}_{s<m<t}\Big\|(t-s)v_\theta^{\mathrm{avg}}(s,t,x_t) - \mathrm{sg}\big(p_{\mathrm{tgt}}^{s\to t}\big)\Big\|_2^2. \tag{84}$$

The semigroup constraint alone admits trivial (collapsed) minimizers; it must be anchored by the $s \to t$ boundary, where (76) degenerates to the instantaneous velocity. This boundary term is exactly flow matching:

$$\mathcal{L}_{\mathrm{rot}}^{\mathrm{bd}} = \mathbb{E}_t\big\|A_\theta^{\mathrm{avg}}(t,t,R_t,x_t) - \omega_t\big\|_2^2, \qquad \mathcal{L}_{\mathrm{trans}}^{\mathrm{bd}} = \mathbb{E}_t\big\|v_\theta^{\mathrm{avg}}(t,t,x_t) - v_t\big\|_2^2. \tag{85}$$

The total objective is

$$\mathcal{L}^{\text{sg-MF}} = \underbrace{\mathcal{L}_{\mathrm{rot}}^{\mathrm{bd}} + \mathcal{L}_{\mathrm{trans}}^{\mathrm{bd}}}_{\text{flow matching (anchor)}} + \underbrace{\mathcal{L}_{\mathrm{rot}}^{\mathrm{sg}} + \mathcal{L}_{\mathrm{trans}}^{\mathrm{sg}}}_{\text{semigroup consistency}}. \tag{86}$$

Every term uses only forward evaluations of f_θ together with $\exp$ / $\log$ on $\mathrm{SO}(3)$ and addition on $\mathbb{R}^3$: there is no Jacobian–vector product and no explicit right Jacobian J.

Remark 12 (The normalization scalar must stay outside $\log(\exp\cdot\exp)$). If one prefers to regress the average velocity $A_\theta^{\mathrm{avg}}(s,t)$ itself rather than the log-displacement, the target is $A_{\mathrm{tgt}}^{\mathrm{avg}}(s,t) = \frac{1}{t-s}A_{\mathrm{tgt}}^{s\to t}$ with $A_{\mathrm{tgt}}^{s\to t}$ from (81). The scalar $\frac{1}{t-s}$ must remain *outside* $\log(\exp\cdot\exp)$: because the two segment generators do not commute, absorbing it into the two exponentials would rescale each segment angle and corrupt the BCH cross term, no longer producing $\log D(s,t)$. On the translation branch the same normalization is harmlessly absorbed into the linear weights. This is the finite-interval counterpart of the observation made for (71).

K.4 Consistency: zero loss implies exact reconstruction

The boundary term fixes the instantaneous limit and the semigroup term propagates it to all intervals; together they pin down the unique correct flow map. The following is the finite (JVP-free) analogue of Proposition 5.

Proposition 10 (Uniqueness of the semigroup minimizer)**.** *Suppose A_θ^{avg} is continuous and, along the interpolation path, satisfies the boundary condition $A_\theta^{\mathrm{avg}}(t,t,R_t,x_t)=\omega_t$ for all t together with the rotation semigroup identity*

$$\exp\big((A_\theta^{s\to m})^\wedge\big)\,\exp\big((A_\theta^{m\to t})^\wedge\big)=\exp\big((A_\theta^{s\to t})^\wedge\big),\qquad \forall\, s\le m\le t,$$

where $A_\theta^{a\to b}:=(b-a)A_\theta^{\mathrm{avg}}(a,b,R_b,x_b)$. Then $\exp((A_\theta^{s\to t})^\wedge)=R_s^\top R_t$ for all $s\le t$; in particular, with $s=0,t=1$, $\hat R_0:=R_1\exp(-(A_\theta^{0\to 1})^\wedge)=R_0$. The analogous statement holds for translation with $v_\theta^{\mathrm{avg}}(t,t)=v_t$.

Proof. Fix t and set $G(s):=\exp((A_\theta^{s\to t})^\wedge)\in\mathrm{SO}(3)$, so $G(t)=I$. For $h>0$ the semigroup identity gives $G(s)=\exp((A_\theta^{s\to s+h})^\wedge)\,G(s+h)$, hence

$$G(s+h)=\exp\big(-(A_\theta^{s\to s+h})^\wedge\big)\,G(s).$$

By the boundary condition and continuity, $A_\theta^{s\to s+h}=h\,A_\theta^{\mathrm{avg}}(s,s+h)=h\,\omega_s+o(h)$, so $\exp(-(A_\theta^{s\to s+h})^\wedge)=I-h\,\omega_s^\wedge+o(h)$ and therefore

$$\frac{d}{ds}G(s)=-\,\omega_s^\wedge\,G(s),\qquad G(t)=I.$$

On the other hand $D(s,t)=R_s^\top R_t$ obeys, using $\dot R_s=R_s\omega_s^\wedge$ and skew-symmetry,

$$\frac{d}{ds}D(s,t)=\big(\partial_s R_s^\top\big)R_t=-\,\omega_s^\wedge R_s^\top R_t=-\,\omega_s^\wedge D(s,t),\qquad D(t,t)=I.$$

G and $D(\cdot,t)$ solve the same linear ODE with the same terminal condition, so by uniqueness $G(s)=D(s,t)=R_s^\top R_t$ for all $s\le t$. Taking $s=0,t=1$ gives $\exp((A_\theta^{0\to 1})^\wedge)=R_0^\top R_1$, i.e. $\hat R_0=R_0$. For translation, the additive cocycle $p_\theta(s,t)$ with $p_\theta(t,t)$-derivative $v_\theta^{\mathrm{avg}}(t,t)=v_t$ integrates to $p_\theta(s,t)=x_t-x_s$, giving $\hat x_0=x_0$. □

Proposition 10 shows (86) is a valid training objective: its global minimizer is the exact average-velocity field, and the two ingredients are both necessary—the boundary term supplies the instantaneous data velocity, and the semigroup term is the (curvature-exact) propagation rule that extends it to every interval.

K.5 Relation to the differential identity and to α-Flow

Remark 13 (Equivalence with the right-Jacobian identity). The semigroup loss (86) and the differential MeanFlow loss (41) have the same minimizer but realize the right Jacobian differently. Differentiating the boundary-anchored semigroup identity (78) in t at $s\to t$ reproduces the differential identity (8), in which J appears explicitly as the Fréchet derivative of exp. In the finite form, J never appears as a separate factor: its entire effect is absorbed into the BCH series of $\log(\exp\cdot\exp)$ in (81). Concretely, a first-order BCH expansion of (81) reinstates J and recovers (41) in gradient (cf. Remark 8). The two formulations are thus gradient-equivalent to first order, but the finite form is JVP-free and, because all segment angles are bounded by π, avoids the $\mathcal{O}(\varepsilon/t^2)$ amplification of the differential target discussed in Section J.

Remark 14 (α-Flow as an instance). The JVP-free α-Flow objective of Section J is a particular instantiation of (86). Choosing the split point $m=\alpha s+(1-\alpha)t$, replacing the near segment $[m,t]$ model prediction by the *data* velocity $\tilde\omega_t=\omega_t$ (so that $D(m,t)=\exp(\delta\,\omega_t^\wedge)$ with $\delta=\alpha(t-s)$), and keeping the far segment as the stop-gradient model evaluation, turns (81) into the α-Flow

target (71) and (82) into its translation counterpart. The limit $\alpha = 1$ makes the far segment vanish and reduces (86) to the flow-matching boundary (85); the limit $\alpha \to 0$ recovers, via BCH, the right-Jacobian differential loss (Remark 8). The "pure" semigroup form (81)–(82), using the model on both segments, corresponds to the Shortcut choice $\tilde{\omega}_t = u_\theta^A(m, t, R_t)$.

Connection to Riemannian MeanFlow semigroup. The semigroup objective (86) can be viewed as a variant of the semigroup formulation proposed in Riemannian MeanFlow Woo et al. [2026]: their construction is based on the endpoint (geodesic) loss, while ours is derived from and anchored by the flow-matching boundary condition (85). We provide code for this semigroup objective in our repository; however, in our experiments we found that with a limited training budget (e.g., 10–20k steps) semigroup training yields weaker few-step improvements than our JVP-based MeanFlow loss, and therefore we do not use the semigroup loss in our final training recipe. We include it here for theoretical completeness.

L Quaternion Formulation of SE(3)-MeanFlow

The SE(3)-MeanFlow objectives admit an equivalent formulation with unit quaternions on $\mathbb{S}^3$, following the rotation parametrisation of ReQFlow Yue et al. [2025]. We derive it here for completeness; an implementation is provided in our repository alongside the rotation-matrix formulation used throughout the paper.

L.1 Quaternion algebra and conventions

We follow the standard conventions; see Hanson [2005] for a fuller treatment. A quaternion is a pair

$$q = (q_0, \mathbf{q}) \in \mathbb{R} \times \mathbb{R}^3, \qquad \|q\| = 1,$$

with multiplication and inverse

$$q \otimes p = \big(q_0 p_0 - \mathbf{q}^\top \mathbf{p},\ q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p}\big), \qquad q^{-1} = (q_0, -\mathbf{q}).$$

The unit quaternions form the manifold $\mathbb{S}^3$, a double cover of SO(3): the quaternions q and $-q$ represent the same rotation

$$R(q) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}, \qquad q = (w, x, y, z).$$

Recovering q from R is standard (trace-based) up to this sign; we fix it by enforcing $w \geq 0$.

Exponential and logarithm. We define $\mathrm{Exp} : \mathbb{R}^3 \to \mathbb{S}^3$ with the half-angle absorbed,

$$\mathrm{Exp}(\omega) := \Big(\cos \tfrac{\|\omega\|}{2},\ \sin \tfrac{\|\omega\|}{2}\ \tfrac{\omega}{\|\omega\|}\Big), \qquad \mathrm{Exp}(0) = (1, \mathbf{0}), \tag{87}$$

and let Log be its inverse on the hemisphere $w \geq 0$, so $\|\mathrm{Log}(q)\| \leq \pi$. With this convention the covering map is compatible with the matrix exponential,

$$R\big(\mathrm{Exp}(\omega)\big) = \exp(\omega^\wedge), \tag{88}$$

so ω carries the same axis-angle meaning as in the rest of the paper. (The ReQFlow subsection above writes the same object as $\exp(\frac{1}{2}\omega)$; the factor of two is bookkeeping in where the half-angle is placed.) The tangent space is $T_q\mathbb{S}^3 = \{v \in \mathbb{R}^4 : q^\top v = 0\}$.

Kinematics. Let $\omega(t) \in \mathbb{R}^3$ be the *body-frame* angular velocity, i.e. the same quantity as in (1). The left-trivialised kinematics are

$$\dot{q}_t = \tfrac{1}{2}\, q_t \otimes (0, \omega(t)), \tag{89}$$

whose solution is the time-ordered exponential in $\mathbb{S}^3$,

$$q_t = q_s \otimes \mathcal{T}\mathrm{Exp}\Big(\int_s^t \omega(\tau)\, d\tau\Big), \qquad \text{and} \qquad q_t = q_s \otimes \mathrm{Exp}\big((t-s)\,\omega\big) \ \text{ if } \omega \text{ is constant.}$$

The half-angle in (87) absorbs the factor $\frac{1}{2}$ of (89), so no stray factors of two appear below.

Interpolation and data velocity. Mirroring Appendix D, for a data rotation q_0 and a prior rotation q_1 (sign-aligned so that $q_0^\top q_1 \geq 0$, which selects the shorter arc) the conditional path and its constant body angular velocity are

$$q_t = q_0 \otimes \mathrm{Exp}\big(t\,\omega_t\big), \qquad \omega_t := \mathrm{Log}\big(q_0^{-1} \otimes q_1\big) \in \mathbb{R}^3. \tag{90}$$

L.2 Average velocity and the MeanFlow identity

Exactly as in (30), we define the average angular velocity over $[s,t]$ through the time-ordered exponential. Writing the state as $z_t = (x_t, q_t) \in \mathbb{R}^3 \times \mathbb{S}^3$ to make the conditioning explicit,

$$\mathrm{Exp}\big((t-s)\,\omega^{\mathrm{avg}}(s,t,x_t,q_t)\big) := \mathcal{T}\mathrm{Exp}\Big(\int_s^t \omega(\tau, q_\tau, x_\tau)\, d\tau\Big) = q_s^{-1} \otimes q_t. \tag{91}$$

Proposition 11 (Quaternion MeanFlow identity)**.** *Differentiating* (91) *with respect to* t *yields*

$$J\big((t-s)\,\omega^{\mathrm{avg}}\big)\Big(\omega^{\mathrm{avg}}(s,t,x_t,q_t) + (t-s)\tfrac{d}{dt}\omega^{\mathrm{avg}}(s,t,x_t,q_t)\Big) = \omega_t, \tag{92}$$

with J *the* same *right Jacobian* (10) *as in the rotation-matrix formulation, and*

$$\frac{d}{dt}\omega^{\mathrm{avg}} = \frac{\partial}{\partial t}\omega^{\mathrm{avg}} + \big\langle \nabla_q \omega^{\mathrm{avg}}, \dot{q}_t \big\rangle + \big\langle \nabla_x \omega^{\mathrm{avg}}, \dot{x}_t \big\rangle.$$

Proof. The covering map $\mathbb{S}^3 \to \mathrm{SO}(3)$ of (88) is a two-to-one Lie group homomorphism and a local diffeomorphism, and under the identification $(0,\omega) \leftrightarrow \omega^\wedge$ it induces the identity on Lie algebras ($\mathfrak{su}(2) \cong \mathfrak{so}(3)$). Applying it to (91) returns (30) verbatim, so every step of the proofs of Propositions 3 and 4 carries over unchanged. □

Remark 15 (Quaternions do not remove the right Jacobian). Since the two Lie algebras are isomorphic, the quaternion identity (92) is *identical* to the rotation-matrix identity (8), right Jacobian included. Changing the representation therefore cannot simplify the MeanFlow target; it can only change the numerical conditioning of Exp, Log and the Jacobian–vector product used to evaluate it.

L.3 Parametrisation, loss and inference

As in Section I, the network is an endpoint predictor returning $(\hat{q}_0^\theta, \hat{x}_0^\theta)$, converted to the average-velocity head by

$$\omega_\theta^{\mathrm{avg}} = \tfrac{1}{t}\,\mathrm{Log}\big((\hat{q}_0^\theta)^{-1} \otimes q_t\big) \iff \hat{q}_0^\theta = q_t \otimes \mathrm{Exp}\big(-t\,\omega_\theta^{\mathrm{avg}}\big).$$

Regressing the left-hand side of (92) onto the data velocity ω_t of (90) gives the rotation loss

$$\mathcal{L}_{\text{rot}}^{\mathbb{S}^3} = \mathbb{E}_{s<t,\, q(z_0,z_1)}\Big[\big\| J\big((t-s)\omega_\theta^{\text{avg}}\big)\Big(\omega_\theta^{\text{avg}} + (t-s)\,\text{sg}\big(\tfrac{d}{dt}\omega_\theta^{\text{avg}}\big)\Big) - \omega_t \big\|_2^2\Big], \tag{93}$$

the exact analogue of (11); the translation branch and the α-Flow variant of Section J are unchanged, since neither touches the rotation representation. Inference steps backwards by $q_s \leftarrow q_t \otimes \text{Exp}\big(-(t-s)\,\omega_\theta^{\text{avg}}\big)$.

L.4 Practical note

Empirically the quaternion implementation consistently underperformed the rotation-matrix one, and we use the latter for every result in this paper. Two causes are consistent with what we observed. First, the surrounding pipeline (dataset, IPA trunk, auxiliary losses, evaluation) operates on rotation matrices, so the quaternion path inserts repeated matrix $\leftrightarrow$ quaternion conversions whose error accumulates. Second, and more specific to MeanFlow, the double cover $q \sim -q$ means the enforced sign convention $w \geq 0$ can flip along a trajectory; the endpoint prediction $\hat{q}_0^\theta$ then jumps discontinuously, and the forward-mode derivative $\frac{d}{dt}\omega_\theta^{\text{avg}}$ in (93) is corrupted at exactly those steps. Rotation matrices have no such ambiguity. Together with Remark 15—the quaternion target is not analytically simpler—this left no reason to prefer the quaternion branch, and we report no quaternion-based results.

M Training Curriculum: Pre-training and Post-training

Our final model is obtained in three phases: a two-stage *pre-training* that builds a strong multi-step backbone generator, followed by a *post-training* (rectification) stage that sharpens few-step generation. All phases share the same network (Section N) and the same decoupled $\text{SE}(3) = \text{SO}(3) \times \mathbb{R}^3$ objective; they differ only in which consistency target is used and in how the data–noise pairs are coupled.

M.1 Stage 1: JVP-free α-Flow warm-up

Training the endpoint-parameterized MeanFlow target directly from initialization is fragile: as discussed in Sections I and J, the differential rotation target amplifies head error as $\mathcal{O}(\varepsilon/t^2)$ and requires a forward-mode derivative (JVP) through the IPA trunk. We therefore warm up with the JVP-free α-Flow objective of Section J, annealing the consistency-step ratio α from a cap $\alpha_{\max} = 1$ toward a floor $\alpha_{\min} = 0.1$ along a logistic schedule:

$$\alpha(k) = \begin{cases} \alpha_{\max}, & k \leq k_s, \\ \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})\,\sigma_\gamma(\tau_k), & k_s < k < k_e, \\ \alpha_{\min}, & k \geq k_e, \end{cases} \qquad \sigma_\gamma(\tau) = \frac{1}{1 + e^{\,\gamma(\tau - \frac{1}{2})}}, \quad \tau_k = \frac{k - k_s}{k_e - k_s}, \tag{94}$$

where k is the optimizer step and $\tau_k \in [0,1]$ is the normalized progress through the anneal window. Here k_s is the hold phase—the ratio is pinned at the cap for the first k_s steps—k_e is the step by which the floor is reached, and γ sets the steepness of the logistic transition, centered at the window midpoint $k_s + \frac{1}{2}(k_e - k_s)$. We use $k_s = 2000$, $k_e = 150$k, and $\gamma = 8$.

By Remark 8, $\alpha = 1$ reduces the objective exactly to flow matching (25), so during the hold phase training is anchored to the data velocity and cannot collapse; decreasing α thereafter progressively injects the few-step average-velocity consistency that underlies fast sampling. The anneal was scheduled over 150k steps, but we stopped Stage 1 early at 100k—where $\alpha \approx 0.29$—because validation

designability had already plateaued; α therefore never reaches its floor $\alpha_{\min} = 0.1$ during pre-training. This checkpoint initializes Stage 2. Full settings are in Table 7.

Self-conditioning. In all phases we use 50% self-conditioning, following FrameFlow Yim et al. [2023a]/ReQFlow Yue et al. [2025]. With probability 0.5 a detached preliminary forward pass produces an endpoint prediction $\hat{x}_0^\theta$ whose pairwise C_α distogram is appended as an extra edge feature to a second, gradient-carrying forward pass; the remaining half of each batch passes zeros in that slot. At inference the previous step's endpoint prediction is used as the self-conditioning input (Algorithm 4). This improves sample quality at no additional inference cost.

Global-OT coupling. Both pre-training stages instantiate the mini-batch OT coupling of Appendix D.3, but solve a single plan over the *entire* global batch rather than one plan per rank. At each step a `torch.distributed.all_gather` assembles all $M = \sum_{r=1}^{W} B_r$ backbones across the W ranks, and the independent data–noise pairing is replaced by the assignment

$$\pi^\star = \arg\min_{\pi \in \mathcal{S}_M} \sum_{k=1}^{M} c\big(z_0^k,\, z_1^{\pi(k)}\big), \tag{95}$$

with c the decoupled $\mathrm{SE}(3)^N$ transport cost of Appendix D.3 and $(\lambda_R, \lambda_x) = (0.5, 0.5)$. The exact Hungarian assignment is solved on rank 0 with POT Flamary et al. [2021] and $\pi^\star$ broadcast back; the $O(M^3)$ solve is negligible against a forward/backward pass at these batch sizes. Solving (95) once over all M backbones uses the full pairing information, whereas W per-rank plans search a block-diagonal subset of $\mathcal{S}_M$ and under-use it by a factor W; empirically the global coupling roughly halves the early-plateau time on SCOPe. Since mini-batch OT is a biased estimate of the true coupling whose bias decreases with batch size, the global plan yields a straighter induced path than W independent per-rank plans at the same per-GPU memory cost.

Post-training disables it (its self-reflow pairs are already matched; Section M.3).

M.2 Stage 2: endpoint + MeanFlow objective

Starting from the Stage-1 checkpoint, we continue with the endpoint-anchored small-t MeanFlow objective for a maximum budget of 10k steps, and select the released checkpoint at 6.1k steps by validation designability. We retain the 50% self-conditioning and the global-OT coupling of Stage 1; the only changes are the switch to the differential MeanFlow loss and the added endpoint anchor. The loss combines three terms. The endpoint distance is defiend as:

$$\mathcal{L}_{\text{end}} := \sum_{i=1}^{N} \Big[D^2_{\mathrm{SO}(3)}\big(R^i_{\theta,0},\, R^i_0\big) + \big\|x^i_{\theta,0} - x^i_0\big\|_2^2 \Big], \qquad D_{\mathrm{SO}(3)}(R, Q) := \big\| \operatorname{Log}(R^\top Q)^\vee \big\|_2 = \arccos\Big(\tfrac{\mathrm{tr}(R^\top Q)-1}{2}\Big), \tag{96}$$

Remark 16. We emphasize that the split of the training budget between the α-Flow warm-up and the MeanFlow stage is a design choice rather than a requirement: the two can be freely re-balanced (e.g. a shorter warm-up with a longer MeanFlow phase). Because protein-scale training is expensive, we do not sweep this ratio at the protein scale; on the low-dimensional $\mathrm{SO}(3)^2$ benchmark (Appendix H) different warm-up/MeanFlow ratios reach the few-step regime with little difference, so we simply report the single configuration used for the released checkpoint.

Term	Raw	Weight	Weighted
MeanFlow rot ($\mathcal{L}_{\text{rot}}$)	3.87	$\times 0.05$	0.193
MeanFlow trans ($\mathcal{L}_{\text{trans}}$)	2.63	$\times 0.05$	0.131
MeanFlow total	**6.50**		**0.324**
Endpoint rot	0.136	$\times 1.0$	0.136
Endpoint trans	0.060	$\times 1.0$	0.060
Endpoint total	**0.196**		**0.196**

Table 6: Per-term loss magnitudes at $\lambda_{\text{MF}} = 0.05$ (median over 24 SCOPe training batches; `our_loss_norm` disabled, right-Jacobian inverse form). The raw MeanFlow residual is $\sim 33\times$ larger than the endpoint term—owing to the near-data reweighting $1/\max(t, t_\varepsilon)^2$—and $\lambda_{\text{MF}} = 0.05$ brings the two to the same order (≈ 1.6:1).

Loss normalization and stability. Our optional loss-magnitude normalization (`our_loss_norm`) is *disabled* in all phases. For stability we instead rely on three mechanisms: a global gradient-norm clip at 1.0, per-residue rotation- and translation-loss clamps (50 and 5), and the endpoint near-data reweighting $1/\max(t, t_\varepsilon)^2$ ($t_\varepsilon = 0.1$, Stage 2 onward). The rotation and translation branches are weighted by $(w_R, w_x) = (1.0, 1.0)$.

Total objective. The Stage-2 loss is the weighted sum

$$\lambda_{\text{end}}\,\mathcal{L}_{\text{end}} + \lambda_{\text{MF}}\left(\mathcal{L}^{\text{MF}}_{\text{rot}} + \mathcal{L}^{\text{MF}}_{\text{trans}}\right) + \lambda_{\text{aux}}\,\mathbf{1}[t < 0.75]\,\mathcal{L}_{\text{aux}}, \tag{97}$$

with $(\lambda_{\text{end}}, \lambda_{\text{MF}}, \lambda_{\text{aux}}) = (1.0,\ 0.05,\ 2.0)$; the endpoint term anchors the model and stabilizes the MeanFlow loss, and all remaining settings are listed in Table 7. Here $\lambda_{\text{MF}} = 0.05$ is a scale normalizer rather than a tuned trade-off. Because the loss-magnitude normalization is disabled in all phases (see above), the terms of (97) enter at their raw magnitudes, and these differ by more than an order of magnitude: the small-t MeanFlow residuals (62) carry the near-data reweighting $1/\max(t, t_\varepsilon)^2$, which reaches $t_\varepsilon^{-2} = 100$ for $t \le t_\varepsilon = 0.1$, whereas the endpoint term is unweighted. With $\lambda_{\text{MF}} = 0.05$ the weighted MeanFlow contribution is comparable in magnitude to $\lambda_{\text{end}}\mathcal{L}_{\text{end}}$ throughout training—a ratio of roughly 1.6:1 (Table 6). The small numerical value therefore reflects the scale of the raw residual, not a small role in the objective; equivalently, one may enable the loss normalization and set $\lambda_{\text{MF}} = 1$.

M.3 Post-training: self-reflow rectification

Few-step designability is further improved by a rectification stage that replaces the random data–noise coupling with the model's own deterministic transport, following the rectified-flow strategy of ReQFlow Yue et al. [2025].

Self-reflow dataset. Using the Stage-2 model, for each residue length $N \in [60, 128]$ we draw prior samples $z_1 \sim p_{\text{prior}}$ and integrate the model to obtain coupled pairs $(z_1, \hat{z}_0)$ (noise $\to$ generated backbone). Each generated backbone is scored with the self-consistency pipeline (ProteinMPNN followed by ESMFold), and we retain only *designable* samples (scRMSD < 2Å). We keep 50 designable pairs per length—over-sampling 100 generations per length to meet the quota—yielding $69 \times 50 = 3450$ coupled pairs over the SCOPe length range.

Hyperparameter	Stage 1 (α-Flow)	Stage 2 and Post-train (endpt+MF)
Objective	α-Flow (JVP-free)	endpoint + MeanFlow
Initialization	from scratch	Stage-1 ckpt, Stage-2 ckpt
Maximum Steps	150k	10k, 10k
α schedule (cap→floor)	$1 \to 0.1$	—
(k_s, k_e, γ)	$(2\text{k}, 150\text{k}, 8)$	—
Loss weight	(alpha,aux)=1,1	(Endpoint,MF,aux)=1,0.05,2
Meanflow config	—	(version,jmode)=(mf,J^{-1})
Auxiliary time gate	$t < 0.75$	$t < 0.75$
loss normalization	off	off
Rotation-loss clamp	50	50
Trans-loss clamp	5	5
Min. time $t_{\min}$	10^{-6}	10^{-6}
Denom. clamp t_ε	0.1	0.1
Rotation schedule	exp (rate 10)	exp (rate 10)
$(w_R,\, w_x)$	$(1.0, 1.0)$	$(1.0, 1.0)$
$s \mid t$ sampler	$\mathcal{U}[t_{\min}, t]$	$\mathcal{U}[t_{\min}, t]$
$s{=}t$ anchor fraction	0	0
Two-time input ($s < t$)	yes	yes
Self-conditioning	yes (50%)	yes (50%)
JVP	— (JVP-free)	single combined
Coupling	global SE(3) OT	global SE(3) OT
Optimizer, lr	Adam, 10^{-4}	Adam, 10^{-4}
Gradient clip	0 (none)	1.0
Gradient accumulation	2	2
Batch cap ($N^2_{\max}$)	$\propto 1/N^2$ ($5{\times}10^5$)	$\propto 1/N^2$ ($5{\times}10^5$)
Devices	4× H100 (80GB) GPU	4× H100 (80GB) GPU

Table 7: Training configuration for the two pre-training stages. Post-training (self-reflow) reuses the Stage-2 column verbatim, changing only the data coupling: the OT coupling is replaced by the model's own deterministic self-reflow pairs, with OT disabled (Section M.3).

Rectification objective. We then continue training from the Stage-2 checkpoint on these pairs using the *identical* Stage-2 objective (97), but with the independent coupling $q(z_0, z_1)$ replaced by the model-induced deterministic coupling $(z_1, \hat{z}_0)$ and with the OT re-coupling disabled (the pairs are already transport-consistent). Because the target transport is now (approximately) straight, the average-velocity field it must match is closer to constant along each trajectory, which is precisely the regime in which few-step MeanFlow sampling is exact. Checkpoints are saved densely and selected on validation designability and secondary-structure content.

M.4 Checkpoint selection

Rather than selecting the final model by a grid search over training hyper-parameters, we retain the training checkpoint according to a fixed, pre-specified rule on validation-time generation quality. During training we periodically sample unconditional backbones at two inference budgets, $T{=}100$ and $T{=}20$ integration steps, and monitor (i) the Cα–Cα bond-geometry validity (ca_ca) and (ii) the secondary-structure composition (helix and strand fractions). We keep the checkpoint that (a) attains ca_ca > 0.97 at *both* $T{=}100$ and $T{=}20$, so that backbones remain geometrically valid

	Ours	ReQFlow	RMF
node embed size	256	256	768
edge embed size	128	128	384
c_{hidden}	128	128	48
# attention heads	8	8	16
# query/key points	8	8	—
# value points	12	12	16
# IPA blocks	6	6	16
seq. tfmr (heads/layers)	4 / 2	4 / 2	12 / 3
per-block AdaLN gate	**yes**	no	yes
total parameters	∼16.8M	∼16.8M	∼437M

Table 8: Trunk configuration. "Ours" is the ReQFlow Yue et al. [2025] trunk with the only architectural addition being a per-block AdaLN-Zero gate (+0.05M, 0.33% of parameters). Both are ∼26× smaller than RMF Woo et al. [2026] while recovering most of the structural fidelity.

in both the high- and low-step regimes; (b) keeps the strand fraction near or above 0.2, matching the reference distribution and guarding against strand collapse; and (c) among the checkpoints satisfying (a)–(b), maximizes the helix fraction. This criterion was fixed before final evaluation and applied identically across all runs.

N Architecture

Our network is the ReQFlow Yue et al. [2025] trunk left *unchanged*, augmented with only two additions required to turn a single-time flow-matching model into a two-time MeanFlow model: (i) a shared two-time embedding for the MeanFlow inputs (t, s) (§N.3), and (ii) a per-block, zero-initialised AdaLN-Zero gate on each block's rigid update (§N.2). The trunk itself—the invariant point attention (IPA) stack of Jumper et al. [2021] as instantiated by ReQFlow, comprising per block an IPA module, a two-layer sequence transformer, node/edge transitions and a backbone frame update—is not modified. The full model has ∼16.8M parameters, of which the AdaLN additions account for only 0.33% (∼55k); it is therefore ∼26× smaller than RMF (Woo et al. 2026, 437M) while remaining checkpoint-compatible with ReQFlow/QFlow (Table 8).

N.1 Backbone trunk

The trunk follows the ReQFlow configuration: node and edge embedding sizes 256 and 128, IPA hidden width $c_{\text{hidden}} = 128$ with 8 attention heads, 8 query/key points and 12 value points, a 4-head / 2-layer sequence transformer, and 6 IPA blocks. The input embedder is ReQFlow's: a self-conditioned pairwise distogram (22 bins), a relative-position encoding (relpos_$k = 64$), and a diffuse-mask embedding. Because the trunk and embedder are inherited unchanged, our model loads directly from our pretrained checkpoint (in stage 2).

N.2 Block-wise AdaLN-Zero gate

The sole structural change to the trunk is a per-block gate on the rigid update. Each IPA block emits a six-vector backbone update (three rotational, three translational) that is composed onto the

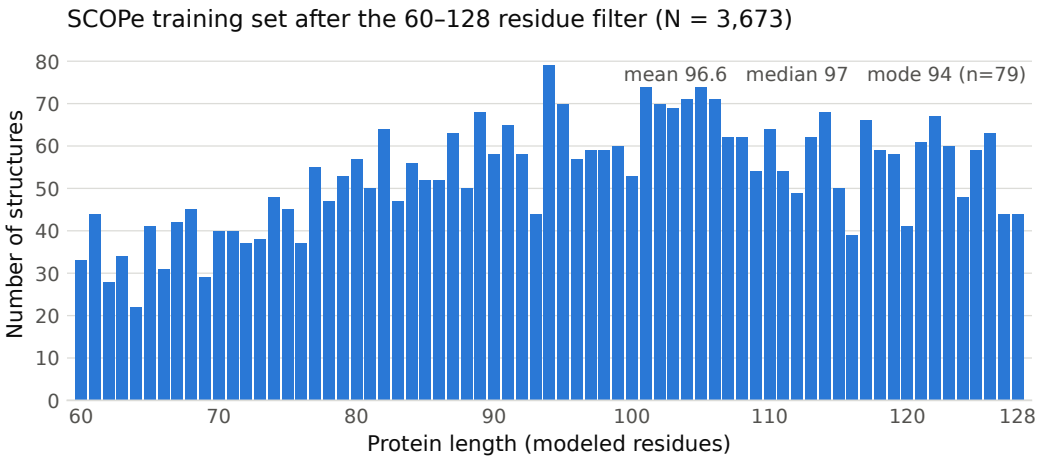

Figure 9: Residue-length distribution of the SCOPe backbone dataset used in our experiments (3,673 backbones with lengths in $[60, 128]$).

running frame. We modulate this update with an AdaLN-Zero Peebles and Xie [2023] scale,

$$\text{update}_b \;\leftarrow\; \big(1 + \gamma_b(\mathbf{c}_b)\big) \odot \text{update}_b,$$

where the gate γ_b is produced by a per-block linear head from the conditioning vector $\mathbf{c}_b = [\,\mathbf{h}_b \parallel \mathbf{c}^{(t)}\,]$, formed by concatenating (i) the block's current *node features* $\mathbf{h}_b$—which are SE(3)-invariant and state-aware (they depend on the current (R_t, x_t) through IPA)—and (ii) a joint time/interval embedding $\mathbf{c}^{(t)}$ built from t and $h = t - s$. Every gate head is zero-initialised, so at initialisation $\gamma_b \equiv 0$, the update is scaled by 1, and the model is *forward-identical* to the unmodified ReQFlow trunk. This lets the two-time MeanFlow conditioning enter each block's geometry update per residue and per step, without perturbing the pretrained trunk at the start of training. In implementation the gate is attached via a forward hook, so the trunk's forward code is untouched.

N.3 Shared two-time embedding

A MeanFlow prediction is conditioned on the pair (t, s) with $0 \le s \le t \le 1$; we let $h = t - s$. Both t and h are passed through a *single* shared sinusoidal-plus-MLP embedder $\phi(\cdot)$ and concatenated in a fixed order, $\mathbf{c}^{(t)} = [\phi(t) \,\|\, \phi(h)]$. Concatenation preserves the (t, h) ordering; the same embedding feeds both the node/edge input features and the per-block AdaLN gate of §N.2. This is the only place the single-time ReQFlow trunk is made time-pair aware.

O Extra Results/Settings in the Protein Experiment

We provide additional visualizations in this section. Figure 10 reports the joint distribution of per-sample helix and strand content (via mdtraj DSSP) for the SCOPe-generated backbones of each method, at T=100 (top) and T=20 (bottom). Each panel is a 2-D histogram over the helix-fraction (x) and strand-fraction (y) plane, sharing a common colour scale; the anti-diagonal reflects the intrinsic helix–strand trade-off within a single chain. Across all methods the mass concentrates along this trade-off with an additional helix-rich mode, consistent with the SCOPe length regime. Our models (SE3MF and RecSE3MF) recover a distribution comparable to the flow-matching baselines rather than collapsing to a single motif. Importantly, the distribution is largely preserved when

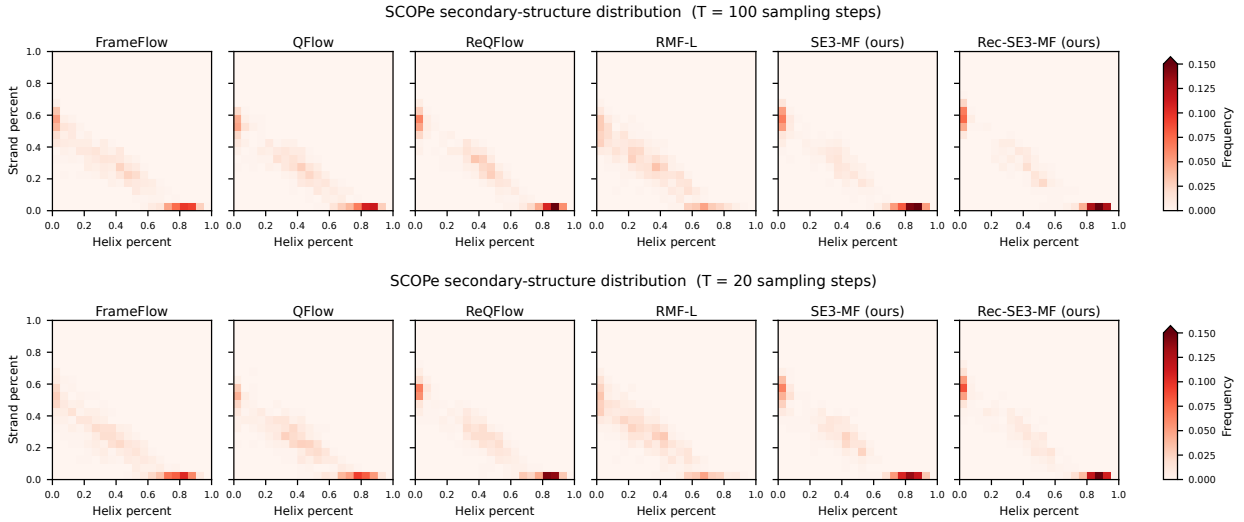

Figure 10: Secondary-structure statistics (top: T=100, bottom: T=20).

the sampling budget is reduced from T=100 to T=20: the few-step regime does not visibly distort the secondary-structure statistics, indicating that the average-velocity consistency learned during training transfers to aggressive step reduction without a mode shift.

Visualization of SCOPe. Figure 9 shows the residue-length distribution of the SCOPe backbones used in our experiments.

Random seed in evaluation. All evaluations use a fixed random seed for exact reproducibility: for each sampled backbone the RNG seed is set deterministically to $12345 + 10^5\,T + 10^3\,N + i$, where T is the number of sampling steps, N the chain length, and i the sample index within that length.

How large is the Jacobian correction on the training distribution? Figure 5 shows that dropping the Jacobian breaks the identity by $\mathcal{O}(t-s)$ on an analytic path. To check that this regime is actually visited during training, we evaluate $\rho = \|(J^{-1}(A_\theta^{s\to t}) - I)\,\omega_t\|_2 / \|\omega_t\|_2$—the relative change the Jacobian makes to the regression target of (12)—over 1.2×10^5 residue-level samples from held-out SCOPe backbones under the Stage-2 time sampler (SE(3)-OT coupling, Table 9). The correction grows monotonically with the interval: median ρ is 0.8% for $t - s < 0.1$, 7.2% for $t - s \in [0.25, 0.5)$, and 28% for $t - s > 0.75$ (P90: 2.6%, 19%, 58%). Since the sampler draws s across the full range below t, 22% of training targets fall in the regime where the correction exceeds 10%. The right Jacobian is therefore not a negligible term on the distribution the objective is trained over, even though the intervals queried at inference ($t - s = 1/T$) are short enough that $J^{-1} \approx I$ there; the log-map approximation $J^{-1} := I$ used by RMF-PT Zhong et al. [2026] discards a correction of this size during training.

O.1 Training budget: steps versus epochs

We report the training budget in optimizer steps, not epochs: the QFlow/ReQFlow loader (from FrameFlow) oversamples each epoch to ≈ 460 steps, while ours does one pass at ≈ 35 steps—a $\sim$13$\times$ gap on the identical SCOPe set that makes epoch counts incomparable. Importantly, both QFlow/ReQFlow and our method use the same hardware (4$\times$80GB GPUs) and the same batch

$t-s$	n	ρ		ϕ (rad)	
		median	P90	median	P90
$[0, 0.1)$	39,044	0.008	0.026	0.061	0.175
$[0.1, 0.25)$	32,212	0.031	0.077	0.283	0.500
$[0.25, 0.5)$	29,502	0.072	0.187	0.589	0.993
$[0.5, 0.75)$	14,957	0.165	0.392	0.891	1.497
$[0.75, 1.0]$	4,320	0.280	0.581	0.841	1.582
overall	120,035	0.033	0.194	0.289	0.955

Table 9: The inverse right Jacobian's relative change to the rotation target, $\rho = \|(J^{-1}(A_\theta^{s\to t}) - I)\,\omega_t\|_2 / \|\omega_t\|_2$, and the driving angle $\phi = \|A_\theta^{s\to t}\|_2$ (rad), binned by the interval $t-s$ over 1.2×10^5 residue-level samples from held-out SCOPe backbones under the Stage-2 (SE(3)-OT) sampler.

cap (see $N^2_{\max}$ in Table 7), so each optimizer step processes the same amount of data; the only difference is that under our loader each sample is seen once per epoch, whereas QFlow/ReQFlow repeats samples multiple times due to oversampling (which depends on the number of GPUs). Steps are directly comparable; as a check, the QFlow checkpoint's 90,160 steps equal its reported 195 epochs [Yue et al., 2025].

O.2 Ablation Study: A Controlled Comparison of Consistency Objectives

The RMF rows of Table 1 come from the released checkpoint, which uses a different trunk (437M versus our 16.8M) and a much larger budget (598k versus our Stage-2 budget), so that comparison does not isolate the objective. Table 10 removes the confound: all three rows start from the *same* Stage-1 α-Flow checkpoint and share the trunk, data, coupling, self-conditioning and optimizer, so the only variable is the Stage-2 consistency target.

Under this matched budget the semigroup target is a weaker self-consistency signal than the differential MeanFlow target. The FM+semigroup variant tracks plain flow matching closely at every budget — its designable fraction 0.886/0.880/0.781/0.533 is within noise of QFlow's 0.885/0.870/0.778/0.559 in Table 1, including the same collapse at T=10, which is the signature of the instantaneous-velocity parameterisation. The endpoint+semigroup variant is stronger in the aggressive regime but still falls short of MeanFlow at every budget. Swapping in our MeanFlow target, on the same trunk and from the same initialization, recovers the few-step regime (0.867 at T=20, 0.728 at T=10).

We read this as a matter of training budget rather than of correctness: the two objectives share a minimizer (Remark 13), but the semigroup constraint propagates the boundary condition only through the interval triples it happens to sample, whereas the differential target imposes the same consistency pointwise at every (s, t). The semigroup route therefore appears to need a substantially longer schedule to reach the few-step regime — consistent with RMF's $\sim$598k steps, and with our observation in Appendix K that semigroup training gave weaker few-step gains than the JVP-based loss at 10–20k steps.

O.3 Additional results for V-QFlow and V-ReQFlow

V-QFlow and V-ReQFlow Yue et al. [2025] do not provide checkpoints trained on SCOPe; therefore, the scores in Table 11 are computed using the released checkpoints trained on PDB. Because this evaluation uses a different training dataset and pipeline from the SCOPe-based baselines, the results

Objective	Samp. Steps	Train Steps	Designability			Div. ↓	Secondary structure	
			Frac. ↑	scRMSD ↓	scTM ↑		Helix / Strand	ca-valid
FM + semigroup (RMF)	100	6.8k	0.886	1.285±0.962	0.892±0.078	**0.376**	0.498/0.199	0.976
	50	-	0.880	1.337±1.027	0.889±0.079	**0.369**	0.466/0.221	0.975
	20	-	0.781	1.665±1.366	0.856±0.101	**0.370**	0.474/0.197	0.975
	10	-	0.533	2.825±2.327	0.765±0.148	**0.368**	0.446/0.171	0.983
endpoint + semigroup	100	6.2k	0.852	1.391±1.321	0.887±0.095	0.396	0.549/0.160	0.971
	50	-	0.835	1.405±1.245	0.885±0.093	0.391	0.521/0.180	0.972
	20	-	0.813	1.534±1.489	0.873±0.096	0.392	0.561/0.152	0.977
	10	-	0.712	1.975±1.783	**0.835±0.120**	0.389	0.551/0.147	0.984
endpoint + MeanFlow (ours)	100	6.1k	**0.936**	**1.106±0.870**	**0.909±0.067**	0.415	0.536/0.200	0.971
	50	-	**0.906**	**1.246±1.190**	**0.902±0.078**	0.413	0.517/0.213	0.971
	20	-	**0.867**	**1.369±1.144**	**0.883±0.087**	0.408	0.505/0.210	0.970
	10	-	**0.728**	**1.940±1.741**	0.832±0.118	0.402	0.497/0.193	0.975

Table 10: **Controlled objective ablation on SCOPe.** All rows start from the same Stage-1 α-Flow checkpoint (step 100k) and share the IPA trunk, data, coupling, self-conditioning and optimizer; only the Stage-2 consistency target is swapped. *Train Steps* counts Stage-2 steps only. The semigroup rows instantiate the objectives of Woo et al. [2026] inside our pipeline; they are *not* a reproduction of that work, whose hyperparameters, model size and training budget we do not replicate. Best per performance metric within each sampling-step group in bold.

are not directly comparable; we thus report them in this separate section. Notably, while the overall score is close to 0.99, the helix accuracy is high whereas the strand secondary-structure accuracy is around 0.1 or lower. This imbalance is concerning biologically because β-strands typically assemble into β-sheets via inter-strand hydrogen bonding and often depend on non-local sequence interactions; correspondingly, strand prediction is generally harder than helix prediction, and very low strand accuracy may indicate that the model fails to capture such long-range constraints Pauling et al. [1951], Zhang and Sagui [2015].

P Computational Resources

All experiments are conducted on a single GPU node with the following configuration. Each node is equipped with 4× NVIDIA H100 80 GB HBM3 GPUs and dual AMD EPYC 9654 96-core CPUs. Training uses all four GPUs with PyTorch Lightning distributed data-parallel (DDP), 14 CPU worker threads per GPU process (56 cores in total), and gradient accumulation of 2; the effective batch is set by a length-based sampler with cap $N_{\max}^2 = 5 \times 10^5$ residues^2 (Table 7).

Method	Steps	Fraction ↑	scRMSD ↓	scTM ↑	Diversity TM ↓	Novelty TM ↓	α-Helix	β-Strand
V-QFlow	100	0.990	0.654 ± 0.446	0.957 ± 0.039	0.384	0.846	0.740 ± 0.260	0.091 ± 0.20
	50	0.980	0.702 ± 0.423	0.952 ± 0.045	0.373	0.837	0.714 ± 0.268	0.109 ± 0.20
	20	0.964	0.869 ± 0.697	0.937 ± 0.053	0.348	0.803	0.710 ± 0.245	0.099 ± 0.18
	10	0.903	1.160 ± 1.183	0.908 ± 0.088	0.332	0.772	0.687 ± 0.227	0.093 ± 0.16
V-ReQFlow	100	0.990	0.675 ± 0.318	0.955 ± 0.033	0.403	0.851	0.808 ± 0.199	0.050 ± 0.15
	50	0.993	0.726 ± 0.625	0.951 ± 0.036	0.396	0.840	0.791 ± 0.216	0.064 ± 0.17
	20	0.975	0.855 ± 0.638	0.936 ± 0.047	0.383	0.812	0.785 ± 0.217	0.064 ± 0.16
	10	0.957	1.083 ± 0.854	0.910 ± 0.065	0.377	0.782	0.788 ± 0.199	0.052 ± 0.14

Table 11: Evaluation results for V-QFlow and V-ReQFlow. These methods use a different data/evaluation pipeline from the other baselines, so we report them separately.