STERIC HAMILTONIAN FLOWS FOR PROTEINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Proteins are central to nearly all biological processes, and their diverse functions arise from their complex 3D structures. Recent advances in deep learning have transformed this field, with diffusion-based generative models revolutionizing protein design by enabling the creation of novel protein structures. However, these methods lack physical inductive biases and neglect the intrinsic physical realism of proteins, driven by noising dynamics that lack grounding in physical principles. To address this, we first introduce a physically motivated structured noising process, grounded in *classical physics*, that unfolds proteins into secondary structures (e.g., α-helices, linear β-sheets) while maintaining bonds and preventing collisions. We integrate this process with $\mathrm{SE}(3)$ flow-matching to model protein backbone distributions while incorporating sequence information for conditional folding. Our method outperforms baselines trained on the same dataset and matches those trained on substantially larger ones across short- and long-chain generation, and motif-scaffolding tasks.

1 INTRODUCTION

Proteins participate in most cellular processes and consist of up to half of all biomass on earth (Schröder, 2017). Understanding their three-dimensional structures is essential to understanding life and addressing global health challenges (Cao et al., 2020; Silva et al., 2019).

In protein *folding*, the physical process of transforming a linear amino acid chain into a functional macromolecule is studied (Onuchic et al., 1997; Dill and MacCallum, 2012; Freddolino et al., 2010), and has inspired several early computational approaches (Hao and Scheraga, 1995; Shakhnovich, 2006; Scheraga et al., 2007). In recent years, this field has been transformed by deep learning breakthroughs (AlQuraishi, 2019; Jumper et al., 2021; Abramson et al., 2024). On the other hand, *generative* protein models aim to learn the distribution $p(\mathbf{x})$ of physically realistic, folded protein structures $\mathbf{x}$ (Watson et al., 2023; Ingraham et al., 2023; Geffner et al., 2025). Most approaches adopt diffusion (Song et al., 2021; Ho et al., 2020) and flow-model (Grathwohl et al., 2018; Lipman et al., 2023) paradigms, where a noise process perturbs proteins into a set of disconnected residues, and a denoiser network learns to reconstruct the folded structure.

Recent advances have extended these models to the space of rotations and translations of protein backbones with $\mathrm{SE}(3)$-invariant generative models (Yim et al., 2023b;a; Bose et al., 2024; Huguet et al., 2024). While linear forward noising processes offer simplicity, they disregard the inherent structure of the data: residues decorrelate independently, destroying spatial continuity along the trajectory. This deprives the reverse network of a structural inductive bias and forces it to learn geometric and physical coherence entirely from scratch. We therefore study the following question:

How can we leverage physical principles to devise a structured noising process for generative models in de novo protein design?

Contributions. Toward that end, we introduce a physics-informed structured noising process derived from Hamiltonian dynamics (Hamilton, 1834), which incorporates simple yet powerful inductive biases that that ensure structural integrity and avoid collisions. We then learn to reverse this process using flow matching on $\mathrm{SE}(3)$, further incorporating sequence information to unify structure generation and folding. In particular, our contributions are,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: **Overview of deep learning methods for protein structure generation.** † indicates the methods that only penalize the model to be bond-persistent at the final steps of sampling. We use the following abbreviations: DDPM (Denoising Diffusion Probabilistic Models), CFM (Conditional Flow Matching), OT (Optimal Transport), and DSM (Denoising Score Matching).

Method	Model	Backbone preserving	Collision free	Sequence Augmented	Coordinate system	Citation
RFDiffusion	DDPM	✗	✗	✗	Cartesian	Watson et al. (2023)
Chroma	DDPM	✗	✗	✗	Cartesian	Ingraham et al. (2023)
FrameFlow	CFM	✓(†)	✗	✗	Frames	Yim et al. (2023a)
FoldFlow	CFM/OT	✓(†)	✗	✗	Frames	Bose et al. (2024)
FoldFlow-2	CFM/OT	✓(†)	✗	✓	Frames	Huguet et al. (2024)
MultiFlow	CFM	✗	✗	✓	Frames	Campbell et al. (2024)
FrameDiff	DSM	✓(†)	✗	✗	Frames	Yim et al. (2023b)
FoldingDiff	DDPM	✓	✗	✗	Angular	Wu et al. (2024)
StericFlow	CFM	✓	✓	✓	Frames + Angular	this work

- We propose a physics-inspired structured noising process that unfolds proteins into linear chains while preserving structural integrity and avoiding clashes, with theoretical convergence guarantees.
- We propose **StericFlow**, which integrate this process with the flow-matching paradigm to model protein structure with sequence information to enhance generative capabilities.
- Empirically, StericFlow outperforms baselines trained on the same dataset and achieves performance comparable to those trained on substantially larger datasets short and long chains, and motif scaffolding tasks.

2 Background

Protein Backbone Parametrization. We adopt the backbone frame parameterization introduced in AlphaFold2 (Jumper et al., 2021). For a protein of length N, the structure is represented by N frames, each of which is $\mathrm{SE}(3)$-equivariant. A frame is defined relative to fixed reference coordinates (idealized values of Alanine) $\mathrm{N}^*, \mathrm{C}_\alpha^*, \mathrm{C}^*, \mathrm{O}^* \in \mathbb{R}^3$, with $\mathrm{C}_\alpha^* = (0,0,0)$ being the origin, assuming idealized bond angles and lengths (Engh and Huber, 2006). The coordinates of residue i are obtained by applying a rigid-body transformation $T_i = (\mathbf{r}_i, \mathbf{x}_i) \in \mathrm{SE}(3)$ to the reference coordinates, i.e.,

$$[\mathrm{N}_i, (\mathrm{C}_\alpha)_i, \mathrm{C}_i] = T_i \cdot [\mathrm{N}^*, \mathrm{C}_\alpha^*, \mathrm{C}^*] \tag{1}$$

where $r_i \in \mathrm{SO}(3)$ denotes the rotation matrix and $x_i \in \mathbb{R}^3$ denotes the translation. Collectively, the set of transformations $\mathbf{T} = [T_1, \ldots, T_N] \in \mathrm{SE}(3)^N$ represent the protein. In addition, we also use an angular representation, denoted by $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$, where each residue $\mathbf{z}_i = (\phi, \psi, \omega, \theta_1, \theta_2, \theta_3) \in [-\pi, \pi]^6$ is represented using the three dihedral (ϕ, ψ, ω) and bond $(\theta_1, \theta_2, \theta_3)$ angles of the backbone. Cartesian coordinates $\mathbf{x}$ can be reconstructed via the mp-NeRF algorithm (Parsons et al., 2005; Alcaide et al., 2022) in a differentiable way, i.e. $\mathbf{x} = \mathrm{nerf}(\mathbf{z})$. See Appendix D for a primer on the coordinate systems.

Flow models. Flow models. Continuous Normalizing Flows (CNFs) (Chen et al., 2018; Grathwohl et al., 2018) learn an unknown data distribution $p_1 = p_{\text{data}}(\mathbf{x})$ by transforming a simple base distribution p_0 into the data distribution via Neural ODEs over the unit interval $t \in [0,1]$,

$$\text{forward}: \quad d\mathbf{x}_t = \mathbf{f}_\theta(\mathbf{x}_t, t)dt, \qquad \mathbf{x}_0 \sim p_0, \tag{2}$$

$$\text{reverse}: \quad d\mathbf{x}_t = -\underbrace{\mathbf{f}_\theta(\mathbf{x}_t, t)}_{\text{velocity field}} dt, \qquad \mathbf{x}_1 \sim p_1 \tag{3}$$

where $\mathbf{f}_\theta$ is a neural network that parametrizes the vector field driving the ODE. Here $t = 0$ indexes the base distribution (prior) and $t = 1$ the data distribution, so Eq. 2 integrates forward in t to generate samples and Eq. 3 integrates the same field backward to recover p_0. Training CNFs typically requires full simulation over $[0,1]$ and log-likelihood estimation via the instantaneous change-of-variables formula. Recently, Flow Matching (FM) (Lipman et al., 2023; Tong et al., 2023) has emerged as an efficient method for training CNFs. The key idea is to construct a smooth probability path $(p_t)_{t\in[0,1]}$, induced by a vector field $u_t(\mathbf{x}_t)$ that interpolates between p_0 and p_1, yielding a simulation-free training objective. However, the closed-form velocity field $u_t(\mathbf{x}_t)$ that generates p_t is generally unavailable. To address this, we instead use conditional vector fields

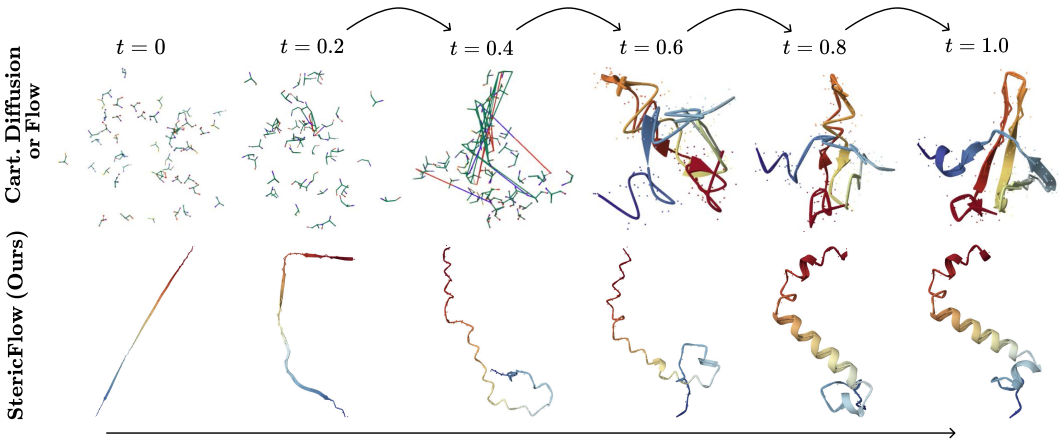


Figure 1: **Generation by StericFlow and cartesian diffusion/flow-based methods**. Inference trajectories for unconditional monomer generation are compared between StericFlow and cartesian diffusion/flow-based methods.

$u_t(\mathbf{x}_t|\mathbf{z})$, which define a conditional probability path $p_t(\mathbf{x}_t|\mathbf{z})$. The unconditional probability path can then be recovered by marginalizing over the conditioning variables $\mathbf{z}$. This gives a tractable objective, trained by regressing against the conditional vector field with $t \in \mathcal{U}[0, 1]$, and $\mathbf{z} \sim q(\mathbf{z})$ as $\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\mathbf{x}_t,\mathbf{z}}\left[||\mathbf{f}_\theta(\mathbf{x}_t, t) - u_t(\mathbf{x}_t|\mathbf{z})||^2\right]$. For more details, see Lipman et al. (2024).

Higher-order diffusion and flow models. Recent advances in generative modeling have extended beyond standard first-order diffusion/flow methods (Song et al., 2021; Lipman et al., 2023) to incorporate higher-order dynamics, beginning with momentum-based Langevin diffusion (Dockhorn et al., 2022) and progressing to higher-order flow formulations (Holderrieth et al., 2024; Zhang et al., 2025; Cao et al., 2025; Su et al., 2025). While these methods achieve improved sample quality and faster convergence in image generation, they generally lack the physical inductive biases necessary for modeling complex biological systems. Consequently, the use of higher-order dynamics for modeling macromolecules remains largely unexplored, representing a critical gap in leveraging physics-informed generative methods for protein structure generation.

3 How realistic is the generative process?

Linear diffusion and flow models are blind to protein physics. Linear diffusion and flow models typically employ a linear Ornstein–Uhlenbeck forward process (Yim et al., 2023b; Bose et al., 2024; Santos and Lin, 2023) in $\mathbb{R}^n$, yielding closed-form transition kernels for the forward marginal p_t. While computationally convenient, this process overlooks the physical realism of proteins (Outeiral et al., 2022; Chakravarty et al., 2023; Uversky, 2002), treating residues as independent points rather than physical entities. Consequently, it *destroys* structural integrity by decorrelating residues and inducing silent collisions, thereby imposing an undesirable inductive bias, as illustrated in Fig. 2 for Cartesian diffusion and flow-based models. This inductive bias in the forward process (Vastola, 2025) is mirrored in the reverse generative process (Ho et al., 2020; Song et al., 2021), and is empirically seen to generate structures with steric clashes, violations of fundamental constraints (Anand et al., 2025), and biases (Lu et al., 2025) that hinder the recovery of motifs.

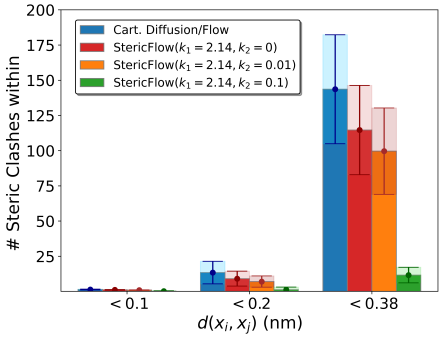


Figure 2: Average number of residue–residue steric collisions (pairwise distance $< d(\mathbf{x}_i, \mathbf{x}_j)$ nm) across the forward trajectory of a protein.

Physics-driven non-linear protein noising. To overcome these issues while maintaining physically meaningful correlations (Brini et al., 2020) during the noising process, we argue that a generative (reverse) process can benefit from a carefully designed forward process that *avoids collisions and*

preserves topological integrity, providing the model with an inductive bias aligned with actual protein dynamics. Such a forward process is naturally higher-order and non-linear, as structural constraints, dependencies, and related effects are not well captured by linear dynamics (Frauenfelder, 2010).

In the following section, we show how to devise a physics-driven structured noising process by drawing inspiration from *classical physics*, and how this process can be seamlessly combined with the flow-matching to model protein structure.

4 METHOD

Our goal is to learn an $\mathrm{SE}(3)$-invariant density ρ_t via flows by pushing forward an initial invariant distribution ρ_0 to the empirical protein distribution ρ_1. Translation invariance can be enforced by centering each protein at its center of mass (Rudolph et al., 2021), yielding an invariant measure on $\mathrm{SE}(3)_0^N$ for a protein of length N. Since $\mathrm{SE}(3)_0^N$ forms a product group, it admits a residue-wise decomposition, allowing the flow to be constructed as independent flows over residues. A recap of Lie groups is provided in App. A.

Decomposing $\mathrm{SE}(3)$ into $\mathrm{SO}(3)$ and $\mathbb{R}^3$. $\mathrm{SE}(3)$ is a Lie group that can be represented as a semidirect product, $\mathrm{SE}(3) \cong \mathrm{SO}(3) \ltimes (\mathbb{R}^3, +)$. Accordingly, the metric can be decomposed as $\langle \mathfrak{x}, \mathfrak{x}' \rangle_{\mathrm{SE}(3)} = \langle \mathfrak{r}, \mathfrak{r}' \rangle_{\mathrm{SO}(3)} + \langle \mathbf{x}, \mathbf{x}' \rangle_{\mathbb{R}^3}$ (Bose et al., 2024). This choice allows the $\mathrm{SE}(3)$-invariant measure to be decomposed into an independent $\mathrm{SO}(3)$-invariant measure and a measure proportional to the Lebesgue measure on $\mathbb{R}^3$ (Pollard, 2002). Consequently, we can construct independent flows on $\mathrm{SO}(3)$ and $\mathbb{R}^3$. We next describe how to design unfolding flows within this framework.

4.1 BUILDING $\mathrm{SO}(3)$ FLOWS

We aim to construct a flow on $\mathrm{SO}(3)$ that transports a prior distribution ρ_0 (e.g. $\mathcal{U}(\mathrm{SO}(3))$) to a target data distribution ρ_1. Following the parametrization strategy outlined in Bose et al. (2024), we define a time-dependent vector field $v_t(\mathbf{r}_t | \mathbf{r}_0, \mathbf{r}_1)$ that lies in the tangent space $\mathcal{T}_{\mathbf{r}_t}\mathrm{SO}(3)$ and depends on both the initial $\mathbf{r}_0$ and the target $\mathbf{r}_1$ (Tong et al., 2023). We adopt an approach based on geodesics on $\mathrm{SO}(3)$, by defining the interpolation between $\mathbf{r}_0$ and $\mathbf{r}_1$ along the geodesic as $\mathbf{r}_t = \exp_{\mathbf{r}_0}(t \log_{\mathbf{r}_0}(\mathbf{r}_1))$.

However, evaluating the exponential and logarithm maps on $\mathrm{SO}(3)$ is computationally expensive and numerically unstable, making error control costly (Al-Mohy and Higham, 2012). To mitigate this, we adopt a numerical strategy proposed by Bose et al. (2024), which involves converting $\mathbf{r}_1$ into its axis-angle representation and applying parallel transport to efficiently compute $\log_{\mathbf{r}_0}(\mathbf{r}_1)$.

Given $\mathbf{r}_t$, we can utilize the Riemannian flow matching framework (Chen and Lipman, 2023) to build a conditional flow, giving $v_t = \dot{\mathbf{r}}_t$, which entails estimating the gradient of $\mathbf{r}_t$ at time t. To mitigate this issue, we instead compute the element of $\mathfrak{so}(3)$ corresponding to the relative rotation between $\mathbf{r}_0$ and $\mathbf{r}_t$, given as $\mathbf{r}_t^\top \mathbf{r}_0$ (Bose et al., 2024). Taking the matrix logarithm of this relative rotation and dividing by t yields a skew-symmetric matrix in $\mathfrak{so}(3)$, representing the velocity vector pointing toward the target $\mathbf{r}_1$. We can then multiply by $\mathbf{r}_t$ to parallel transport over the tangent space. Finally, expressing these operations as $\log_{\mathbf{r}_t}(\mathbf{r}_0)$, we obtain the final velocity field as $v_t = \dot{\mathbf{r}}_t = \log_{\mathbf{r}_t}(\mathbf{r}_0)/t$

4.2 STRUCTURED NOISING IN $\mathbb{R}^3$

We seek to define a structured noising process on $\mathbb{R}^3$: one that unfolds a protein while maintaining chain connectivity and suppressing steric overlap at intermediate times, properties that linear noising destroys. Our setting is sequence-agnostic, since in unconditional generation the amino acid identities are unavailable and atom types, side-chain geometry, and residue-specific interactions are therefore undefined; we consequently work with generic geometric and steric quantities alone.

We build such a process by leveraging the angular representation, which, unlike raw Cartesian coordinates, is inherently translation-invariant and thus a natural choice for this purpose. We formalize this in the following proposition:

Proposition 1 (Translation Invariance of Angular Representation). *Let $\mathbf{x} \in \mathbb{R}^{3N}$ denote the backbone Cartesian coordinates of a protein with N residues, and let $\mathbf{z}(\mathbf{x})$ be its angular representation. Let $\mathcal{T}_\mathbf{t} : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ denote a translation by vector $\mathbf{t} \in \mathbb{R}^3$ applied to all residues. Then, the angular representation is translation-invariant, i.e., $\mathbf{z}(\mathcal{T}_\mathbf{t} \circ \mathbf{x}) = \mathbf{z}(\mathbf{x})$, for all $\mathbf{t} \in \mathbb{R}^3$.*

We provide the proof in Appendix D. Moreover, since there exists a bijection between the angular and Cartesian domains (Parsons et al., 2005; Ingraham et al., 2019), any flow defined in angular space induces a corresponding flow in Cartesian space that preserves translation invariance.

We define the structured forward process via a second-order damped Hamiltonian ODE (Desai et al., 2021; Neary and Topcu, 2023) over the angular space $\mathbf{z}$ toward the prior at $s = T$,

$$\text{unfolding:} \quad \begin{cases} \dot{\mathbf{z}}_s & = \mathbf{v}_s \\ \dot{\mathbf{v}}_s & = f_{\text{unfold}}(\mathbf{z}_s, \mathbf{v}_s) = -\nabla_{\mathbf{z}_s} U(\mathbf{z}_s) - K(\mathbf{v}_s), \end{cases} \tag{4}$$

driven by a potential $U(\mathbf{z}_s)$ and a dampening friction term $K(\mathbf{v}_s)$. Our main goal is to unfold into a secondary structure without clashes. We propose an augmented potential as,

$$U(\mathbf{z}_s) = k_1 U_{\text{target}}(\mathbf{z}_s) + k_2 U_{\text{repulsion}}\big(\text{nerf}(\mathbf{z}_s)\big) \tag{5}$$

that incorporates both harmonic attraction to $\mathbf{z}_{\text{target}}$ (e.g. can be a secondary structure chain such as β-sheet, α-helix, etc.) and inverse-distance steric repulsion,

$$U_{\text{target}}(\mathbf{z}_s) = \sum_i \big(\mathbf{z}_{i,s} - \mathbf{z}_{i,\text{target}}\big)^2, \quad U_{\text{repulsion}}(\mathbf{x}_s) = \sum_{i \neq j} \frac{1}{||\mathbf{x}_{i,s} - \mathbf{x}_{j,s}||}, \tag{6}$$

where i, j denote the residue indices along the backbone, $\mathbf{z}_{\text{target}} \sim p_T(\mathbf{z})$, and k_1, k_2 are the respective weights. The target potential pulls the protein toward the target state, while the repulsion acts as a steric barrier that suppresses collisions along the trajectory. We use JAX autodiff to compute $\nabla_{\mathbf{z}_s} U(\mathbf{z}_s)$, which also flows through the cartesian reconstruction nerf (Bradbury et al., 2018).

Friction. The potential energy $U(\mathbf{z}_s)$ creates highly oscillatory dynamics requiring stabilization. We introduce a drag force $K(\mathbf{v}_s) = \gamma \mathbf{v}_s$ where $\gamma > 0$ is a drag coefficient representing environmental resistance. This dampening term counteracts strong accelerations and stabilizes the system, ensuring convergence to $\mathbf{z}_{\text{target}}$. The following result formalizes its role in guaranteeing convergence,

Proposition 2 (Exponential contraction under damping). *Let $\mathcal{H}$ be the Hamiltonian of the system described in Eq. 4. Under damping $\gamma > 0$:*

- *S1. $\mathcal{H}$ is a Lyapunov function with $\dot{\mathcal{H}} \leq 0$, implying $\lim_{s\to\infty} \|\mathbf{v}(s)\| = 0$ and trajectories converge to a critical point $\mathbf{z}^*$ of $U(\mathbf{z})$;*
- *S2. Convergence is exponential: $\|\mathbf{z}_s - \mathbf{z}^*\| \leq c\, e^{-\lambda s}$ for some $c, \lambda > 0$ depending on γ and the curvature of U at $\mathbf{z}^*$;*
- *S3. For $k_1 \gg k_2$, $\mathbf{z}^*$ lies within $\mathcal{O}(k_2/k_1)$ of $\mathbf{z}_{\text{target}}$.*

The proof is given in Appendix D. Proposition 2 provides the well-posedness required for flow matching via exponential contraction toward a well-defined attractor. We integrate Eq. 4 in simulation time s up to a horizon T chosen such that $c\, e^{-\lambda T} \leq \sigma_\beta$ by *S2.*, and rescale as $t = 1 - s/T$, so that $t = 0$ is the prior and $t = 1$ the data, leaving the flow-matching parametrization unchanged.

Terminal distributions. The terminal distribution is chosen to represent a *weakly informative* prior. In the context of proteins, an unfolded linear chain of amino acids provides a physically meaningful prior, commonly referred to as a linear β-sheet [1] (Caballero et al., 2014). Denoting the state by $\mathbf{u} = [\mathbf{z}, \mathbf{v}]$, the terminal and data distribution is defined as,

$$p_0(\mathbf{u}) = \underbrace{\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\beta, \sigma_\beta^2 I)}_{\text{position } p_0(\mathbf{z})} \cdot \underbrace{\mathcal{N}(\mathbf{v}|\mathbf{0}, \sigma_v^2 I)}_{\text{velocity}}, \quad p_1(\mathbf{u}) = p_{\text{data}}(\mathbf{z}) \cdot \mathcal{N}(\mathbf{v}|\mathbf{0}, \sigma_v^2 I), \tag{7}$$

where there is a small variation σ_β^2 around the β-sheet angles μ_β [2] and the initial velocity is Gaussian. We have observations from data distribution without velocities, and we augment the data with Gaussian velocities of variance σ_v^2. The hyperparameter values are provided in Table 7. We also evaluate an α-helical variant in Section 6.1.

Computing Forward Simulations. The inverse-distance repulsion makes the forward transition kernel analytically intractable in closed form, so we rely on forward simulations, integrated over

[1] https://en.wikipedia.org/wiki/Beta_sheet

[2] β-sheet angles

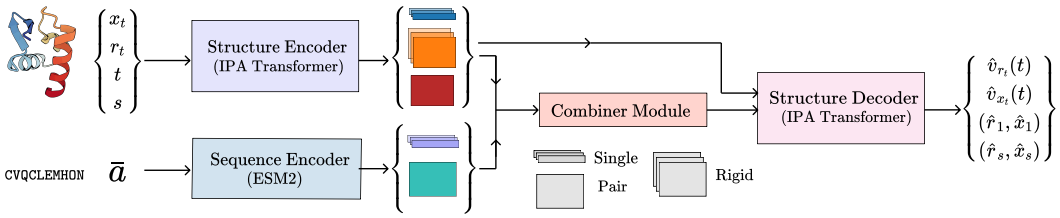


Figure 3: **StericFlow Model Pipeline** (f_η). The model takes as input a noised structural state together with the sequence information. These are first processed independently by a structure encoder and a sequence encoder. The resulting representations are then integrated through a combiner module, after which a structure decoder predicts both the velocity field and the auxiliary predictions.

$[0, T]$ and rescaled onto $[0, 1]$ as above. For any given time step t with a given data point $\mathbf{u}_1 \sim p_1(\mathbf{u})$ and the prior $\mathbf{u}_0 \sim p_0(\mathbf{u})$, the forward distribution is given by:

$$p(\mathbf{u}_t|\mathbf{u}_1, \mathbf{u}_0) = \mathcal{N}\left(\begin{bmatrix}\mathbf{z}_t \\ \mathbf{v}_t\end{bmatrix}; \begin{bmatrix}\mu_{\mathbf{z}_t} \\ \mu_{\mathbf{v}_t}\end{bmatrix}, \begin{bmatrix}\sigma_z^2 \\ \sigma_v^2\end{bmatrix} \mathrm{I}\right), \quad \text{where} \begin{bmatrix}\mu_{\mathbf{z}_t} \\ \mu_{\mathbf{v}_t}\end{bmatrix} = \begin{bmatrix}\mathbf{z}_1 \\ \mathbf{v}_1\end{bmatrix} - \int_t^1 \begin{bmatrix}\dot{\mathbf{z}}_t \\ \dot{\mathbf{v}}_t\end{bmatrix} d\tau \tag{8}$$

where $\sigma_{z,v} = \sqrt{0.1}$ is the variance typically assumed to be fixed across intermediate distributions and chosen based on the data, and $\mu_{\mathbf{z}_t, \mathbf{v}_t}$ is obtained by solving a second-order ODE system that governs the dynamics under the potential. Recall that due to bijection, we can obtain the flow in cartesian space by applying nerf as, $p_{\text{cart}}(\mathbf{u}_t|\mathbf{u}_1, \mathbf{u}_0) = p(\text{nerf}(\mathbf{u}_t)|\,\text{nerf}(\mathbf{u}_1), \text{nerf}(\mathbf{u}_0))$, where $\mathbf{u}_1 \sim p_1(\mathbf{u})$, and $\mathbf{u}_0 \sim p_0(\mathbf{u})$ (see App. D.2 for details). Given a dataset, we can simulate forward trajectories and store the resulting sample trajectories, which can then be used to train a model to learn the reverse process via conditional flow matching (Lipman et al., 2023) in $\mathbb{R}^3$.

4.3 Model Architecture

We employ a multimodal network $f_\eta(\mathbf{x}_t, \mathbf{r}_t, \bar{a}, t, s)$ similar to Huguet et al. (2024), that takes as input the noised translation–rotation frames, sequence information, and the time step, and predicts the corresponding velocity along with auxiliary outputs. The architecture comprises the following:

Structure Encoder. To encode structural information, we adopt the invariant point attention (IPA) transformer (Jumper et al., 2021), an $\mathrm{SE}(3)$-equivariant architecture. The IPA is highly flexible: it can both consume and generate structures as N rigid frames, while simultaneously producing single and pairwise representations of the input.

Sequence Encoder. We incorporate sequence information using the 650M-parameter pre-trained ESM2 model (Lin et al., 2022), which provides sequence representations. It outputs both single and pair representations for a given amino acid sequence, analogous to the representations produced by the structure encoder. To enable both conditional and unconditional training, sequence information a is randomly masked with probability p_{uncond},

$$\bar{a} = \mathbb{1}(p_{\text{uncond}} < 0.5) \cdot a + \big(1 - \mathbb{1}(p_{\text{uncond}} < 0.5)\big) \cdot \emptyset$$

where $p_{\text{uncond}} \sim \mathcal{U}[0, 1]$, and $\mathbb{1}(p_{\text{uncond}} < 0.5)$ is the indicator function that ensures sequence information is available 50% of the time.

Combiner Module and Structure Decoder. After encoding both the input structure and sequence, we construct a joint representation of the single and pair embeddings, following Huguet et al. (2024), which iteratively updates them via triangular self-attention. These updated representations are processed by an IPA transformer decoder to predict velocity fields and auxiliary outputs.

The full model pipeline is illustrated in fig. 3, and the training objective is described next.

5 Learning Objective

The forward unfolding dynamics are modeled as Hamiltonian ODEs (Neal et al., 2011; Holderrieth et al., 2024). Following Holderrieth et al. (2024), the reverse-time generative process maps to a deterministic Fokker–Planck probability flow learned via standard flow matching. Accordingly, we define the learning objective using flow matching as follows,

Flow matching Loss. We utilize the conditional flow matching (CFM) loss (Lipman et al., 2023; Tong et al., 2023) to learn the velocity fields. Specifically, for $\mathrm{SO}(3)$ and $\mathbb{R}^3$, the corresponding CFM objectives are defined as

$$\mathcal{L}_{\mathrm{SO}(3)} = \mathbb{E}_{t,\mathbf{r}_t,\mathbf{r}_0,\mathbf{r}_1}\left\|\frac{\log_{\mathbf{r}_t}(\mathbf{r}_0)}{t} - \hat{v}_{\mathbf{r}_t}(t)\right\|^2_{\mathrm{SO}(3)}, \; \mathcal{L}_{\mathbb{R}^3} = \mathbb{E}_{t,\mathbf{x}_t,\mathbf{x}_0,\mathbf{x}_1}\left\|v(\mathbf{x}_t|\mathbf{x}_1,\mathbf{x}_0) - \hat{v}_{\mathbf{x}_t}(t)\right\|_2^2 \quad (9)$$

where $\mathbf{r}_t \sim p(\mathbf{r}_t|\mathbf{r}_1,\mathbf{r}_0)$, $\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_1,\mathbf{x}_0)$, $\hat{v}_{\mathbf{x}_t}(t), \hat{v}_{\mathbf{r}_t}(t)$ denotes the velocity fields predicted by the model, $t \sim \mathcal{U}[0,1]$, $(\mathbf{x}_0,\mathbf{x}_1) \sim q(\mathbf{x}_0,\mathbf{x}_1) = p(\mathrm{nerf}(\mathbf{z}_0))p(\mathrm{nerf}(\mathbf{z}_1))$, $(\mathbf{r}_0,\mathbf{r}_1) \sim q(\mathbf{r}_0,\mathbf{r}_1) = \rho(\mathbf{r}_0)\rho(\mathbf{r}_1)$, with $\log_{\mathbf{r}_t}(\mathbf{r}_0)/t$ denoting the velocity field on $\mathrm{SO}(3)$, and $v(\mathbf{x}_t|\mathbf{x}_1,\mathbf{x}_0)$ the corresponding velocity field on $\mathbb{R}^3$.

Look-Ahead (LA) Loss. To help the model learn non-linear unfolding trajectories, we introduce a novel look-ahead loss that penalizes the model for inaccurate predictions of future states. This loss is conceptually similar to the one proposed in Zhou et al. (2025). Specifically, it is defined as

$$\mathcal{L}_{\mathrm{LA}} = \mathbb{E}_{t,s}\left[\frac{1}{4N}\sum_{n=1}^{N}\sum_{a\in\Omega}||a_n^{(s)} - \hat{a}_n^{(s)}||^2\right], \text{ where } t \sim \mathcal{U}[0,1], \text{ and } s \sim \mathcal{U}[t,1] \quad (10)$$

where $a_n^{(s)} = \zeta_{\text{frame-to-atom}}(\mathbf{r}_s,\mathbf{x}_s)$, $\hat{a}_n^{(s)} = \zeta_{\text{frame-to-atom}}(\hat{\mathbf{r}}_s,\hat{\mathbf{x}}_s)$, $\zeta_{\text{frame-to-atom}}$ represent the transformation from frames to coordinates, and $\Omega = \{\mathrm{N},\mathrm{C},\mathrm{O},\mathrm{C}_\alpha\}$.

Auxiliary Losses. To encourage the model to capture fine-grained local characteristics, we incorporate two auxiliary losses (Yim et al., 2023b) at the final step of generation defined as,

$$\mathcal{L}_{\mathrm{bb}} = \frac{1}{4N}\sum_{n=1}^{N}\sum_{a\in\Omega}||a_n^{(1)} - \hat{a}_n^{(1)}||^2, \quad \mathcal{L}_{\mathrm{2D}} = \frac{1}{Z}\sum_{n,m=1}^{N}\sum_{a,b\in\Omega}\mathbb{1}(d_{ab}^{nm} < 0.6)||d_{ab}^{nm} - \hat{d}_{ab}^{nm}||^2 \quad (11)$$

where $a_n^{(1)} = \zeta_{\text{frame-to-atom}}(\mathbf{r}_1,\mathbf{x}_1)$, $\hat{a}_n^{(1)} = \zeta_{\text{frame-to-atom}}(\hat{\mathbf{r}}_1,\hat{\mathbf{x}}_1)$, $\Omega = \{\mathrm{N},\mathrm{C},\mathrm{O},\mathrm{C}_\alpha\}$, $d_{ab}^{nm} = ||a_n^{(1)} - b_m^{(1)}||$, $\hat{d}_{ab}^{nm} = ||\hat{a}_n^{(1)} - \hat{b}_m^{(1)}||$, and $Z = (\sum_{n,m=1}^{N}\sum_{a\in\Omega}\mathbb{1}(d_{ab}^{nm} < 0.6)) - N$. Here $\mathbb{1}(d_{ab}^{nm} < 0.6)$ is an indicator function that restricts the loss to atom pairs within 0.6nm.

We combine the flow-matching loss with the auxiliary and look-ahead losses to get the final loss:

$$\mathcal{L} = \mathcal{L}_{\mathrm{SO}(3)} + \mathcal{L}_{\mathbb{R}^3} + \mathcal{L}_{\mathrm{LA}} + \lambda \cdot \mathbb{1}(t > 0.75)\,(\mathcal{L}_{\mathrm{bb}} + \mathcal{L}_{\mathrm{2D}}) \quad (12)$$

where the auxiliary losses are applied only during the later stages of generation ($t > 0.75$), as indicated by the indicator function $\mathbb{1}(t > 0.75)$, with a scaling λ.

6 Experiments

Tasks. We evaluate StericFlow's generative capabilities in three settings: (i) unconditional monomer backbone generation across varying protein lengths (Section 6.1); (ii) sequence-conditioned generation, where the structure is generated from a given sequence (Section 6.2); and (iii) motif-scaffolding, where conditioned on a motif we generate the full structure (Section 6.3).

Data. We train StericFlow on monomers with lengths between 60 and 512 residues and structural resolution < 5Å, obtained from the Protein Data Bank (PDB) (Berman et al., 2000). Following the filtering criteria outlined in Yim et al. (2023b); Bose et al. (2024), we obtain a dataset of 24,003 proteins. For each protein, we run forward unfolding simulations and store the resulting trajectories for model training. More details can be found in Appendix B.

Competing methods. We compare StericFlow against a diverse set of protein structure generation methods. These include diffusion-based approaches such as FrameDiff (Yim et al., 2023b), RFDiffusion (Watson et al., 2023), Chroma (Ingraham et al., 2023), EigenFold (Jing et al., 2023), FoldingDiff (Wu et al., 2024) and Genie (Lin and AlQuraishi, 2023); flow-based approaches such as FoldFlow's (Bose et al., 2024; Huguet et al., 2024), FrameFlow (Yim et al., 2023a), and Proteína (Geffner et al., 2025); as well as ESM3 (Hayes et al., 2025).

Table 2: **StericFlow Unconditional Backbone Generation Performance**. Comparison of designability, novelty, secondary structure composition, and diversity of generated protein structures without using ESM2 embedding. RFDiffusion uses pretraining, while FoldFlow-2, ESM-3, Chroma, and Proteína leverage additional/different datasets (e.g. $\mathcal{D}_{\text{AFDB}}$, $\mathcal{D}_{\text{ext}}$) for training or fine-tuning. Best results are in **bold**, second-best are underlined.

	Method	Designability	Novelty		Diversity		Sec. Struct %
		Frac. < 2Å (↑)	TM-Frac. < 0.3 (↑)	avg. max TM (↓)	Cluster (↑)	TM-Sc. (↓)	(α/β)
$\mathcal{D}_{\text{PDB}} \cup \mathcal{D}_{\text{ext}} \cup \mathcal{D}_{\text{AFDB}}$	RFDiffusion†	0.94	0.11 ± 0.02	0.42 ± 0.01	0.46 (217)	0.25	64.3/17.2
	FoldFlow-2†	0.97	0.36 ± 0.03	0.36 ± 0.01	0.34 (210)	0.20	82.7/2.0
	Proteína†	0.99	0.38 ± 0.01	0.31 ± 0.01	0.30 (294)	0.39	62.2/9.9
	ESM3†	0.22	0.23 ± 0.03	0.42 ± 0.01	0.58 (64)	0.42	64.5/8.5
	Chroma†	0.74	0.21 ± 0.03	0.41 ± 0.01	0.51 (190)	0.27	69.0/12.5
	Genie†	0.55	0.12 ± 0.02	0.43 ± 0.01	0.27 (203)	**0.22**	72.7/4.8
$\mathcal{D}_{\text{PDB}}$	RFDiffusion (no-pretraining)	0.17	0.01 ± 0.005	0.62 ± 0.02	0.13 (87)	0.41	70.4/19.4
	FoldingDiff	0.10	0.18 ± 0.01	0.44 ± 0.02	0.28 (107)	0.26	78.1/5.7
	EigenFold	0.71	0.13 ± 0.01	0.47 ± 0.02	0.33 (142)	0.27	−/−
	Proteus	0.91	0.14 ± 0.01	0.39 ± 0.02	0.23 (157)	0.26	84.1/3.2
	Genie2	0.92	0.17 ± 0.01	$\mathbf{0.37} \pm 0.02$	**0.59** (281)	0.26	84.1/3.2
	FrameDiff	0.65	0.02 ± 0.01	0.54 ± 0.04	0.39 (126)	0.23	64.9/11.2
	FrameFlow	0.88	0.11 ± 0.01	0.48 ± 0.02	0.53 (236)	**0.22**	55.7/18.4
	FoldFlow (CFM)	0.92	0.18 ± 0.02	0.45 ± 0.02	0.20 (98)	0.24	86.1/1.2
	FoldFlow (OT)	0.94	0.18 ± 0.01	0.41 ± 0.02	0.37 (178)	0.26	84.1/3.2
	Linear-OU FM	0.93	0.14 ± 0.01	0.44 ± 0.02	0.35 (164)	0.25	−/−
	StericFlow ($k_2 = 0$)	0.94	0.16 ± 0.01	0.41 ± 0.03	0.49 (210)	0.25	75.6/4.3
	StericFlow (α-prior)	**0.98**	0.17 ± 0.02	0.40 ± 0.02	0.52 (228)	0.23	80.2/6.8
	StericFlow (β-prior)	0.97	$\mathbf{0.19} \pm 0.02$	0.38 ± 0.02	0.50 (220)	**0.22**	75.2/7.7

6.1 Unconditional Monomer Protein Generation

We evaluate StericFlow's unconditional generation on **Designability** (the fraction of generated proteins that are designable), **Novelty**, **Diversity**, and **secondary structure composition** (definitions in App. B.2), for both short (up to 250 residues) and long (up to 500 residues) structures.

Short Chain Generation. We evaluate performance across sequence lengths $\{50, 100, 150, 200, 250\}$ with 50 samples per length. Table 2 summarizes StericFlow's results and raw validity metrics before refolding in Table 11. StericFlow yields the most designable structures among methods trained on $\mathcal{D}_{\text{PDB}}$, approaching Proteína (trained on a substantially larger dataset) while surpassing RFDiffusion and ESM3. RFDiffusion attains comparable designability only when initialized from pre-trained weights (Yim et al., 2023b). We further compare our model variants spanning different prior end states and a $k_2 = 0$ variant dropping the steric term. We see that incorporating the steric term does improve designability and performance is steady across different prior states.

Table 3: **Long Chain Generation.** All methods are trained on $\mathcal{D}_{\text{PDB}}$ up to 512 residues. RFDiffusion (trained only to 384 residues), Genie2 and Proteína (trained on additional data) are excluded.

Method	Desig.	Div. (TM-Sc.) ↓	Nov. (TM-Frac.) ↑
FoldingDiff	0.05	0.17	0.10
FrameDiff	0.20	0.31	0.10
FoldFlow	0.35	0.29	0.11
FrameFlow	0.29	0.31	0.11
StericFlow	**0.38**	**0.28**	**0.12**

Long Chain Generation. We evaluate performance across longer sequence lengths of $\{400, 450, 500\}$ with 50 samples per length. Table 3 restricts the comparison to methods trained on $\mathcal{D}_{\text{PDB}}$ under the same length budget: RFDiffusion is excluded because it is trained only on proteins up to 384 residues and, Genie2 and Proteína are trained on larger corpora making comparison unfair. Among matched baselines, StericFlow outperforms across this longer length regime, generating more designable structures.

6.2 Sequence Conditioned Generation

We benchmark StericFlow on sequence-conditioned folding, where the task is to predict a protein's 3D structure from its amino acid sequence. Evaluation is conducted on two datasets: (i) CAMEO2022 Ye et al. (2024), and (ii) a custom PDB split comprising 24,003 filtered monomers, and divided into training and test sets. We compare against gold-standard folding methods and state-of-the-art sequence-conditioned generative models, FoldFlow-2 and MultiFlow (Campbell et al., 2024), both retrained exclusively on our training data without additional pretraining or fine-tuning. Table 4 com-

Table 4: **Sequence-conditioned generation Performance**. (*Left*) Performance on CAMEO2022 (Ye et al., 2024) dataset, and † denotes the method trained on different and larger datasets. (*Right*) Performance on a held out test set for sequence conditioned monomer folding. In a head-to-head, direct, and fair comparison, StericFlow outperforms baseline methods, achieving better results.

Method	TM Score	RMSD (Å)	GDT-TS	IDDT	CA clash (%)	PepBond break (%)
EigenFold†	0.74	7.65	0.70	0.73	8.0	N/A
ESMFold†	0.84	3.98	0.82	0.87	0.3	4.7
RoseTTAFold2†	0.86	3.52	0.84	0.89	0.3	5.5
OpenFold†	0.87	3.21	0.85	0.89	0.4	2.0
AlphaFold 2†	0.87	3.21	0.86	0.90	0.3	4.8
FoldFlow-2	0.55	14.21	0.55	0.53	0.9	4.7
StericFlow	0.60	12.11	0.60	0.58	0.2	2.1

Method	RMSD (Å)	
	$\ell_{prot} < 200$	$\ell_{prot} \geq 200$
ESMFold†	1.64 ± 2.73	2.82 ± 3.71
MultiFlow	17.41 ± 2.17	20.63 ± 2.75
FoldFlow-2	13.52 ± 1.84	17.88 ± 2.13
StericFlow	$\mathbf{11.08} \pm 1.02$	$\mathbf{15.11} \pm 1.79$

pares StericFlow against baseline methods. Although the folding methods are trained on substantially larger datasets, making direct comparison unfair, StericFlow outperforms FoldFlow 2 and MultiFlow, which are trained on the same data in a head-to-head comparison. Moreover, StericFlow achieves the lowest clash and break rate, demonstrating the effectiveness of our approach.

6.3 Motif-Scaffolding

We also evaluate the performance of StericFlow on the motif-scaffolding experiment based on the RFDiffusion benchmark Watson et al. (2023). To enable this, we augment the model with two features: (i) the motif structure is kept fixed while only non-motif positions defined by the motif mask are unfolded during forward simulation, and (ii) the fixed motif coordinates are provided as conditional inputs for structural context throughout denoising. Figure 4 describes the performance of StericFlow across baselines. Notably, StericFlow achieves strong performance, outperforming both FrameFlow and RFDiffusion in multiple tasks. It achieves 1404 successes, substantially higher than FrameFlow (1099) and RFDiffusion (889).

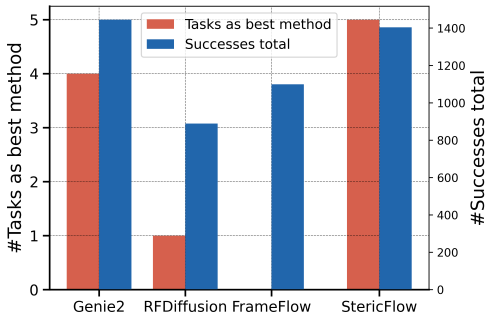


Figure 4: StericFlow outperforms baselines in motif-scaffolding tasks.

7 Ablation Studies

We evaluate the performance and sensitivity of StericFlow across several key dimensions:

1. **Computational Efficiency**: We benchmark the runtime complexity, sampling speed, and parameter footprint of our model against key baselines (App. C.2, C.3).
2. **Effect of varying k_1, k_2 energy weighting**: We analyze how the balance between the energy weighting influences the forward process, described in App. C.1.
3. **Clash profile in the reverse generative process**: We study the clash profile of the sampling trajectory to evaluate the method's efficacy in ensuring physical feasibility (see App. C.4).
4. **Sensitivity & Robustness**: We evaluate model robustness across reverse step-sizes, hyperparameter choices, and external oracle dependencies like ESMFold (App. C.6, C.7).

8 Conclusion

We introduce StericFlow, a generative model built on a physics-informed structured noising process for protein backbones. Unlike linear forward processes, ours keeps intermediate states chain-connected and clash-aware, so these constraints enter as inductive bias rather than being left for the network to recover. Combined with flow matching on $\mathrm{SE}(3)$, this yields accurate generation that respects protein geometry. StericFlow outperforms baselines trained on the same dataset and matches those trained on substantially larger ones across short- and long-chain generation, and motif scaffolding. Future work includes extending beyond monomers and training on larger scale on datasets such as AFDB (Varadi et al., 2022).

AI USE STATEMENT

In this work, we used generative AI tools for grammar refinement, proofreading, and improving prose clarity.

REPRODUCIBILITY STATEMENT

Our work utilizes fully open-source data from the Protein Data Bank, and we follow the preprocessing steps described in the baseline methods, which are also publicly available in their respective papers. Details of the model architecture, hyperparameters, and training setup are provided in the appendix. In addition, code snippets illustrating the forward simulation procedure are included in the appendix for clarity and reproducibility. We will make our code public upon acceptance.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

Awad H Al-Mohy and Nicholas J Higham. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM Journal on Scientific Computing*, 34(4):C153–C169, 2012.

Eric Alcaide, Stella Biderman, Amalio Telenti, and Cyrus Maher. MP-NeRF: A massively parallel method for accelerating protein structure reconstruction from internal coordinates. *Journal of Computational Chemistry*, 2022.

Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 2019.

Rishabh Anand, Chaitanya K Joshi, Alex Morehead, Arian R Jamasb, Charles Harris, Simon V Mathis, Kieran Didi, Rex Ying, Bryan Hooi, and Pietro Liò. Rna-frameflow: Flow matching for de novo 3d rna backbone design. *ArXiv*, pages arXiv–2406, 2025.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000.

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 *ICLR*, 2024.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL github.com/jax-ml/jax.

Emiliano Brini, Carlos Simmerling, and Ken Dill. Protein storytelling through physics. *Science*, 370 (6520):eaaz3041, 2020.

Diego Caballero, Jukka Määttä, Alice Qinhua Zhou, Maria Sammalkorpi, Corey S O'Hern, and Lynne Regan. Intrinsic α-helical and β-sheet conformational preferences: A computational case study of alanine. *Protein Science*, 23(7):970–980, 2014.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 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.

Yang Cao, Yubin Chen, Zhao Song, and Jiahao Zhang. Towards high-order mean flow generative models: Feasibility, expressivity, and provably efficient criteria. *arXiv preprint arXiv:2508.07102*, 2025.

Devlina Chakravarty, Joseph W Schafer, Ethan A Chen, Joseph R Thole, and Lauren L Porter. Alphafold2 has more to learn about protein energy landscapes. *BioRxiv*, 2023.

Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

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

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.

Shaan A Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physical Review E*, 104(3):034312, 2021.

Ken Dill and Justin MacCallum. The protein-folding problem, 50 years on. *Science*, 2012.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *ICLR*, 2022.

RA Engh and R Huber. Structure quality and target parameters. In *International tables for crystallography volume F: crystallography of biological macromolecules*, pages 382–392. Springer, 2006.

Hans Frauenfelder. *The physics of proteins: an introduction to biological physics and molecular biophysics*. Springer Science & Business Media, 2010.

Lydia Freddolino, Christopher B Harrison, Yanxin Liu, and Klaus Schulten. Challenges in protein-folding simulations. *Nature physics*, 6(10):751–758, 2010.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, et al. Proteina: Scaling flow-based protein structure generative models. *arXiv preprint arXiv:2503.00710*, 2025.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Brian C Hall. Lie groups, lie algebras, and representations. In *Quantum Theory for Mathematicians*, pages 333–366. Springer, 2013.

William Rowan Hamilton. *On a general method in dynamics*. Richard Taylor United Kindom, 1834.

Ming-Hong Hao and Harold A Scheraga. Statistical thermodynamics of protein folding: Comparison of a mean-field theory with monte carlo simulations. *The Journal of chemical physics*, 102(3): 1334–1348, 1995.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025.

Alex Herbert and MJE Sternberg. Maxcluster: a tool for protein structure comparison and clustering, 2008.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Peter Holderrieth, Yilun Xu, and Tommi Jaakkola. Hamiltonian score matching and generative flows. *Advances in Neural Information Processing Systems*, 37:110464–110493, 2024.

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. In *NeurIPS*, 2024.

John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International conference on learning representations*, 2019.

John Ingraham, Max Baranov, Zak Costello, Karl Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana Lord, Christopher Ng-Thow-Hing, Erik Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, 2023.

Bowen Jing, Ezra Erives, Peter Pao-Huang, Gabriele Corso, Bonnie Berger, and Tommi Jaakkola. Eigenfold: Generative protein structure prediction with diffusion models. *arXiv preprint arXiv:2304.02198*, 2023.

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

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

Hassan K Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Patrick Kunzmann and Kay Hamacher. Biotite: a unifying open source computational biology framework in python. *BMC bioinformatics*, 19(1):346, 2018.

Gilles Labesse, N Colloc'h, Joël Pothier, and J-P Mornon. P-sea: a new efficient assignment of secondary structure from cα trace of proteins. *Bioinformatics*, 13(3):291–295, 1997.

Yeqing Lin and Mohammed AlQuraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds. *arXiv preprint arXiv:2301.12485*, 2023.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022:500902, 2022.

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 Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Tianyu Lu, Melissa Liu, Yilin Chen, Jinho Kim, and Po-Ssu Huang. Assessing generative model coverage of protein structures with shapes. *bioRxiv*, 2025.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Cyrus Neary and Ufuk Topcu. Compositional learning of dynamical system models using port-hamiltonian neural networks. In *Learning for Dynamics and Control Conference*, pages 679–691. PMLR, 2023.

José Nelson Onuchic, Zaida Luthey-Schulten, and Peter G Wolynes. Theory of protein folding: the energy landscape perspective. *Annual review of physical chemistry*, 48(1):545–600, 1997.

Carlos Outeiral, Daniel A Nissley, and Charlotte M Deane. Current structure predictors are not learning the physics of protein folding. *Bioinformatics*, 38(7):1881–1887, 2022.

Frank C Park and Roger W Brockett. Kinematic dexterity of robotic mechanisms. *The International Journal of Robotics Research*, 13(1):1–15, 1994.

Jerod Parsons, Bradley Holmes, Maurice Rojas, Jerry Tsai, and Charlie Strauss. Practical conversion from torsion space to cartesian space for in silico protein synthesis. *Journal of computational chemistry*, 26(10):1063–1068, 2005.

David Pollard. *A user's guide to measure theoretic probability*. Number 8. Cambridge University Press, 2002.

Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021.

Javier E Santos and Yen Ting Lin. Using ornstein-uhlenbeck process to understand denoising diffusion probabilistic model and its noise schedules. *arXiv preprint arXiv:2311.17673*, 2023.

Harold A Scheraga, Mey Khalili, and Adam Liwo. Protein-folding dynamics: overview of molecular simulation techniques. *Annu. Rev. Phys. Chem.*, 58(1):57–83, 2007.

Tim Schröder. The protein puzzle. *Max Planck Research*, 3, 2017.

Eugene Shakhnovich. Protein folding thermodynamics and dynamics: where physics, chemistry, and biology meet. *Chemical reviews*, 106(5):1559–1588, 2006.

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, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Maojiang Su, Jerry Yao-Chieh Hu, Yi-Chen Lee, Ning Zhu, Jui-Hui Chung, Shang Wu, Zhao Song, Minshuo Chen, and Han Liu. High-order flow matching: Unified framework and sharp statistical rates. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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.

Vladimir N Uversky. Natively unfolded proteins: a point where biology waits for physics. *Protein science*, 11(4):739–756, 2002.

Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic acids research*, 50(D1):D439–D444, 2022.

John Vastola. Generalization through variance: how noise shapes inductive biases in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7lUdo8Vuqa.

Joseph Watson, David Juergens, Nathaniel Bennett, Brian Trippe, Jason Yim, Helen Eisenach, Woody Ahern, Andrew Borst, Robert Ragotte, Lukas Milles, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 2023.

Kevin Wu, Kevin Yang, Rianne van den Berg, Sarah Alamdari, James Zou, Alex Lu, and Ava Amini. Protein structure generation via folding diffusion. *Nature communications*, 2024.

Fei Ye, Zaixiang Zheng, Dongyu Xue, Yuning Shen, Lihao Wang, Yiming Ma, Yan Wang, Xinyou Wang, Xiangxin Zhou, and Quanquan Gu. Proteinbench: A holistic evaluation of protein foundation models. *arXiv preprint arXiv:2409.06744*, 2024.

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 Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE(3) diffusion model with application to protein backbone generation. In *ICML*, 2023b.

Yichi Zhang, Yici Yan, Alex Schwing, and Zhizhen Zhao. Towards hierarchical rectified flow. *arXiv preprint arXiv:2502.17436*, 2025.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025.

A Short review of Lie and SE(3) groups

A.1 Lie Groups

Symmetries refer to transformations of an object that preserve certain structural properties. When these symmetries form a continuous set and are equipped with a composition operation that satisfies the group axioms, they define a Lie group. Formally, a group is a set G together with a binary operation $\circ : G \times G \to G$ that satisfies the following properties:

- Associativity: $(x \circ y) \circ z = x \circ (y \circ z), \forall\, x, y, z \in G$
- Identity: There exists an element $e \in G$, such that $x \circ e = e \circ x = x, \forall\, x \in G$
- Inverses: For every $x \in G$, there exists $x^{-1} \in G$, such that $x \circ x^{-1} = x^{-1} \circ x = e$

A Lie group extends this structure by also being a smooth manifold, meaning that the group operations—multiplication $(x, y) \to xy, \forall\, x, y \in G$ and inversion $x \to x^{-1}$ are smooth maps.

Given a $y \in G$, we define a diffeomorphism $L_y : G \to G, x \to yx$ known as left multiplication. Given a vector field X on the group G, we say that X is left-invariant if it remains unchanged under the pushforward induced by left multiplication. Formally, this means:

$$L_y^* X = X, \forall\, y \in G \tag{13}$$

where L_y^* denotes the differential (pushforward) of the left multiplication map. This pushforward naturally identifies the tangent spaces via:

$$\mathcal{T}_x G \xrightarrow{L_y^*} \mathcal{T}_{yx} G \tag{14}$$

As a consequence, a vector field is entirely determined by its value at the identity element $e \in G$ i.e. $\mathcal{T}_e$. We can further equip a real vector space V with a bilinear operation known as the Lie bracket $[\cdot, \cdot] : V \times V \to V$, which satisfies two key properties:

- Antisymmetry: $[x, y] = -[y, x],\ \forall\, x, y \in V$
- Jacobi identity: $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0,\ \forall\, x, y, z \in V$

A vector space V endowed with such a bracket is called a Lie algebra. For a Lie group G, the tangent space at the identity element $\mathcal{T}_e G$ naturally forms a Lie algebra. This Lie algebra is often denoted $\mathfrak{G}$. There exists a smooth, invertible map known as the exponential map $\exp : \mathfrak{G} \to G$, which maps elements of the Lie algebra to elements of the Lie group. Its inverse, when defined, is called the logarithmic map: $\log : G \to \mathfrak{G}$. These maps allow us to move between the Lie algebra and the Lie group, and in the case of matrix Lie groups, they coincide with the standard matrix exponential and logarithm.

Finally, we note that the set of all $n \times n$ non-singular (invertible) matrices forms a Lie group. The group operation is matrix multiplication, and this set can be viewed as a smooth manifold. This group is known as the *General Linear Group*, denoted $GL(n)$. Any closed subgroup of $GL(n)$ is referred to as a matrix Lie group, which is among the most widely studied and practically important classes of Lie groups. For matrix Lie groups, the exponential and logarithmic maps align with the matrix exponential and matrix logarithm, respectively. For a comprehensive introduction to Lie groups and their algebraic and geometric structures, we refer the reader to Hall (2013).

A.2 SE(3): Special Euclidean Group in 3 Dimensions

One of the most extensively studied closed subgroups of $GL(n)$ is the Special Orthogonal Group in three dimensions, denoted $\mathrm{SO}(3)$ This group consists of all 3×3 real rotation matrices, which preserve both lengths and orientations in three-dimensional space. More generally it can be defined as,

$$\mathrm{SO(n)} = \{\mathrm{R} \in GL(n) | \mathrm{R}^\top \mathrm{R} = \mathrm{I}, \det(\mathrm{R}) = 1\} \tag{15}$$

The condition $\mathrm{R}^\top \mathrm{R} = \mathrm{I}$ ensures that the transformation is orthogonal (i.e., distance-preserving), and $\det(\mathrm{R}) = 1$ guarantees that it is a proper rotation, excluding reflections. Moreover, translations in 3D space can also be represented as a matrix Lie group. A translation by a vector $s \in \mathbb{R}^3$ can be expressed using a homogeneous transformation matrix of the form:

$$\begin{bmatrix} \mathrm{I} & s \\ 0 & 1 \end{bmatrix} \tag{16}$$

The group operation corresponds to vector addition of the translation components: $s_1 + s_2$. This group of translations forms a matrix Lie group isomorphic to $(\mathbb{R}^3, +)$ the additive group of 3D vectors. Combining both the $\mathrm{SO}(3)$ and $\mathbb{R}^3$ groups, we can represent rigid transformations of objects in 3D space. This group is known as the Special Euclidean group in three dimensions, denoted $\mathrm{SE}(3)$, which can further decomposed as $\mathrm{SE}(3) \cong \mathrm{SO}(3) \ltimes (\mathbb{R}^3, +)$ under certain conditions. An element of $\mathrm{SE}(3)$ can be represented as a homogeneous transformation matrix:

$$\mathrm{SE}(3) = \left\{ (r, s) = \begin{pmatrix} r & s \\ 0 & 1 \end{pmatrix} : r \in \mathrm{SO}(3), s \in (\mathbb{R}^3, +) \right\} \tag{17}$$

Under a specific choice of Riemannian metric (Park and Brockett, 1994), the inner product on the Lie algebra $\mathfrak{se}(3)$ can be decomposed as $\langle \mathfrak{x}, \mathfrak{x}' \rangle_{\mathrm{SE}(3)} = \langle r, r' \rangle_{\mathrm{SO}(3)} + \langle s, s' \rangle_{\mathbb{R}^3}$, which forms the foundation of our model.

B Additional Experiment Details

B.1 Data

We train StericFlow on monomers of length 60 to 512 and resolution better than $5\mathring{\mathrm{A}}$ obtained from the Protein Data Bank (PDB) (Berman et al., 2000). To ensure higher structural quality, we filtered the dataset to retain proteins with substantial secondary structure content. Similar to Yim et al. (2023b); Bose et al. (2024), we applied DSSP (Kabsch and Sander, 1983) and excluded monomers with more than 50% loop content, yielding a final dataset of 24,003 proteins belonging to 4532 clusters based on 40% sequence identity. The distribution of sequence lengths of protein ℓ_{prot} in this dataset is shown in Figure 5.

B.2 Benchmarking Metrics

Designability. A protein backbone is considered designable if there exists an amino acid sequence. We evaluate designability following the protocol of Yim et al. (2023b). For each backbone generated by a model, we sample sequences using ProteinMPNN (Dauparas et al., 2022) with a temperature of 0.1. Each sequence is then folded with ESMFold (Lin et al., 2023), and the resulting structure is compared against the original backbone using root mean square deviation (RMSD). A sample is classified as designable if its minimum RMSD—referred to as the self-consistency RMSD (scRMSD)—is below $2\mathring{\mathrm{A}}$. The overall designability score of a model is defined as the fraction of generated backbones that satisfy this criteria.

Diversity (TM-score & Cluster). We evaluate diversity using the measures described in Bose et al. (2024). We compute the average pairwise TM-score among designable samples for each protein length, and then aggregate these averages across all lengths. Since TM-scores range from 0 to 1, with higher values reflecting greater structural similarity, lower scores indicate higher diversity. The second metric is described in Huguet et al. (2024) which amounts to the number of generated clusters with a TM-score threshold of 0.5 (Herbert and Sternberg, 2008).

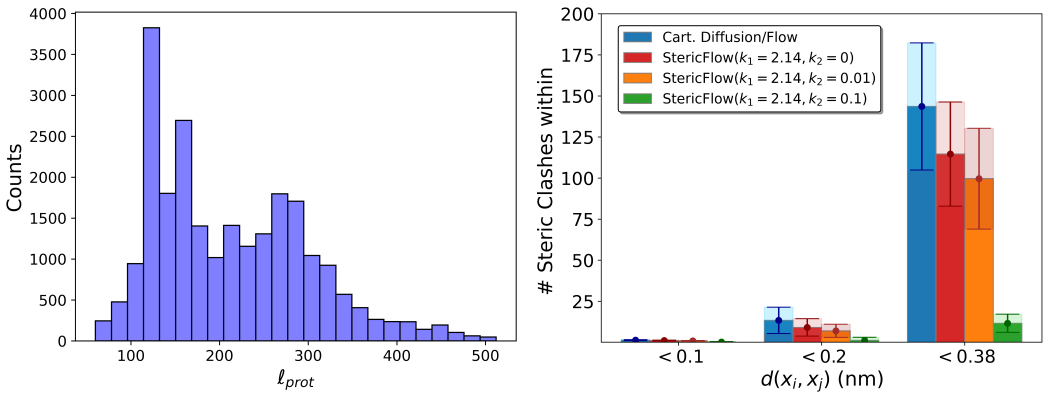


Figure 5: (*left*) Distribution of the sequence length of protein ℓ_{prot} in the obtained monomer dataset, (*right*) Average number of residue–residue steric collisions (defined as pairwise residue distance $< d(\mathbf{x}_i, \mathbf{x}_j)$(nm)) observed across the forward trajectory of a protein of length 116, for Cartesian diffusion/flow baselines and variants of our StericFlow method with different weightings (k_1, k_2). We observe that the number of collisions decreases as the weight on Coulombic repulsion (k_2) increases within the StericFlow framework.

Novelty (avg $\max$ **TM).** We quantify novelty using two metrics, following Bose et al. (2024). The first computes, for each designable generated protein, the maximum TM-score with respect to the training data. We then report the average of these maximum values across all designable samples. Lower scores indicate greater novelty.

Novelty (TM-Frac). The second metric, introduced in Lin and AlQuraishi (2023), reports the fraction of generated proteins that are both designable and novel, i.e., whose maximum TM-score against the training set falls below a given threshold.

Secondary Structure content. We analyze the secondary structure composition of designable backbones using Biotite's (Kunzmann and Hamacher, 2018) implementation of the P-SEA algorithm (Labesse et al., 1997). For each sample, we compute the proportions of α-helices, β-sheets, and coils. These are reported as normalized fractions:

$$\frac{\alpha}{\alpha+\beta+c}, \quad \frac{\beta}{\alpha+\beta+c}, \quad \frac{c}{\alpha+\beta+c}$$

corresponding to helices, sheets, and coils, respectively.

B.3 Sequence Conditioned Monomer Folding

We partition the dataset into a $0.9 : 0.1$ train–test split, selecting clusters randomly. Out of a total of 4,542 clusters, 4,087 are assigned to the training set and 455 to the test set. Figure 6 shows the distributions of protein sequence lengths in both sets, indicating that they are well-matched. Models are trained exclusively on the training set and evaluated on the test set to assess their performance in predicting folded structures.

B.4 Neural Network Hyperparameters

The training setup is summarized in table 6, with network hyperparameters listed in table 5. We adopt the "length batching" scheme from (Yim et al., 2023b), where each batch contains the same protein sampled at different time steps. The batch size varies approximately as num_residues$^2/M$, with M as a hyperparameter. StericFlow is implemented in PyTorch and trained on a single H200 140GB NVIDIA GPU.

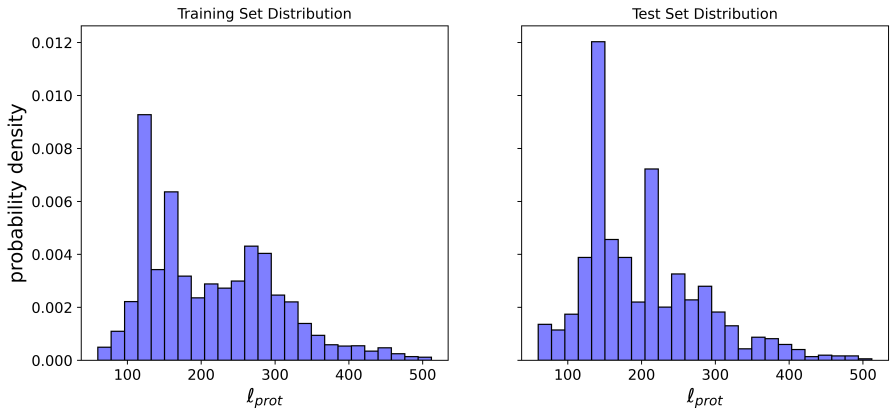


Figure 6: Training and Testing distribution for length of protein ℓ_{prot}.

Table 5: Default hyperparameters for StericFlow Model.

Module	Hyperparameter	Meaning	Value
Structural Encoder	Node embedding size	Node embedding dimension	256
	Edge embedding size	Edge embedding dimension	128
	Hidden dimension	Hidden dimension	256
	Heads	Number of heads in the transformer	8
	Blocks	Number of transformer layers	4
Structural Decoder	Node embedding size	Node embedding dimension	256
	Edge embedding size	Edge embedding dimension	128
	Hidden dimension	Hidden dimension	256
	Heads	Number of heads in the transformer	8
	Blocks	Number of transformer layers	4

Table 6: Training parameters for StericFlow Model.

Parameter	Value
Optimizer	Adam
Learning Rate	0.0001
$\beta_1, \beta_2, \epsilon$	0.9, 0.999, 1e-8
Effective M (max squared residues per batch)	500k
Sequence masking probability	50%
Minimum number of residues	60
Maximum number of residues	512

Table 7: Simulation parameters for StericFlow Model.

Hyper-parameter	Value
γ	$0.01\ell_{prot}$
σ_β^2	0.1
μ_β	$[2.26, 3.14, -2.44, 1.095, 0.98, 1.20]$
σ_v^2	0.1

C Additional Results

C.1 Effect of varying k_1, k_2 energy weighting.

We conducted an ablation study to assess the effect of varying the energy weightings k_1, k_2 that govern the forward dynamics of our model. As shown in fig. 2, different weighting schemes yield distinct forward processes, with a different number of steric clashes. Within the StericFlow framework, we find that increasing the weight on Coulombic repulsion (k_2) reduces the number of clashes.

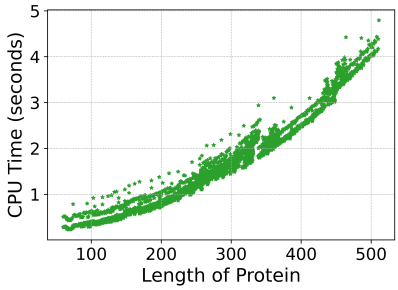


Figure 7: Runtime Complexity for forward simulation.

C.2 Computational Complexity

We evaluate the runtime complexity of our forward simulation (see section 4.2) as a function of protein length. Figure 7 shows the scaling behavior in terms of CPU time (seconds). As expected, longer proteins require more time for forward simulation. While longer proteins take more time, overall runtime remains modest, and parallelization or multi-processing can further accelerate the simulation, ensuring it is not a computational bottleneck

C.3 Inference speed and Trainable Parameters

We compare the model size and inference speed of StericFlow against prior methods in the table below. Inference time is measured for generating a protein structure of length 100 on a GPU over 100 timesteps of the reverse diffusion process. Note that the large parameter count (650M) corresponds to the pretrained ESM language model used in FoldFlow-2 and StericFlow. As shown, StericFlow

Table 8: Inference speed and Trainable Parameters.

Method	Model size	Trainable params	Inference time (sec.)
FrameDiff	$\sim$ 20M	$\sim$ 18M	$\sim$ 4.6
FoldFlow	$\sim$ 20M	$\sim$ 19M	$\sim$ 4.7
FoldFlow-2	$\sim$ 672M	$\sim$ 21M	$\sim$ 5.1
StericFlow	$\sim$ 670M	$\sim$ 20M	$\sim$ 5.1

achieves a comparable inference speed to previous methods while maintaining a similar number of trainable parameters. This efficiency is notable given the inclusion of the large pretrained language model, which performs substantial computations.

C.4 Clashes analysis in reverse process.

We conducted an additional analysis tracking steric clashes throughout the trajectory. Clashes are defined as pairwise residue contacts with pairwise Euclidean distance $d(x_i, x_j) < 0.38$ and are averaged over 5 protein structures for each length in $\{100, 150, 200\}$.

The results are shown in the Fig. 8 below demonstrate that StericFlow consistently produces fewer clashes than a vanilla diffusion/flow baseline across all tested lengths and throughout the entire reverse process — not only in the final generated structures. It indicates that the physically grounded forward process, which explicitly incorporates Coulomb repulsion to discourage steric overlaps during unfolding.

C.5 Effect of varying time steps.

We evaluate the impact of the number of time steps used in solving the reverse process on the designability and novelty. table 9 show that StericFlow achieves strong performance even with relatively few steps, while maintaining stability as the number of steps increases.

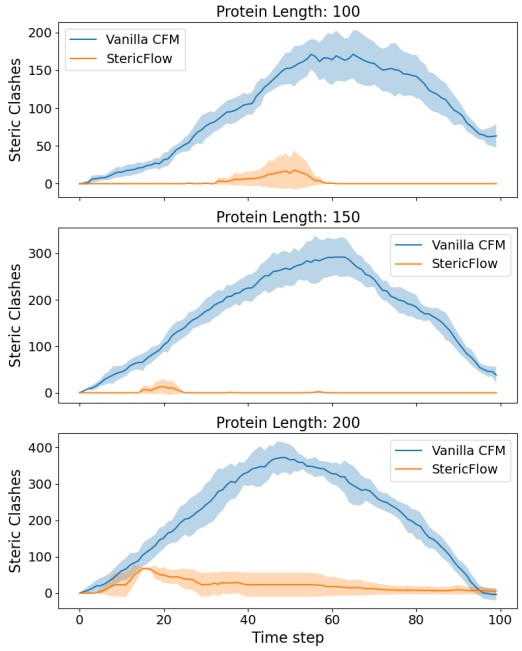


Figure 8: Time-series analysis of steric clashes based on the time-steps during generative process going from T=100 to T=0.

Table 9: Ablation study showing the effect of varying the number of time steps during inference on designability and novelty metrics.

Time-steps	Frac. $< 2\text{Å}$ (↑)	TM-Frac. < 0.3 (↑)
50	0.805	0.201 ± 0.024
100	0.817	0.197 ± 0.021
200	0.811	0.199 ± 0.011

C.6 Different structure predictor models

We conducted an ablation replacing ESMFold with AlphaFold2 as the folding oracle, sampling 50 protein structures of lengths in the range $50 - 250$ residues. The results are shown in the table below. We observe that we obtain the same level of designability even using AlphaFold2 as the oracle method.

Table 10: Ablation on structure predictor model.

Method	Designability
AlphaFold2	0.96
ESMFold	0.97

Table 11: Raw validity metrics before inverse folding.

Method	% $d(x_i, x_j) < 0.38$ nm
FrameDiff	1.3
FoldFlow	0.5
StericFlow	**0.03**

C.7 SENSITIVITY OF k_1, k_2, AND γ IN GENERATED SAMPLES.

The k_1, k_2, and γ govern the forward process in our method, akin to the scheduler choice in CFM. We have conducted an ablation study with different values of k_1, k_2, and γ and evaluated the designability of generated structures by sampling 50 protein structures of lengths in 50-250 residues.

The results are shown in the table below. We observe that the generated structures are sensitive to these hyperparameters, as they define the intermediate trajectory dynamics.

(k_1, k_2, γ)	Designability
$(0.01\ell_{\text{prot}}, 0.01, 0.01\ell_{\text{prot}})$	0.90
$(0.02\ell_{\text{prot}}, 0.02, 0.02\ell_{\text{prot}})$	0.96

D ANGULAR AND FRAME REPRESENTATION

D.1 ANGULAR REPRESENTATION

A protein backbone structure can be fully described using nine parameters per residue: three bond lengths, three bond angles, and three dihedral torsion angles. Under the common assumption of idealized bond lengths, only six angular parameters remain necessary to represent the internal structure. These angles are summarized in table 12. Together, the nine parameters provide a lossless mapping between Cartesian and internal (angular) representations of the protein backbone.

Recall that given four atoms a, b, c, d with positions $\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d$, the dihedral angle is defined as

$$\text{Dihedral}(a, b, c, d) = \text{atan2}\left(\hat{b}_2 \cdot (\hat{n}_1 \times \hat{n}_2), \hat{n}_1 \cdot \hat{n}_2\right) \tag{18}$$

where

$$\hat{b}_2 = \frac{\mathbf{x}_c - \mathbf{x}_b}{\|\mathbf{x}_c - \mathbf{x}_b\|}, \quad \hat{n}_1 = \frac{(\mathbf{x}_b - \mathbf{x}_a) \times (\mathbf{x}_c - \mathbf{x}_b)}{\|\mathbf{x}_b - \mathbf{x}_a\|\|\mathbf{x}_c - \mathbf{x}_b\|}, \quad \hat{n}_2 = \frac{(\mathbf{x}_c - \mathbf{x}_b) \times (\mathbf{x}_d - \mathbf{x}_c)}{\|\mathbf{x}_c - \mathbf{x}_b\|\|\mathbf{x}_d - \mathbf{x}_c\|} \tag{19}$$

Similarly, given three atoms a, b, c, the bond angle is defined as

$$\text{Angle}(a, b, c) = \arccos\left(\frac{(\mathbf{x}_a - \mathbf{x}_b) \cdot (\mathbf{x}_c - \mathbf{x}_b)}{\|\mathbf{x}_a - \mathbf{x}_b\|\|\mathbf{x}_c - \mathbf{x}_b\|}\right) \tag{20}$$

Using these definitions, the internal coordinates of the i-th residue are obtained as

$$\psi_i = \text{Dihedral}\left((\mathbf{x}_{\text{N}})_i, (\mathbf{x}_{\text{C}_\alpha})_i, (\mathbf{x}_{\text{C}})_i, (\mathbf{x}_{\text{N}})_{i+1}\right) \tag{21}$$

$$\phi_i = \text{Dihedral}\left((\mathbf{x}_{\text{C}})_i, (\mathbf{x}_{\text{N}})_{i+1}, (\mathbf{x}_{\text{C}_\alpha})_{i+1}, (\mathbf{x}_{\text{C}})_{i+1}\right) \tag{22}$$

$$\omega_i = \text{Dihedral}\left((\mathbf{x}_{\text{C}_\alpha})_i, (\mathbf{x}_{\text{C}})_{i+1}, (\mathbf{x}_{\text{N}})_{i+1}, (\mathbf{x}_{\text{C}_\alpha})_{i+1}\right) \tag{23}$$

$$(\theta_1)_i = \text{Angle}\left((\mathbf{x}_{\text{N}})_i, (\mathbf{x}_{\text{C}_\alpha})_i, (\mathbf{x}_{\text{C}})_i\right) \tag{24}$$

$$(\theta_2)_i = \text{Angle}\left((\mathbf{x}_{\text{C}_\alpha})_i, (\mathbf{x}_{\text{C}})_i, (\mathbf{x}_{\text{N}})_{i+1}\right) \tag{25}$$

$$(\theta_3)_i = \text{Angle}\left((\mathbf{x}_{\text{C}})_i, (\mathbf{x}_{\text{N}})_{i+1}, (\mathbf{x}_{\text{C}_\alpha})_{i+1}\right) \tag{26}$$

Table 12: Angular representation.

Angle	Description
ϕ	Dihedral $C_i N_{i+1} C_{\alpha_{i+1}} C_{i+1}$
ψ	Dihedral $N_i C_{\alpha_i} C_i N_{i+1}$
ω	Dihedral $C_{\alpha_i} C_i N_{i+1} C_{\alpha_{i+1}}$
θ_1	Bond angle $\angle N_i C_{\alpha_i} C_i$
θ_2	Bond angle $\angle C_{\alpha_i} C_i N_{i+1}$
θ_3	Bond angle $\angle C_i N_{i+1} C_{\alpha_{i+1}}$

D.2 Forward Unfolding process in Cartesian Coordinates

Given a forward-simulated angular trajectory $\mathbf{u}_{0:T}$ as described in Section 4.2, the corresponding Cartesian flow can be obtained via nerf as $\mathbf{x}_{0:T}, \mathbf{v}_{0:T} = \text{nerf}(\mathbf{u}_{0:T})$, where computing velocities requires a Jacobian-based mapping between angular and Cartesian domains.

Alternatively, one can first transform the angular trajectory $\mathbf{z}_{0:T}$ into Cartesian space as $\mathbf{x}_{0:T} = \text{nerf}(\mathbf{z}_{0:T})$, and then compute velocities v_t using finite differences (forward or backward) or cubic splines[3] . Moreover, a flow can also be defined directly in Cartesian coordinates by transforming the unfolding equations through nerf, as discussed above.

We can also derive an approximate equivalent framework for unfolding in Cartesian coordinates (centered by the geometric mean to ensure translation invariance), as described in Section 4.2. These unfolding equations operate directly in Cartesian space, incorporating Coulombic repulsion and an attractive potential toward the target state. This formulation provides greater flexibility in simulating the forward process.

Formally, the process can be expressed as a second-order Decay–Hamiltonian system (Neary and Topcu, 2023; Hamilton, 1834) evolving over Cartesian space:

$$\text{unfolding:} \qquad \begin{cases} \dot{\mathbf{x}}_t &= \mathbf{v}_t \\ \dot{\mathbf{v}}_t &= f_{\text{unfold}}(\mathbf{x}_t, \mathbf{v}_t) = -\underbrace{\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)}_{\text{potential force}} - \underbrace{K(\mathbf{v}_t)}_{\text{drag force}}, \end{cases} \tag{27}$$

where $U(\mathbf{x}_t)$ corresponds to an Ornstein–Uhlenbeck energy potential, and $K(\mathbf{v}_t)$ denotes a frictional drag term. We can also write the combined potential in terms of Cartesian coordinates as,

$$U(\mathbf{x}_t) = k_1 U_{\text{target}}(\mathbf{x}_t) + k_2 U_{\text{repulsion}}(\mathbf{x}_t) \tag{28}$$

that incorporates both Ornstein-Uhlenbeck attraction to $\mathbf{x}_{\text{target}}$ and Coulomb-like repulsion

$$U_{\text{target}}(\mathbf{x}_t) = \frac{1}{2} \sum_i \left(\mathbf{x}_{i,t} - \mathbf{x}_{i,\text{target}}\right)^2, \quad U_{\text{repulsion}}(\mathbf{x}_t) = \frac{1}{2} \sum_{i \neq j} \frac{1}{||\mathbf{x}_{i,t} - \mathbf{x}_{j,t}||}, \tag{29}$$

Here, i, j denotes the index of backbone residues, $\mathbf{x}_{\text{target}} \sim p_0\big(\text{nerf}(\mathbf{z}_0)\big)$ with $\mathbf{z}_0 \sim p_0(\mathbf{z})$, and k_1, k_2 are potential weights. The target energy is a quadratic potential attracting the protein to the target state, while Coulombic repulsion (based on Euclidean distance) prevents collisions. Gradients $\nabla_{\mathbf{x}_t} U(\mathbf{x}_t)$ are computed via JAX autodiff.

To curb the oscillatory behavior from the potential, we utilize a linear drag force, $K(\mathbf{v}_t) = \gamma \mathbf{v}_t$, where $\gamma > 0$ is the drag coefficient. This damping term counteracts large accelerations and stabilizes the dynamics, ensuring convergence to $\mathbf{x}_{\text{target}}$ as $t \to \infty$.

We parametrize the final target state to be a secondary structure linear chain predefined as linear β-sheets. Denoting $\mathbf{u} = [\mathbf{x}, \mathbf{v}]$, the terminal distribution is defined as:

$$p_0(\mathbf{u}) = \underbrace{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\beta_x}, \sigma^2_{\beta_x} I)}_{\text{position } p_0(\mathbf{z})} \cdot \underbrace{\mathcal{N}(\mathbf{v}|\mathbf{0}, \sigma_v^2 I)}_{\text{velocity}}, \tag{30}$$

where $\boldsymbol{\mu}_{\beta_x} = \text{nerf}(\boldsymbol{\mu}_\beta)$, with a small variation $\sigma^2_{\beta_x}$, around the coordinates computed from the β-sheet angles μ_β. We also assume a data distribution

$$p_1(\mathbf{u}) = p_{\text{data}}(\mathbf{x}) \cdot \mathcal{N}(\mathbf{v}|\mathbf{0}, \sigma_v^2 I), \tag{31}$$

from which we have observations without velocities, and we augment the data with Gaussian velocities of variance σ_v^2.

Now we can follow the same methodology to compute forward simulations as described in Section 4.2 as,

$$p(\mathbf{u}_t|\mathbf{u}_1, \mathbf{u}_0) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_t \\ \mathbf{v}_t\end{bmatrix}; \begin{bmatrix}\mu_{\mathbf{x}_t} \\ \mu_{\mathbf{v}_t}\end{bmatrix}, \begin{bmatrix}\sigma_x^2 \\ \sigma_v^2\end{bmatrix} \mathrm{I}\right), \quad \begin{bmatrix}\mu_{\mathbf{x}_t} \\ \mu_{\mathbf{v}_t}\end{bmatrix} = \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{v}_1\end{bmatrix} - \int_t^1 \begin{bmatrix}\dot{\mathbf{x}}_t \\ \dot{\mathbf{v}}_t\end{bmatrix} d\tau \tag{32}$$

[3] https://github.com/patrick-kidger/torchcubicspline

which provides us with the evolution of Cartesian coordinates and the corresponding velocities, which can be used to train the model with conditional flow matching. Below, describes a code snippet for the simulation,

```
import numpy as np
import jax
import jax.numpy as jnp
from jax.tree_util import Partial
from functools import partial

def cdist2(X):
  r2 = jnp.sum( (X[:,None,:] - X[None,:,:])**2, axis=-1)
  return r2

def cdist(X):
  r2 = cdist2(X)
  I = jnp.eye(X.shape[0])
  r = jnp.sqrt(r2 + I) - I  # avoid sqrt(0) at diagonal, leads to grad=
    nan
  return r

def energy_attractor( theta,final_state,k=1 ):
    ener = jnp.sum((theta  - final_state)**2)
    return ener

def energy_coulomb( X ):
    N = X.shape[0]
    ids = 1 / ( cdist(X) + jnp.eye(N) )
    ener = jnp.sum(ids.sort(axis=-1)[:,:])
    return ener

@jax.jit
def energy_total(X, final_state,k1=1,k2=1):
    e_angle = energy_attractor(X,final_state)
    e_repulsion = energy_coulomb(X)
    return k1*e_angle + k2*e_repulsion

### Exemplar usage to simulate for a protein
cart_coords = X     #protein cartesian coords (e.g. C_alpha coordinates)
target = X_target   #target cartesian coords
Nt = 100            #Number of time-steps for forward simulation
dt = 1/Nt
drag_const = gamma   #Drag force gamma constant
k1 = k1              #weighting for attractor force
k2 = k2              #weighting for repulsive force
#JAX Autodiff to compute grad
du = jax.grad( partial(energy_total, final_state=X_target,k1=k1,k2=k2) )
#Initialization for the system
H_cart = np.zeros((Nt,X.shape[0],X.shape[1]))
H_cart[0] = X
H_vel = np.random.normal(H_cart) #normal intialization or zero
    initialization

for i in range(1, Nt):
    H_cart[i] = H_cart[i-1] + H_vel[i-1] * dt
    force = du(H_cart[i-1])
    H_vel[i] = H_vel[i-1] - force * dt - drag_const*H_vel[i-1]* dt
```

D.3 Proof of Proposition 1

To prove the statement, it suffices to show that under the translation of the Cartesian coordinates of a protein, the angular representation remains invariant.

Let $\mathbf{x} \in \mathbb{R}^{3N}$ denote the Cartesian coordinates of a protein with N residues, and let $\mathbf{z}(\mathbf{x})$ denote its angular representation. For the i-th residue, we denote

$$\mathbf{x}_i = [(\mathbf{x}_{\mathrm{N}})_i, (\mathbf{x}_{\mathrm{C}})_i, (\mathbf{x}_{\mathrm{C}_\alpha})_i]$$

as the Cartesian coordinates of the backbone atoms (Nitrogen, Carbon, and α-Carbon, respectively). The corresponding angular representation is given as

$$\mathbf{z}(\mathbf{x}_i) = \left[\phi_i, \psi_i, \omega_i, (\theta_1)_i, (\theta_2)_i, (\theta_3)_i\right] \tag{33}$$

where ϕ_i, ψ_i, ω_i are the backbone torsion angles and $(\theta_1)_i, (\theta_2)_i, (\theta_3)_i$ are the bond angles. Under the action of the translation operator $\mathcal{T}_{\mathbf{t}}$ on $\mathbf{x}_i$, the translated Cartesian coordinates can be represented as $\mathbf{x}'_i = \mathbf{x}_i + \mathbf{t}$, i.e.,

$$\mathbf{x}'_i = [(\mathbf{x}'_{\mathrm{N}})_i, (\mathbf{x}'_{\mathrm{C}})_i, (\mathbf{x}'_{\mathrm{C}_\alpha})_i] \tag{34}$$

Recall the definitions of the bond angle and dihedral angle in Eq. 20 and Eq. 18. Both are computed from relative distances, which remain invariant under global translations of the Cartesian coordinates. For instance,

$$(\mathbf{x}'_{\mathrm{N}})_i - (\mathbf{x}'_{\mathrm{C}})_i = (\mathbf{x}_{\mathrm{N}})_i - (\mathbf{x}_{\mathrm{C}})_i, \quad (\mathbf{x}'_{\mathrm{C}_\alpha})_i - (\mathbf{x}'_{\mathrm{C}})_i = (\mathbf{x}_{\mathrm{C}_\alpha})_i - (\mathbf{x}_{\mathrm{C}})_i \tag{35}$$

which implies that the bond and dihedral angles remain unchanged under translation. Formally, this leads to having the same angular representation for both translated and original coordinates.

$$\mathbf{z}(\mathbf{x}_i) = \mathbf{z}(\mathbf{x}'_i) \tag{36}$$

D.4 Proof of Proposition 2

The forward process is governed by a Hamiltonian $\mathcal{H}$ consisting of two components: a kinetic energy term defined by the velocity of each particle, and a potential energy $V(\mathbf{z})$ comprising a harmonic attractive term and an inverse-distance repulsion term. It can be written as,

$$\mathcal{H}(\mathbf{z}, \mathbf{v}) = \text{Kinetic Energy} + \text{Potential Energy} = \sum_i \frac{1}{2}\mathbf{v}_i^2 + V(\mathbf{z}), \tag{37}$$

$$V(\mathbf{z}) = k_1 \sum_i ||\mathbf{z}_i - \mathbf{z}_{i,\text{target}}||^2 + k_2 \sum_{i \neq j} \frac{1}{||\mathbf{x}_i - \mathbf{x}_j||} \tag{38}$$

where $\mathbf{x} = \mathrm{nerf}(\mathbf{z})$. The system evolves under this potential, subject to frictional (damping) resistance characterized by $\gamma > 0$, as described in the manuscript. Assuming unit mass, the damped equation of motion is given by:

$$\dot{\mathbf{v}} = -\gamma \mathbf{v} - \nabla V(\mathbf{z}) \tag{39}$$

$$\dot{\mathbf{z}} = \mathbf{v} \tag{40}$$

Now, differentiating $\mathcal{H}$ with time and substituting Eq. 39–40, we get,

$$\dot{\mathcal{H}}(\mathbf{z}, \mathbf{v}) = \sum_i \mathbf{v}_i \cdot \dot{\mathbf{v}}_i + \sum_i \nabla_{\mathbf{z}_i} V(\mathbf{z}) \cdot \dot{\mathbf{z}}_i \tag{41}$$

$$= \sum_i \mathbf{v}_i \cdot \big(-\gamma \mathbf{v}_i - \nabla_{\mathbf{z}_i} V(\mathbf{z})\big) + \sum_i \nabla_{\mathbf{z}_i} V(\mathbf{z}) \cdot \mathbf{v}_i = -\gamma \sum_i ||\mathbf{v}_i||^2 \leq 0 \tag{42}$$

where the potential terms cancel, leaving only the dissipation. This shows that $\mathcal{H}$ decreases along trajectories, with $\dot{\mathcal{H}} = 0$ precisely when $\mathbf{v}_i = 0$ for all i. Therefore, $\mathcal{H}$ is a valid Lyapunov function: it is bounded below and strictly decreasing whenever the system is in motion.

Part (i): convergence to a critical point. Since $\mathcal{H}$ is monotonically non-increasing and bounded below, by the monotone convergence theorem it converges to a finite limit,

$$\lim_{t \to \infty} \mathcal{H}(\mathbf{z}(t), \mathbf{v}(t)) = \mathcal{H}^* < \infty. \tag{43}$$

Integrating $\dot{\mathcal{H}}$ over $t \in [0, \infty)$ gives,

$$\int_0^\infty \gamma \sum_i ||\mathbf{v}_i(t)||^2 \, dt = \mathcal{H}(0) - \mathcal{H}^* < \infty \tag{44}$$

where $\mathcal{H}(0)$ is the Hamiltonian at $t = 0$. Hence the total kinetic energy dissipated over all time is finite and $\|\mathbf{v}(t)\|^2$ is integrable. Moreover, since $\mathcal{H}$ is non-increasing, the sublevel set $\{\mathcal{H} \leq \mathcal{H}(0)\}$ is forward-invariant, and finiteness of V on this set bounds the pairwise separations $\|\mathbf{x}_i - \mathbf{x}_j\|$ away from zero: trajectories never approach the repulsion singularity. Consequently ∇V is bounded along trajectories, so by Eq. 39 $\dot{\mathbf{v}}$ is bounded and $\|\mathbf{v}(t)\|^2$ is uniformly continuous. Barbalat's Lemma (Khalil, 2002) then yields

$$||\mathbf{v}(t)|| \to 0 \text{ as } t \to \infty. \tag{45}$$

Since $\mathbf{v}(t) \to 0$ and therefore $\dot{\mathbf{v}} \to 0$, Eq. 39 gives $\nabla_{\mathbf{z}} V(\mathbf{z}) \to 0$. The system thus approaches a critical point $\mathbf{z}^*$ of V, characterized by

$$\nabla_{\mathbf{z}_i} V(\mathbf{z}^*) = 2k_1(\mathbf{z}_i^* - \mathbf{z}_{i,\text{target}}) - k_2 \sum_{j \neq i} \frac{\mathbf{x}_i^* - \mathbf{x}_j^*}{||\mathbf{x}_i^* - \mathbf{x}_j^*||^3} = 0 \qquad \forall\, i, \tag{46}$$

which establishes part (i).

Part (iii): location of the critical point. The potential is dominated by the harmonic term, i.e. $k_1 \gg k_2$, so we treat the repulsion as a perturbation. Write $\mathbf{z}_i^* = \mathbf{z}_{i,\text{target}} + \delta_i$, and correspondingly $\mathbf{x}_i^* = \mathbf{x}_{i,\text{target}} + \mathcal{O}(\|\delta\|)$ through nerf. Expanding the repulsion gradient in Eq. 46 about the target configuration, the leading-order balance is

$$2k_1\delta_i = k_2 \sum_{j \neq i} \frac{\mathbf{x}_{i,\text{target}} - \mathbf{x}_{j,\text{target}}}{||\mathbf{x}_{i,\text{target}} - \mathbf{x}_{j,\text{target}}||^3} + \mathcal{O}\big(k_2\|\delta\|\big). \tag{47}$$

Because the terminal configuration is an extended chain, its pairwise separations are bounded away from zero, so the sum on the right is bounded by a constant C depending only on $\mathbf{z}_{\text{target}}$ and the chain length N. Hence $\|\delta_i\| \leq Ck_2/(2k_1) + \mathcal{O}(k_2\|\delta\|/k_1)$, and for $k_1 \gg k_2$ the second term is negligible, giving

$$\mathbf{z}_i^* = \mathbf{z}_{i,\text{target}} + \mathcal{O}\left(\frac{k_2}{k_1}\right) \approx \mathbf{z}_{i,\text{target}}. \tag{48}$$

Thus the steady state lies in an $\mathcal{O}(k_2/k_1)$ neighborhood of $\mathbf{z}_{\text{target}}$, establishing part (iii).

Part (ii): exponential rate. Parts (i) and (iii) establish convergence and its location, but not the rate. To obtain part (ii), linearize Eq. 39-40 about the equilibrium $(\mathbf{z}^*, \mathbf{0})$. Writing $\mathbf{e} = \mathbf{z} - \mathbf{z}^*$ and $\mathbf{H} = \nabla^2 V(\mathbf{z}^*)$, the linearized system is

$$\ddot{\mathbf{e}} = -\gamma\dot{\mathbf{e}} - \mathbf{H}\mathbf{e}, \qquad \text{equivalently} \qquad \frac{d}{dt}\begin{bmatrix}\mathbf{e}\\ \dot{\mathbf{e}}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{0} & \mathrm{I}\\ -\mathbf{H} & -\gamma\mathrm{I}\end{bmatrix}}_{\mathbf{A}}\begin{bmatrix}\mathbf{e}\\ \dot{\mathbf{e}}\end{bmatrix}. \tag{49}$$

Since V is dominated by the harmonic term for $k_1 \gg k_2$, $\mathbf{H}$ is symmetric positive definite with eigenvalues $\{\omega_m^2\}$ satisfying $\omega_m^2 = 2k_1 + \mathcal{O}(k_2)$; in particular $\mathbf{z}^*$ is a strict local minimum rather than a saddle. Diagonalizing $\mathbf{H}$ decouples the system into scalar damped oscillators $\ddot{e}_m = -\gamma\dot{e}_m - \omega_m^2 e_m$, with characteristic roots

$$\mu_m^{\pm} = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\omega_m^2}}{2}. \tag{50}$$

Every root has negative real part: for $\gamma^2 < 4\omega_m^2$ the roots are complex with real part $-\gamma/2$, and for $\gamma^2 \geq 4\omega_m^2$ they are real and negative. Hence $\mathbf{A}$ is Hurwitz, and setting

$$\lambda = \min_m \left|\mathrm{Re}(\mu_m^{\pm})\right| = \begin{cases}\gamma/2, & \gamma^2 < 4\omega_{\min}^2,\\ \frac{1}{2}\big(\gamma - \sqrt{\gamma^2 - 4\omega_{\min}^2}\big), & \gamma^2 \geq 4\omega_{\min}^2,\end{cases} \tag{51}$$

there exists $c > 0$ with $\|\mathbf{z}_t - \mathbf{z}^*\| \leq c\, e^{-\lambda t}$ for trajectories in a neighborhood of $\mathbf{z}^*$, which the Lyapunov argument of part (i) guarantees they enter. The rate therefore depends only on the damping γ and the curvature of V at $\mathbf{z}^*$.

D.5 Frame Representation

We continue from section 2 to describe the frame representation used by AF2 derived from Engh and Huber (2006) in more detail and follow the same definitions as described in Yim et al. (2023b). As discussed, $\mathrm{N}^*, \mathrm{C}_\alpha^*, \mathrm{C}^*, \mathrm{O}^*$ represents the idealized atom coordinates that assume chemically ideal bond lengths and angles. For unconditional generation, we take the idealized values of Alanine which are,

$$\mathrm{N}^* = (-0.525, 1.363, 0.0) \tag{52}$$
$$\mathrm{C}_\alpha^* = (0.0, 0.0, 0.0) \tag{53}$$
$$\mathrm{C}^* = (1.526, 0.0, 0.0) \tag{54}$$
$$\mathrm{O}^* = (0.627, 1.062, 0.0) \tag{55}$$

The backbone oxygen atom requires a torsional angle φ_n to determine its coordinates:

$$\mathrm{O}_n = T_n \cdot T(\varphi_n) \cdot \mathrm{O}^*, \tag{56}$$

where φ_n is the torsional angle for residue n and

$$T^*(\varphi_n) = (\mathbf{r}_x(\varphi_n), \mathbf{x}_\varphi) \tag{57}$$

is the transformation frame following the methodology described in (Yim et al., 2023b). Here, φ_n is a tuple specifying a point on the unit circle, $\varphi_n = [\varphi_{n,1}, \varphi_{n,2}]$, and

$$\mathbf{r}_x(\varphi_n) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \varphi_{n,1} & -\varphi_{n,2} \\ 0 & \varphi_{n,2} & \varphi_{n,1} \end{bmatrix}, \quad \mathbf{x}_\varphi = (1.526, 0, 0). \tag{58}$$

Thus, mapping from frames to atomic coordinates can be expressed as a single transformation as,

$$[\mathrm{N}_n, (\mathrm{C}_\alpha)_n, \mathrm{C}_n, \mathrm{O}_n] = \zeta_{\text{f2a}}(T_n, \varphi_n), \tag{59}$$

where ζ_{f2a} represents the full frame-to-atom transformation.

Now, we describe how to construct frames from coordinates by utilizing rigidFrom3Point algorithm in AF2 as,

$$\begin{aligned} v_1 &= \mathrm{C}_n - (\mathrm{C}_\alpha)_n, \quad v_2 = \mathrm{N}_n - (\mathrm{C}_\alpha)_n \\ e_1 &= v_1 / \|v_2\|, \quad u_2 = v_2 - e_1(e_1^\top v_2) \\ e_2 &= u_2 / \|u_2\|, \quad e_3 = e_1 \times e_2 \\ \mathbf{r}_n &= \mathrm{concat}(e_1, e_2, e_3) \\ \mathbf{x}_n &= (\mathrm{C}_\alpha)_n \\ T_n &= (\mathbf{r}_n, \mathbf{x}_n) \end{aligned}$$

where the initial computations are based on Gram-Schmidt. This whole procedure can be expressed as a single transformation as,

$$T_n = \zeta_{\text{atom-to-frame}}(\mathrm{N}_n, (\mathrm{C}_\alpha)_n, \mathrm{C}_n). \tag{60}$$