NOTES ON :
SE(3) Diffusion Model with Application to Protein Backbone Generation

by Yim, Trippe, De Bortoli, Mathieu, et. al

Romain Hû, Thomas Winninger
M2 MVA
Algorithms for Protein Science Course

1 Generating protein backbones

The goal of the paper is to develop a sampling method for generating protein backbones from a diffusion process. In these notes, we recap the framework, methods and results obtained by the authors [1]. We especially focus on the design of a diffusion process on $\mathrm{SE}(3)^N$.

1.1 Representation of amino acids

Every amino acid in a protein contains a backbone structure of the form $N - C_\alpha - C - O$. This structure can be easily represented as a 3D frame by positioning C_α on 0 and treating the atoms C and N as vectors v_1, v_2 whose lengths and angle are determined from available data. Additionally, an angle ψ determines the oritentation of the oxygen atom O around the axis of v_1 as illustrated.

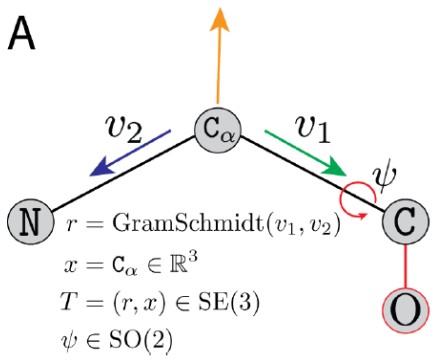

Figure 1: Illustration of an amino acid backbone as a 3D frame.

To position a frame X in $\mathbb{R}^3$, it is enough to apply a rigid transformation $T \cdot X = RX + x$ to it where:

$$\begin{cases} T = (R, x) \in \mathrm{SE}(3) \\ R \in \mathrm{SO}(3) \\ x \in \mathbb{R}^3 \end{cases}$$

The n^{th} residue can thus be represented by an element $T^n \in \mathrm{SE}(3)$. Generating a backbone structure with this representation therefore amounts to sample a sequence of elements $T^1, ...T^n$ from a probability distribution defined on $\mathrm{SE}(3)$. This distribution will precisely be obtained as the output of a diffusion process.

1.2 Diffusion processes

A diffusion process consists to transform an initial distribution $p_I \in \Delta(X)$ into a target distribution $p_T \in \Delta(X)$. Usually, the initial distribution p_I is chosen to be sampled easily (e.g. Gaussian or uniform). In machine learning, we are particularly interested in transforming p_I into a distribution similar to that of available data. This allows to generate new samples that remain close to real data with a possibility for variations.

A well-known method to achieve this is to use time reversal. We start with a variable X_0 sampled from the data distribution p_T, and let it evolve according to an SDE with brownian noise until it converges to the initial distribution p_I:

$$\begin{cases} \dot{X}_t = -b_t(X_t) + dB_t \\ X_0 \sim p_T \end{cases}$$

In this case, the associated Fokker-Planck equation, which describes the evolution of the density over time, is given by:

$$p_t = \nabla^* b_t p_t + \frac{1}{2}\Delta p_t$$

This tells us that the density p_t follows the drift b_t while spreading progressively over time due to the Laplacian. We can then use the Stein score $\nabla_x \log p_t$ as the optimal control of the reverse process:

$$\begin{cases} \dot{X}_t = b_t(X_t) + \nabla_x \log p_t + dB_t \\ X_0 \sim p_I \end{cases}$$

In reality, the marginal score $\nabla_x \log p_t$ is unavailable. However, it is proven that $\mathbb{E}_{X_0 \mid X_t}[\nabla \log p(X_t \mid X_0)] = \nabla \log p(X_t)) = \nabla \log p_t$ which motivates the use of the conditional score $\nabla_{X_t} \log p(X_t \mid X_0)$ as a proxy for approximating the true score. In practice, this is done by optimizing a neural network to learn the conditional score.

This exact method can be used for generating protein backbones. However, as residues are represented by elements of $\mathrm{SE}(3)$, it is necessary to define the diffusion process on $\mathrm{SE}(3)^N$.

2 Diffusion process on $\mathrm{SE}(3)$

In this section, we review the method used by the authors to design a diffusion process on $\mathrm{SE}(3)^N$. In particular, we will see how to build a brownian motion on $\mathrm{SE}(3)$ and the associated Fokker-Planck equation using the Laplace Beltrami operator.

2.1 Choice of metric

As a group, $\mathrm{SE}(3)$ can be written as $\mathrm{SO}(3) \ltimes \mathbb{R}^3$ as the composition of two rotations and two translations is given by:

$$(R, x) \cdot (R', x') = (RR', x + Rx')$$

However, when moving across $\mathrm{SE}(3)$, e.g. via brownian motion, one is not interested in the group structure. If one forgets that structure, $\mathrm{SE}(3)$ is simply a manifold and can be written as $\mathrm{SO}(3) \times \mathbb{R}^3$. This representation has the advantage of splitting the metric. $\forall (R, x), (R', x') \in \mathrm{SE}(3)$:

$$\langle (R,x) \mid (R',x') \rangle_{\mathrm{SE}(3)} = \langle R \mid R' \rangle_{\mathrm{SO}(3)} + \langle x \mid x' \rangle_{\mathbb{R}^3}$$

where the metric on $\mathrm{SO}(3)$ is given by the Killing form (which bi-invariant by the group action). This separation of metrics will greatly simplify all the objects introduced thereafter, beginning with brownian motion.

2.2 Brownian motion on $\mathrm{SO}(3)$

As we decompose $\mathrm{SE}(3)$ into the manifold $\mathrm{SO}(3) \times \mathbb{R}^3$ a brownian motion on $\mathrm{SE}(3)$ splits as:

$$B^{\mathrm{SE}(3)} = \left(B^{\mathrm{SO}(3)}, B^{\mathbb{R}^3}\right)$$

Brownian motion on $\mathbb{R}^3$ is already well-defined, so all that remains is to define a brownian motion on $\mathrm{SO}(3)$. To achieve this, we recall that as a Lie group $\mathrm{SO}(3)$ has a corresponding Lie algebra $\mathfrak{so}(3)$ which is a vector space of matrices with a well known basis:

$$Y_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad Y_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad Y_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Moreover, the Lie algebra is related to the group by the exponential map $\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$, which, in the case of compact Lie groups is surjective. In $\mathrm{SO}(3)$, the exponential map is tractable and given by the matrix exponential:

$$\exp(R) = \sum_{k \geq 0} \frac{R^k}{k!}$$

Therefore, to create a brownian motion increment on $\mathrm{SO}(3)$ starting at element R_t, one simply has to move in a random direction in $\mathfrak{so}(3)$, apply the exponential map and compose with R_t:

$$\begin{cases} R_{t+\Delta t} = R_t \exp\left(\sum_i z_i Y_i\right) \\ z_i \sim \mathcal{N}(0, \Delta t) \end{cases}$$

We now turn to the definition of the Fokker-Planck equation induced by this brownian motion. First, we must make a short detour through parametrizations and representations of $\mathrm{SO}(3)$.

2.3 Parametrization of $\mathrm{SO}(3)$

Elements of $\mathrm{SO}(3)$ can be represented through different parametrizations, such as Euler angles or quaternions. However, in the following, we will use the axis-angle parametrization: any element $R \in \mathrm{SO}(3)$ can be written as:

$$R = \exp(\theta K)$$

with angle $\theta \in [0, 2\pi]$ and axis $w = (w_1, w_2, w_3) \in \mathbb{S}^2$ such that $K = \sum_i w_i Y_i$. In this form R represents the rotation of angle θ around the axis w. Under this parametrization, the exponential map can be easily computed using the Rodrigues formula:

$$\exp(\theta K) = \mathrm{Id} + \sin(\theta) K + (1 - \cos(\theta)) K^2$$

Moreover, this map has an inverse that allows to recover the axis-angle parametrization of an element R:

$$\begin{cases} \theta(R) = \cos^{-1}\left(\frac{\mathrm{Tr}(R)-1}{2}\right) \\ K(R) = \frac{(R_{32}-R_{23})Y_1+(R_{13}-R_{31})Y_2+(R_{21}-R_{12})Y_2}{2\sin(\theta)} \end{cases}$$

2.4 Representations and characters of Lie groups

As a Lie group, $\mathrm{SE}(3)$ admits several representations. A representation is given by a homomorphism $\rho : \mathrm{SE}(3) \to \mathrm{GL}(V)$ where V is a vector space, that is, a map that satisfies:

$$\begin{cases} \rho(gh) = \rho(g)\rho(h) \\ \rho(e) = 0 \end{cases}$$

If V has dimension l, ρ is called an l dimensional representation. Certain representations are called irreducible, which means that no subspace of V is invariant by the action ρ, except V itself or 0.

A character is a 1-dimensional representation $\chi_l : \mathrm{SE}(3) \to \mathbb{R}$ that is usually given by applying the trace operator after the representation:

$$\chi(g) = \mathrm{Tr}(\rho(g))$$

Again, a character has dimension l if ρ has dimension l. The l-dimensional characters of $\mathrm{SO}(3)$ are known and are given, in axis-angle parametrization, by:

$$\chi_l(R) = \frac{\sin\left(\left(l+\frac{1}{2}\right)\theta(R)\right)}{\sin\left(\frac{1}{2}\theta(R)\right)}$$

2.5 The Laplace-Beltrami operator

For a C^∞ function $f : \mathrm{SO}(3) \to \mathbb{R}$, the Laplace-Beltrami operator is defined as:

$$\Delta_{\mathrm{SO}(3)} f(R) = \sum_i \partial_t^2 f(R\exp(tY_i))$$

This definition is very similar to the standard Laplace operator in Euclidean space except that infinitesimal displacements are realized through the exponential map.

It is a result of the Peter-Weyl theorem that the characters χ_l form an orthonormal basis in the space of square integrable functions $L^2(\mathrm{SO}(3))$. Together with Schur's lemma, it guarantees that the Laplace-Beltrami operator can be diagonalized and its eigenfunctions are precisely the characters of $\mathrm{SO}(3)$ with eigenvalues $\lambda_l = -l(l+1)$. We therefore have, $\forall f \in L^2(\mathrm{SO}(3))$:

$$\begin{aligned} \Delta_{\mathrm{SO}(3)} f &= \Delta_{\mathrm{SO}(3)} \sum_l a_l \chi_l \text{ for some } a_l \in \mathbb{R} \\ &= \sum_l a_l \Delta_{\mathrm{SO}(3)} \chi_l \\ &= \sum_l -a_l l(l+1)\chi_l \end{aligned}$$

This can be used to solve the heat equation in $\mathrm{SO}(3)$:

$$\begin{aligned} \dot{f}_t = \Delta_{\mathrm{SO}(3)} f_t &\Leftrightarrow \sum_l a_l'(t)\chi_l = \sum_l -a_l(t) l(l+1)\chi_l \\ &\Leftrightarrow a_l(t) = a_l(0)e^{-l(l+1)t} \end{aligned}$$

In the context of diffusion, the heat equation in $\mathrm{SO}(3)$ is given by $\dot{p}_t = \frac{1}{2}\Delta_{\mathrm{SO}(3)} p_t$ with $p_t = p(R_t \mid R_0)$. The solution of this equation is given by:

$$\begin{aligned} p(R_t \mid R_0) &= \sum_l d_l e^{-\frac{1}{2}\lambda_l t} \chi_l(R_0^{-1} R_t) \\ &= \sum_l (2l+1) e^{-\frac{1}{2} l(l+1) t} \frac{\sin\left(\left(l + \frac{1}{2}\right)\theta(R_0^T R_t)\right)}{\sin\left(\frac{1}{2}\theta(R_0^T R_t)\right)} \end{aligned}$$

Moreover, because of the separation of metrics, the Laplace-Beltrami operator on $\mathrm{SE}(3)$ splits:

$$\Delta_{\mathrm{SE}(3)} f = \Delta_{\mathrm{SO}(3)} f + \Delta_{\mathbb{R}^3} f$$

2.6 Score on $\mathrm{SE}(3)$

Another consequence of the separation of metrics is that the gradient of a function $f : \mathrm{SE}(3) \to \mathbb{R}$ also splits:

$$\nabla_{R,x} f(R, x) = (\nabla_x f(R, x), \nabla_R f(R, x))$$

In our case, computing $\nabla_{x_t} \log p(x_t \mid x_0)$ is easy and is given by:

$$\nabla_{x_t} \log p(x_t \mid x_0) = \frac{e^{-\frac{1}{2}t} x_0 - x_t}{1 - e^{-t}}$$

All that remains is to compute $\nabla_{R_t} \log p(R_t \mid R_0)$. Recall the inverse of the exponential map for the axis-angle parametrization, giving the angle θ of the parametrization:

$$\theta(R) = \cos^{-1}\left(\frac{\mathrm{Tr}(R) - 1}{2}\right)$$

It's gradient is given by:

$$\nabla_R \theta(R) = \frac{R \log(R)}{\theta(R)}$$

with $\log(R)$ the matrix logarithm. This allows to obtain the gradient of the score function for the diffusion process in $\mathrm{SO}(3)$:

$$\begin{aligned} \nabla_{R_t} \log p(R_t \mid R_0) &= \frac{\nabla_{\theta(R_0^T R_t)} p(R_t \mid R_0)}{p(R_t \mid R_0)} \nabla_{R_t} \theta(R_0^T R_t) \\ &= \frac{\nabla_{\theta(R_0^T R_t)} p(R_t \mid R_0)}{p(R_t \mid R_0)} \frac{R_0^T R_t \log(R_0^T R_t)}{\theta(R_0^T R_t)} \end{aligned}$$

In practice, this formula and the formula for $p(R_t \mid R_0)$ are truncated at a sufficiently large number of terms in the sum. All the tools are now set to define the diffusion process on $\mathrm{SE}(3)^N$:

$$\text{Forward diffusion: } \begin{cases} \dot{x}_t = -\frac{1}{2}x_t + dB_t^{\mathbb{R}^3} \\ \dot{R}_t = dB_t^{\mathrm{SO}(3)} \\ (R_0, x_0) \sim p_T \end{cases}$$

$$\text{Backward diffusion: } \begin{cases} \dot{x}_t = \frac{1}{2}x_t + \nabla_{x_t} \log p(x_t \mid x_0) + dB_t^{\mathbb{R}^3} \\ \dot{R}_t = \nabla_{R_t} \log p(R_t \mid x_0) + dB_t^{\mathrm{SO}(3)} \\ (R_0, x_0) \sim p_I \end{cases}$$

3 $\mathrm{SE}(3)$ invariance of the diffusion process

A desirable property for the diffusion model on $\mathrm{SE}(3)^N$ is to have $\mathrm{SE}(3)$-invariance. However, as $\mathrm{SE}(3)$ is not compact due to the translations living in $\mathbb{R}^3$, it is not possible to build a probability distribution that is $\mathrm{SE}(3)$-invariant on $\mathrm{SE}(3)^N$ (althrough it is still possible to build a general measure that is).

However, the authors show that any $\mathrm{SE}(3)$-invariant general measure μ on $\mathrm{SE}(3)^N$ can be disintegrated as:

$$\mu = \eta \otimes \overline{\mu}$$

with $\overline{\mu}$ proportional to the Lebesgue measure on $\mathbb{R}^3$ and η an $\mathrm{SO}(3)$-invariant probability distribution on the subgroup of $\mathrm{SE}(3)^N$ which has its center of mass fixed to 0 i.e. $\frac{1}{N}\sum_i x_i = 0$. This subgroup is noted $\mathrm{SE}(3)_0^N$.

Keeping the diffusion process on $\mathrm{SE}(3)_0^N$ using the previous diffusion equations is easy, since it suffices to add a matrix operator P that removes the center of mass to the equations of the translation components, i.e. $P(x) = x - \frac{1}{N}\sum_i x_i$. The updated equations for translations look like:

$$\text{Forward centered diffusion: } \dot{x}_t = -\frac{1}{2}Px_t + PdB_t^{\mathbb{R}^3}$$

$$\text{Backward centered diffusion: } \dot{x}_t = \frac{1}{2}Px_t + P\nabla_{x_t} \log p(x_t \mid x_0) + PdB_t^{\mathbb{R}^3}$$

Under these equations, the forward process converges to the $\mathrm{SO}(3)$-invariant measure on $\mathrm{SE}(3)_0^N$,

$$p_I = P_{\#}(\mathcal{N}(0,1)^{\otimes N} \otimes \mathcal{U}(\mathrm{SO}(3))^{\otimes N})$$

with $\mathcal{U}(\mathrm{SO}(3))$ the uniform Haar measure on $\mathrm{SO}(3)$.

The last guarantee is given by Proposition 3.6 from the authors. It states that if a diffusion process on a Lie group G starts from an H-invariant distribution, where H is a subgroup of G, with H-equivariant drift and diffusion coefficients, then the distribution remains H-invariant throughout the backward evolution and its score remains H-equivariant.

Given that $\mathrm{SO}(3)$ is a subgroup of $\mathrm{SE}(3)_0^N$, the backward process initialized at $(R_0, x_0) \sim p_I$ remains $\mathrm{SO}(3)$-invariant and the score remains $\mathrm{SO}(3)$-equivariant. This observation incites to incorporate $\mathrm{SO}(3)$-invariance into the score network.

4 Practice

In practice, the scores $\nabla_{x_t} \log p(x_t \mid x_0)$ and $\nabla_{R_t} \log p(R_t \mid R_0)$ are unavailable. They are usually approximated via a neural network s_t^θ trained to minimize the denoising score matching loss (DSM):

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}\left[\lambda_t \left\| s_t^\theta(T_t) - \nabla_{T_t} \log p(T_t \mid T_0) \right\|^2\right]$$

with λ_t a weighting schedule and $T_t = (R_t, x_t)$. The backward equations then read,

$$\begin{cases} \dot{x}_t = \frac{1}{2} P x_t + s_t^\theta(x_t) + P dB_t^{\mathbb{R}^3} \\ \dot{R}_t = s_t^\theta(R_t) + dB_t^{\text{SO}(3)} \\ (R_0, x_0) \sim p_I \end{cases}$$

However, because an explicit formula for the score $\nabla_{T_t} \log p(T_t \mid T_0)$, depending on T_0, is available in our case, the network only needs to estimate $\hat{T}_0^t = f_t^\theta(T_t)$ before plugging it in the score formula. Optimizing the DSM loss ensures that:

$$\nabla_{T_t} \log p\left(T_t \mid \hat{T}_0^t\right) = \nabla_{T_t} \log p(T_t \mid f_t^\theta(T_t) = s_t^\theta(T_t) \approx \nabla_T \log p(T_t)$$

The network f^θ corresponds to the FramePred algorithm, whose architecture is based on ALphaFold2. We note SE(3)-invariance of Invariant Point Attention (IPA) allows to achieve SE(3)-invariance of the output. The complete algorithm including FramePred and SE(3) diffusion is called FrameDiff and its final output corresponds to a prediction $f_t^\theta(T_t)$ after enough iterations t of the backward equation.

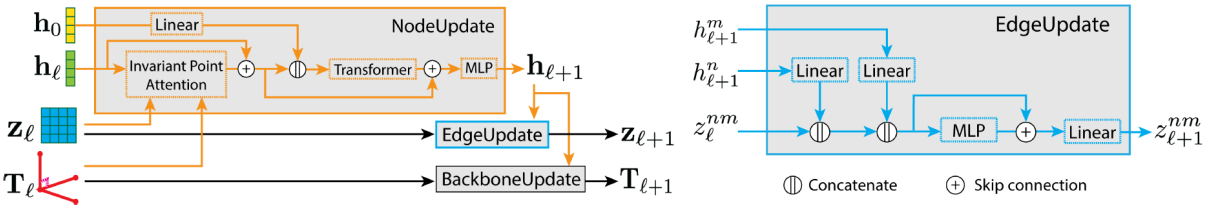

Figure 2: one layer of the FramePred network with h_l corresponding to node embeddings, z_l to edge embeddings and T_l the predicted frame.

4.1 Auxiliary losses

In practice, the authors include several auxiliary losses in the FrameDiff model to improve its performance. In order to avoid steric clashes or chain breaks at the end of the diffusion process, where the topology begins to be fine grained, structural losses can be added to the predicted positions of atoms:

$$\mathcal{L}_{\text{bb}} = \frac{1}{4N} \sum_{a,n} \|a_n - \hat{a}_n\|^2$$

$$\mathcal{L}_{\text{2D}} = \frac{1}{Z} \sum_{\substack{n,m \\ a,b,d_{\text{ab}}^{\text{nm}} < 0.6}} \left\| d_{\text{ab}}^{\text{nm}} - \hat{d}_{\text{ab}}^{\text{nm}} \right\|^2$$

where $d_{\text{ab}}^{\text{nm}}$ represents the true distance between atoms a, b in residues n, m and Z a normalization. The $\mathcal{L}_{\text{bb}}$ loss is a direct MSE on atomic positions while $\mathcal{L}_{\text{2D}}$ is a direct supervision on the distance between atoms that are very close to each other. These structural losses are particularly important in order to generate the angles ψ determining the position of the oxygen atoms.

Multiple parameters can also be adjusted such as the weighting schedule λ_t, diffusion coefficients for R_t, x_t in the backward equations or the weight of auxiliary losses.

4.2 Results

The authors measure the performance of FrameDiff on three axis:

- Designability: is the generated backbone realistic ? To achieve this, the authors use proteinMPNN to generate a sequence of amino acids from a backbone. Then, they inject this sequence into the ESMFold algorithm to predict a plausible fold for this sequence. The outputs of FrameDiff and ESMFold are then compared using scRMSD and scTM. This process is illustrated in Figure 3.
- Diversity: is measured using the MaxCluster algorithm with a set TM threshold to cluster the outputs of FrameDiff.
- Novelty: outputs of FrameDiff are compared to available structured in the PDB using FoldSeek and the scTM metric.

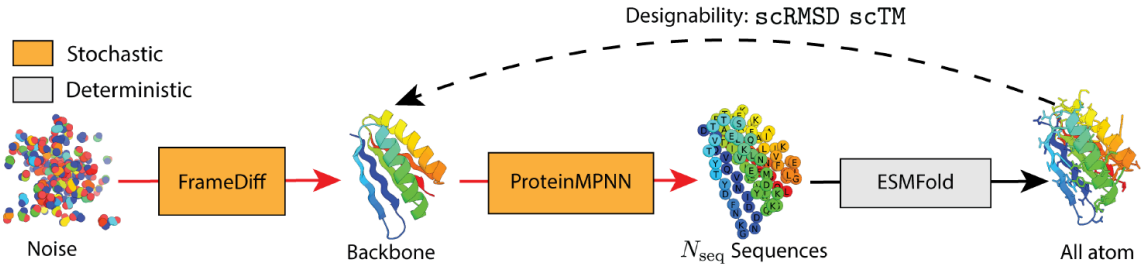

Figure 3: Protocol for computing designability of a generated backbone.

Figure 4 showcases some performances of FrameDiff. Designability is high for backbone of length 100 or less and quickly explodes for longer samples. Figure 4A also showcases a comparison with the RFDiffusion model who remains more performant than FrameDiff on designabilty, at the cost of having 4 fold more parameters. For designable outputs of FrameDiff, most of them remain close to known structures in the PDB (upper left quadrant in Figure 4B). However, the model is able to produce a few samples that are both designable and novel (lower left quadrant in Figure 4B).

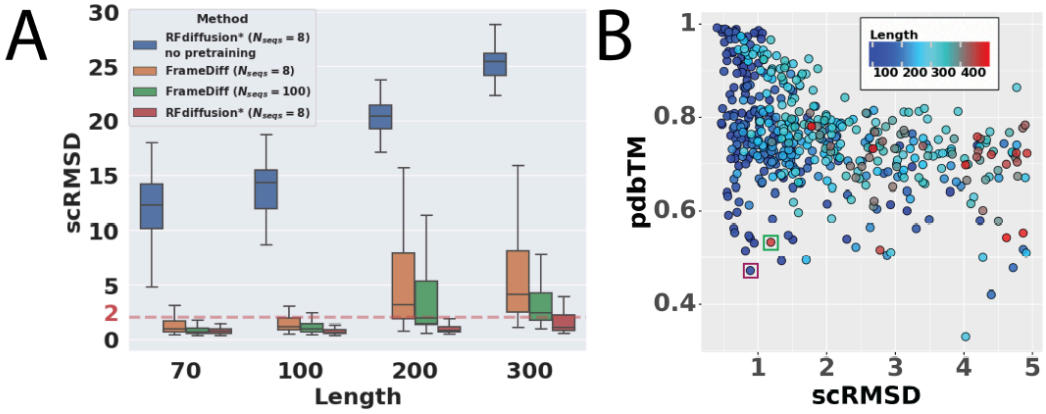

Figure 4: (A) Designability (scRMSD) against length of the generated backbone for different models. (B) Novelty (pdbTM) against designability (scRMSD) for FrameDiff.

4.3 Experiments

One important aspect of the geometry of $\mathrm{SO}(3)$, is that it makes it possible to conserve the angles and bound lengths of the residues during the noising process. Thus, making the rigid-body assumption

possible, whereas an Euclidean noise, applying per-atom independent Gaussian noise breaks bounds and mess angles, as shown in Figure 5.

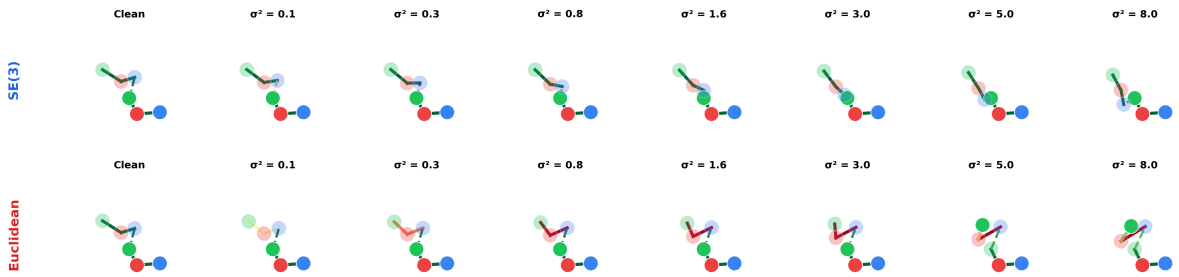

Figure 5: Noise propagation in both $SO(3)$ and $\mathbb{R}^3$. Blue: N, red: C_α, green: C, red bond: broken bond. The transparent residue is moving while the solid residue is fixed.

Using $\mathrm{SO}(3)$ makes it also possible to perform correct interpolations, between two positions or two conformations.

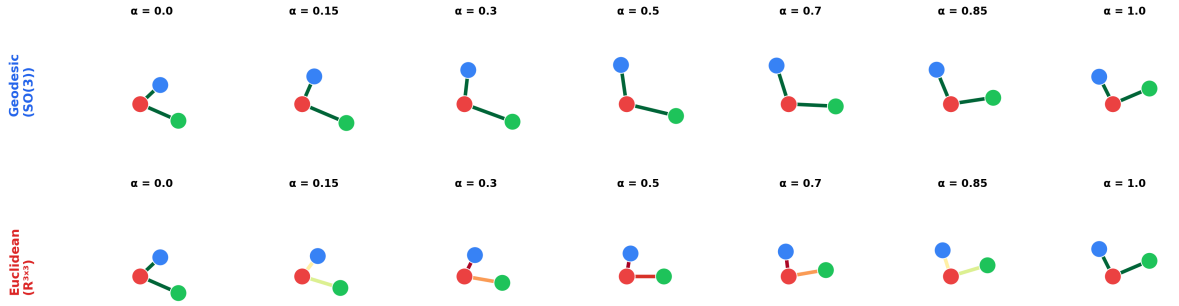

Figure 6: Interpolation in $SO(3)$ and $\mathbb{R}^3$. Blue: N, red: C_α, green: C, peptide bond: - -, red bond: broken bond.

In Figure 6, we see an interpolation between two rotations applied to a residue frame. The geodesic (SO(3), top): uses exp/log maps, stays on the manifold at every step. The frame stays rigid, the bond lengths are preserved. On the contrary, the Euclidean (bottom): linearly interpolates matrix entries $R(\alpha) = (1-\alpha)R_0 + \alpha R_1$. It leaves $\mathrm{SO}(3)$ mid-interpolation, the frame shears, and bonds distort.

5 Conclusion & future work

This paper laid a rigorous framework to apply transformation and diffusion on protein residues. It also makes it possible to generate new samples that generalize beyond the PDB. On the diffusion side, research has evolved toward flow matching methods, which was adopted for this task with FrameFlow [2]. More recently, methods like ReQFlow [3] propose a new structure: quaternions in $\mathbb{S}^3$, which produces more straight geodesics, which is a important factor for flow matching and optimal transport. Both methods make sampling much faster than diffusion, and produce higher quality samples, with higher designability.

Bibliography

[1] J. Yim *et al.*, "SE(3) diffusion model with application to protein backbone generation," *International Conference of Machine Learning (ICML) 2023*, 2023, [Online]. Available: http://arxiv.org/abs/2302.02277v3

[2] J. Yim *et al.*, "Fast protein backbone generation with SE(3) flow matching." [Online]. Available: https://arxiv.org/abs/2310.05297

[3] A. Yue, Z. Wang, and H. Xu, "ReQFlow: Rectified Quaternion Flow for Efficient and High-Quality Protein Backbone Generation." [Online]. Available: https://arxiv.org/abs/2502.14637