# EQUIJUMP: PROTEIN DYNAMICS SIMULATION VIA SO(3)-EQUIVARIANT STOCHASTIC INTERPOLANTS

**Allan dos Santos Costa**[*†]
MIT Center for Bits and Atoms
NVIDIA
allanc@mit.edu

**Ilan Mitnikov**[*]
MIT Center for Bits and Atoms
ilanm@mit.edu

**Franco Pellegrini**[*]
SISSA

**Ameya Daigavane**
MIT Atomic Architects

**Mario Geiger**
NVIDIA

**Zhonglin Cao**
NVIDIA

**Karsten Kreis**
NVIDIA

**Tess Smidt**
MIT Atomic Architects

**Emine Kucukbenli**
NVIDIA

**Joseph Jacobson**
MIT Center for Bits and Atoms

## ABSTRACT

Mapping the conformational dynamics of proteins is crucial for elucidating their functional mechanisms. While Molecular Dynamics (MD) simulation enables detailed time evolution of protein motion, its computational toll hinders its use in practice. To address this challenge, multiple deep learning models for reproducing and accelerating MD have been proposed drawing on transport-based generative methods. However, existing work focuses on generation through transport of samples from prior distributions, that can often be distant from the data manifold. The recently proposed framework of stochastic interpolants, instead, enables transport between arbitrary distribution endpoints. Building upon this work, we introduce EquiJump, a transferable SO(3)-equivariant model that bridges all-atom protein dynamics simulation time steps directly. Our approach unifies diverse sampling methods and is benchmarked against existing models on trajectory data of fast-folding proteins. EquiJump achieves state-of-the-art results in dynamics simulation with a transferable model on all of the fast-folding proteins.

## 1 INTRODUCTION

Proteins are the workhorses of the cell, and simulating their dynamics is critical to biological discovery and drug design (Karplus and Kuriyan, 2005). Molecular Dynamics (MD) simulation is an important tool that leverages physics for time evolution, enabling precise exploration of the conformational space of proteins (Hollingsworth and Dror, 2018). However, sampling with physically-accurate molecular potentials requires small integration time steps, often making the simulation of phenomena at relevant biological timescales prohibitive (Lane et al., 2013). To tackle this challenge, several studies have adopted deep learning models to capture surrogates of MD potentials and dynamics (Noé et al., 2020; Durumeric et al., 2023; Arts et al., 2023). More recent works (Schreiner et al., 2023; Li et al., 2024; Jing et al., 2024b) have proposed to use deep learning-based simulators trained on long-interval snapshots of MD trajectories to predict future states given some starting configuration. These models draw from neural transport models (Ho et al., 2020; Lipman et al., 2022), learning a conditional or guided bridge between a prior distribution ($\rho_0 = \mathcal{N}$) and the target data manifold of simulation steps ($\rho_1 = \rho_{\text{data}}$). In contrast, the recent paradigm of Stochastic Interpolants (Albergo et al., 2023a; Albergo and Vanden-Eijnden, 2023) provides a method for directly bridging distinct arbitrary distributions. In this work, we build upon this framework and introduce **EquiJump**, a Two-Sided Stochastic Interpolant model which bridges between long-interval timesteps of protein simulation directly (Figure 1). EquiJump is SO(3)-equivariant and simulates all heavy atoms directly

[*] Equal Contribution; [†] Work performed during internship at NVIDIA

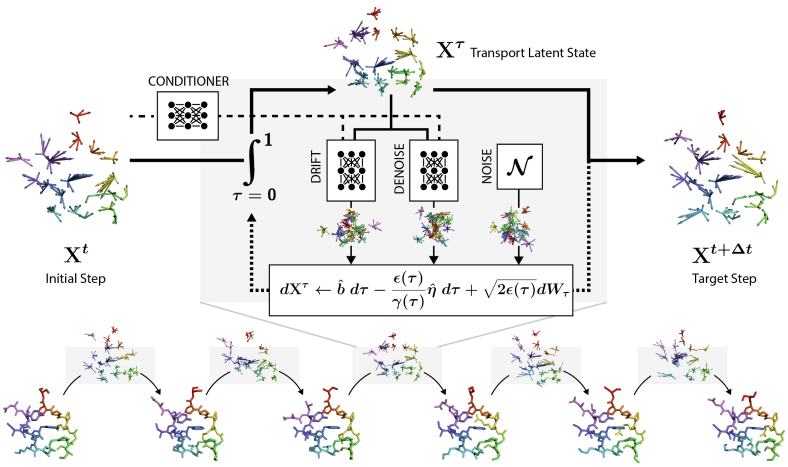


Figure 1: **Direct bridging of 3D Protein Simulation**: EquiJump runs an stochastic interpolants-based transport on 3D coordinates and geometric features to generate future time frames from an initial state. Gray boxes depict transport across the latent space, which takes Gaussian perturbations and uses learned noise and drift to transform all-atom proteins across time and 3D space.

in 3D. We train a transferable model on 12 fast-folding proteins (Majewski et al., 2023; Lindorff-Larsen et al., 2011) and successfully recover their dynamics.

## 2 Related Work

Advances in protein modeling through deep learning have led to the development of several models capable of replicating MD trajectories. Most common MD learning approaches are trained on forces (Wang et al., 2019; Husic et al., 2020; Satorras et al., 2021; Batatia et al., 2022). CG-MLFF Majewski et al. (2023) implements a unified transferable model for multiple proteins with a coarse-grained model. (Fu et al., 2023) learns to predict accelerated, coarse-grained dynamics of polymers with GNNs. More recent approaches use a generative backbone to proxy dynamics potentials. Köhler et al. (2023) utilizes normalizing flows (Gabrié et al., 2022) for coarse grained dynamics, while Arts et al. (2023) builds upon Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) for handling both ensemble and dynamics generation. Generative models that are capable of sampling ensembles were also employed to reproducing MD distributions. AlphaFlow (Jing et al., 2024a) uses Flow Matching (Lipman et al., 2022) to sample equilibrium states, while BioEmu (Lewis et al., 2024) leverages DDPM for generating MD conformers. Closer to our approach, (Daigavane et al., 2024) utilizes Walk-Jump Sampling (Saremi and Hyvärinen, 2019) to sample from the Boltzmann distribution of small-peptide conformations. Alternatively, recent generative models generate next-step predictions by transforming samples from a prior distribution, while conditioning on a source configuration. Timewarp (Klein et al., 2023) enhances MCMC sampling with conditional normalizing flows, while ITO (Schreiner et al., 2023) learns a conditional diffusion model for next-step prediction. Similarly, F$^3$low (Li et al., 2024) employs Optimal Transport Guided Flow Matching (Zheng et al., 2023). MDGen (Jing et al., 2024b) applies One-Sided Stochastic Interpolants (Ma et al., 2024) to generate time frames. Finally, concurrent work (Han et al., 2024; Luo et al., 2024) are similar to EquiJump, and focus on generating next-step 3D configurations with transport from a source step.

## 3 Methods

### 3.1 Two-Sided Stochastic Interpolants for Dynamics Simulation

Neural transport methods have demonstrated outstanding performance in generative tasks (Ma et al., 2024; Liu et al., 2023; Lipman et al., 2022; Ho et al., 2020). Stochastic Interpolants (Albergo et al., 2023a; Albergo and Vanden-Eijnden, 2023) are a recently proposed class of generative models that have reached state-of-the-art results in image generation (Ma et al., 2024; Albergo et al., 2023b).

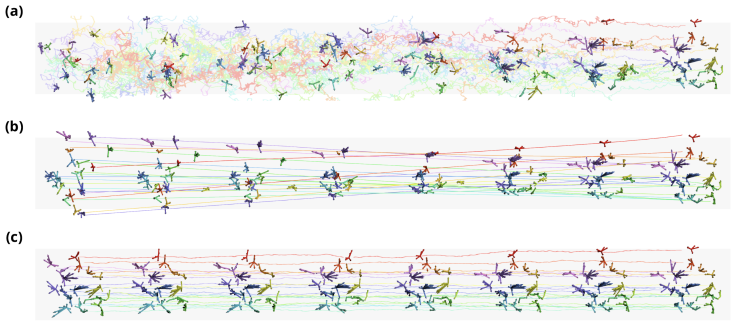


Figure 2: **Neural Transport of Tensor Clouds**. **(a) DDPM** defines an SDE for denoising samples from a Gaussian prior, while standard **(b) Flow Matching** traces a velocity field-based ODE for moving the Gaussian samples. **(c) Two-Sided Stochastic Interpolants** instead enable transporting through a local, normally-perturbed latent space that remains close to the manifold of the data.

One-sided stochastic interpolants, which generalize flow matching and denoising diffusion models, transport samples from a prior distribution $\mathbf{X}_0 \sim \mathcal{N}$ to a target data distribution $\mathbf{X}_1 \sim \rho_1$ by utilizing latent variables $\mathbf{Z} \sim \mathcal{N}$ through the stochastic process $\{\mathbf{X}_\tau\}$:

$$\mathbf{X}_\tau = J(\tau, \mathbf{X}_1) + \alpha(\tau)\mathbf{Z} \tag{1}$$

where $\tau \in [0, 1]$ is the time parameterization. The interpolant function $J$ satisfies boundary conditions $J(0, \mathbf{X}_1) = 0$ and $J(1, \mathbf{X}_1) = \mathbf{X}_1$, and the noise schedule $\alpha$ satisfies $\alpha(0) = 1$ and $\alpha(1) = 0$. In contrast, two-sided stochastic interpolants enable learning the transport from $\mathbf{X}_0 \sim \rho_0$ to $\mathbf{X}_1 \sim \rho_1$ when $\rho_0$ and $\rho_1$ are arbitrary probability distributions (Figure 2). Two-sided interpolants are described by the stochastic process $\{\mathbf{X}_\tau\}$:

$$\mathbf{X}_\tau = I(\tau, \mathbf{X}_0, \mathbf{X}_1) + \gamma(\tau)\mathbf{Z} \tag{2}$$

where $\tau \in [0, 1]$ and to ensure boundary conditions, the interpolant $I$ and noise schedule $\gamma$ must satisfy the following: $I(0, \mathbf{X}_0, \mathbf{X}_1) = \mathbf{X}_0$ and $I(1, \mathbf{X}_0, \mathbf{X}_1) = \mathbf{X}_1$, and $\gamma(0) = \gamma(1) = 0$. The probability $p(\tau, \mathbf{X})$ of a stochastic interpolant satisfies the transport equation:

$$\partial_\tau p(\tau, \mathbf{X}) + \nabla \cdot (b(\tau, \mathbf{X})p(\tau, \mathbf{X})) = 0 \tag{3}$$

and the boundary conditions $p(0, \mathbf{X}) = p_0$ and $p(1, \mathbf{X}) = p_1$. Here, $b(\tau, \mathbf{X})$ is the expected velocity:

$$b(\tau, \mathbf{X}) = \mathbb{E}\left[\partial_\tau \mathbf{X}_\tau \mid \mathbf{X}_\tau = \mathbf{X}\right] = \mathbb{E}\left[\partial_\tau I(\tau, \mathbf{X}_0, \mathbf{X}_1) + \partial_\tau \gamma(\tau)\mathbf{Z} \mid \mathbf{X}_\tau = \mathbf{X}\right] \tag{4}$$

With stochastic sampling, we can similarly define the noise term $\eta(\tau, \mathbf{X})$ as:

$$\eta(\tau, \mathbf{X}) = \mathbb{E}[\mathbf{Z} \mid \mathbf{X}_\tau = \mathbf{X}] \tag{5}$$

In practice, the exact forms of $b$ and $\eta$ are not known for arbitrary distributions $p_0$, $p_1$, and are thus parameterized by neural networks. (Albergo et al., 2023a) shows that we can learn the functions $\hat{b} \approx b$ and $\hat{\eta} \approx \eta$ by optimizing:

$$\min_{\hat{b}} \int_0^1 \mathbb{E}\Big[\frac{1}{2}\|\hat{b}(\tau, \mathbf{X}_\tau)\|^2 - (\partial_\tau I(\tau, \mathbf{X}_0, \mathbf{X}_1) + \partial_\tau \gamma(\tau)\mathbf{Z}) \cdot \hat{b}(\tau, \mathbf{X}_\tau)\Big] d\tau \tag{6}$$

$$\min_{\hat{\eta}} \int_0^1 \mathbb{E}\Big[\frac{1}{2}\|\hat{\eta}(\tau, \mathbf{X}_\tau)\|^2 - \mathbf{Z} \cdot \hat{\eta}(\tau, \mathbf{X}_\tau)\Big] d\tau \tag{7}$$

We can then sample $\mathbf{X}_{\tau=1} \sim p(\tau = 1, \mathbf{X}_1)$ through the stochastic differential equation::

$$d\mathbf{X}_\tau = \left(\hat{b}(\tau, \mathbf{X}_\tau) - \frac{\epsilon(\tau)}{\gamma(\tau)}\hat{\eta}(\tau, \mathbf{X}_\tau)\right) d\tau + \sqrt{2\epsilon(\tau)} dW_\tau \tag{8}$$

where $W_\tau$ is the Weiner process. With the expected velocity $\hat{b}$ and noise $\hat{\eta}$, the above equations can be integrated numerically starting from $(\tau = 0, \mathbf{X}_0 \sim p_0)$ to $(\tau = 1, \mathbf{X}_1 \sim p_1)$. We note that following from eq. (8) the probability $p(\tau, \mathbf{X}_\tau)$ is SO(3)-equivariant when $\hat{b}$ and $\hat{\eta}$ are SO(3)-equivariant and $dW_\tau$ is isotropic. We extend the Two-Sided Stochastic Interpolant framework to learn a time evolution operator from trajectory data $[\mathbf{X}^t]_{t=1}^L$. Given a source time step $\mathbf{X}^t$ and its consecutive target step $\mathbf{X}^{t+1}$, we define the distribution boundaries of our interpolant as $\rho_0 = \rho(\mathbf{X}^t)$ and $\rho_1 = \rho(\mathbf{X}^{t+1} \mid \mathbf{X}^t)$ (Figure 1). We note that the conditional nature of the target distribution requires that our predictions for drift $\hat{b}$ and noise $\hat{\eta}$ are explicitly conditioned on the source step $\mathbf{X}^t$.

### 3.2 MULTIMODAL INTERPOLANTS FOR ALL-ATOM PROTEIN STRUCTURES

We treat data $\mathbf{X}$ represented as geometric features positioned in three-dimensional space, $\mathbf{X} = [(\mathbf{V}_i, \mathbf{P}_i)]_{i=1}^N$, which we refer to as the *Tensor Cloud* representation (Figure 2). In this formulation, each $\mathbf{V}_i$ is a tensor of irreducible representations (irreps) of O(3) or SO(3), associated with a 3D coordinate $\mathbf{P}_i \in \mathbb{R}^3$. The feature representations $\mathbf{V}$ are arrays of irreps up to order $l_{\max}$ where for each $l \in [0, l_{\max}]$, the tensor $\mathbf{V}^l$ represents geometric features with dimensions $\mathbf{V}^l \in \mathbb{R}^{H\times(2l+1)}$, where $H$ denotes the feature multiplicity. We extend interpolant eq. (8) to the multi-modal type $\mathbf{X}_i = (\mathbf{V}_i, \mathbf{P}_i)$ by learning independent geometric features and coordinate updates as $d\mathbf{X}_i^\tau = (d\mathbf{V}_i^\tau, d\mathbf{P}_i^\tau)$. For computing the losses eqs. (6) and (7), we compute the Tensor Cloud dot product by independently treating each component $\mathbf{X}_i \cdot \mathbf{X}_j = \mathbf{V}_i \cdot \mathbf{V}_j + \mathbf{P}_i \cdot \mathbf{P}_j$.

We represent a protein monomer $(\mathbf{R}, \mathbf{X})$ as a sequence $\mathbf{R}$ and a Tensor Cloud $\mathbf{X}$. Our model is designed to update $\mathbf{X}$ while being conditioned on $\mathbf{R}$. Each residue $i$ consists of three components: a residue label $\mathbf{R}_i \in \mathcal{R} = \{\text{ALA, GLY}, \dots\}$, the $C_\alpha$ 3D coordinate $\mathbf{P}_i^\alpha \in \mathbb{R}^3$, and a geometric feature of order $l = 1$ with multiplicity 13, $\mathbf{V}_i^A \in \mathbb{R}^{13\times3}$. This feature encodes the relative 3D vector from the $C_\alpha$ to all other heavy atoms in the residue, following a canonical ordering (King and Koes, 2020; Costa et al., 2024). For residues with fewer than 13 non-$C_\alpha$ heavy atoms, we pad corresponding vectors.

### 3.3 NEURAL NETWORK ARCHITECTURE AND TRAINING

To efficiently process 3D data, we utilize established modules of Euclidean-equivariant neural networks (Geiger and Smidt, 2022; Miller et al., 2020). We design a network with 4 headers to predict the drift $\hat{b} = (\hat{b}_\mathbf{V}, \hat{b}_\mathbf{P})$ and noise $\hat{\eta} = (\hat{\eta}_\mathbf{V}, \hat{\eta}_\mathbf{P})$ terms (Figure 5). Given a configuration $\mathbf{X}^t$, we first prepare a hidden representation $\tilde{\mathbf{X}}^t = \mathbf{f}_{\text{cond}}(\mathbf{R}, \mathbf{X}^t)$. The 4 headers take $(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)$ to produce predictions of each component of the drift $\hat{b}$ and the noise $\hat{\eta}$. For efficiency, the embedding $\tilde{\mathbf{X}}^t$ is made independent of $\tau$, and only the headers are used in the integration loop of the latent transport. Refer to Appendix A.2 for more details. We train and sample EquiJump following Algorithms 1 and 2:

**Algorithm 1** EquiJump Training

**Require:** Sequence $\mathbf{R}$
**Require:** Trajectory Data $[\mathbf{X}^t]_{t=1}^T$
**Require:** Interpolant Parameters $I_\tau, \gamma(\tau)$
**Require:** Networks $\hat{b}_\mathbf{V}$, $\hat{b}_\mathbf{P}$, $\hat{\eta}_\mathbf{V}$, $\hat{\eta}_\mathbf{P}$, $\mathbf{f}_{\text{cond}}$
$t \sim \mathcal{U}(1, T-1)$
$\tau \sim \mathcal{U}(0, 1)$
$\mathbf{Z}^\tau \sim \mathcal{N}(0, \mathbb{I})$
$\tilde{\mathbf{X}}^t \leftarrow \mathbf{f}_{\text{cond}}(\mathbf{R}, \mathbf{X}^t)$
$\mathbf{X}_\tau^t \leftarrow (1-\tau)\cdot\mathbf{X}^t + \tau\cdot\mathbf{X}^{t+1} + \gamma(\tau)\mathbf{Z}^\tau$
$\hat{\eta} \leftarrow \big(\hat{\eta}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{\eta}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
$\hat{b} \leftarrow \big(\hat{b}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{b}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
**Gradient Step**
$-\nabla\Big(\frac{1}{2}\|\hat{b}\| - \hat{b}\cdot\big(\partial_\tau I_\tau(\mathbf{X}^t, \mathbf{X}^{t+1}) + \dot{\gamma}(\tau)\cdot\mathbf{Z}^\tau\big) + \frac{1}{2}\|\hat{\eta}\| - \hat{\eta}\cdot\mathbf{Z}^\tau\Big)$

**Algorithm 2** EquiJump Sampling

**Require:** Sequence $\mathbf{R}$
**Require:** Start Step $\mathbf{X}^t$
**Require:** Interpolant Parameters $\epsilon(\tau), \gamma(\tau)$
**Require:** Networks $\hat{b}_\mathbf{V}$, $\hat{b}_\mathbf{P}$, $\hat{\eta}_\mathbf{V}$, $\hat{\eta}_\mathbf{P}$, $\mathbf{f}_{\text{cond}}$
**Require:** Integration Timestep $d\tau$
$\mathbf{X}_{\tau=0}^t \leftarrow \mathbf{X}^t$
$\tilde{\mathbf{X}}^t \leftarrow \mathbf{f}_{\text{cond}}(\mathbf{R}, \mathbf{X}^t)$
**for** $(\tau \leftarrow 0\,;\, \tau < 1\,;\, \tau \leftarrow \tau + d\tau)$ **do**
  $\mathbf{Z}^\tau \sim \mathcal{N}(0, \mathbb{I})$
  $\hat{\eta} \leftarrow \big(\hat{\eta}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{\eta}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
  $\hat{b} \leftarrow \big(\hat{b}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{b}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
  $d\mathbf{X}^\tau \leftarrow \big(\hat{b} - \frac{\epsilon(\tau)}{\gamma(\tau)}\hat{\eta}\big)d\tau + \sqrt{2\epsilon(\tau)}\mathbf{Z}^\tau$
  $\mathbf{X}_{\tau+d\tau}^t \leftarrow \mathbf{X}_\tau^t + d\mathbf{X}_\tau^t$
**return** $\mathbf{X}_{\tau=1}^t$

### 3.4 EQUILIBRATION OF MODEL DYNAMICS

To measure long-term behavior of our models, we estimate the stationary distribution of the learned dynamics and apply a correction to the density of sample observables. We leverage Time-lagged Independent Component Analysis (TICA) (Pérez-Hernández et al., 2013) and build Markov State Models (MSM) on clusters over TIC components. We obtain reference TICA components from the original trajectories considering a similarity based on the Euclidean distance between $C_\alpha$ and a lag time of 2ns. We find 100 clusters using k-means on the first 4 TIC dimensions, and build a Markov State Model (MSM) on the basis of these clusters by estimating the transition matrix at long time-lag (45 to 95ns). Finally, from the MSM largest eigenvectors we obtain the steady state probability

of each cluster, which is used to reweight the distribution of measurements from generated data, correcting the bias from simulation starting points.

## 4 Experiments and Results

To evaluate the capability of our model in reproducing protein dynamics, we leverage the large-scale dataset of 12 fast-folding proteins produced by (Majewski et al., 2023), and originally investigated in (Lindorff-Larsen et al., 2011). The dataset consists of millions of snapshots of MD for 12 proteins ranging from 10 to 80 residues, where in each trajectory snapshots are taken at the rate of 100 ps. We refer to the original work and Appendices A.3 and A.4 for more details.

### 4.1 Generative Transport Comparison

We perform a comparative analysis of our method against the baseline methods for protein simulation (Figure 2), evaluating their learned long-term dynamics on Protein G. We compare against DDPM (Ho et al., 2020; Arts et al., 2023; Schreiner et al., 2023), standard Flow Matching (Lipman et al., 2022; Li et al., 2024), and One-Sided Interpolants (Albergo et al., 2023a). In Figure 3, we show the free energies (MSM-reweighted, as in 3.4) along the first two TIC components (Pérez-Hernández et al., 2013) for best performing models. In Table 1, we compare the Jensen-Shannon divergence (Lin, 1991) of observables against reference for each model. These results reveal that the local transport of EquiJump is well-suited for the consecutive-step dynamics of MD trajectories.

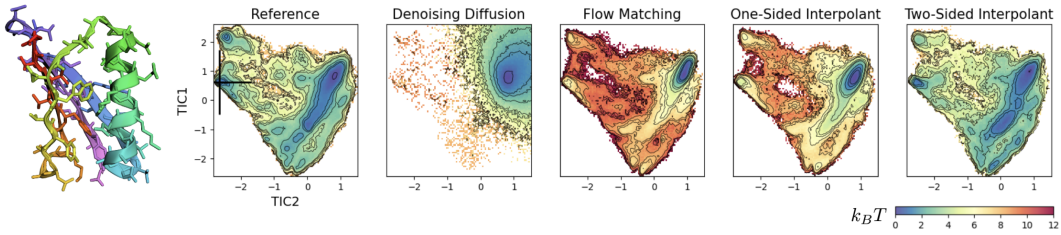


Figure 3: **Protein G and Free Energies on its TIC components for different transport**. (Left) Protein G crystal. (Right) Estimated free energies on the first TIC components for samples produced by DDPM, Flow Matching and Stochastic Interpolants.

| | **DDPM** | | | **Flow Matching** | | | **One-Sided Interpolant** | | | **EquiJump** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma_{\mathbf{P}}^2$ | 1 | 3 | 5 | 1 | 3 | 5 | 1 | 3 | 5 | **1** | 3 | 5 |
| **TIC1** | 0.215 | 0.178 | 0.216 | 0.055 | 0.104 | 0.049 | 0.022 | 0.028 | 0.072 | **0.004** | 0.010 | 0.070 |
| **TIC2** | 0.191 | 0.154 | 0.167 | 0.049 | 0.110 | 0.069 | 0.023 | 0.031 | 0.099 | **0.004** | 0.009 | 0.065 |
| **RMSD** | 0.219 | 0.316 | 0.297 | 0.219 | 0.341 | 0.160 | 0.118 | 0.164 | 0.237 | **0.008** | 0.017 | 0.103 |
| **GDT** | 0.267 | 0.253 | 0.226 | 0.164 | 0.264 | 0.110 | 0.091 | 0.122 | 0.176 | **0.008** | 0.012 | 0.088 |
| **RG** | 0.257 | 0.302 | 0.132 | 0.208 | 0.347 | 0.173 | 0.141 | 0.171 | 0.252 | **0.025** | 0.033 | 0.162 |
| **FNC** | 0.281 | 0.374 | 0.192 | 0.129 | 0.160 | 0.082 | 0.071 | 0.077 | 0.142 | **0.003** | 0.011 | 0.111 |

Table 1: **Jensen-Shannon Divergence** of key observables from reference density. **TIC1** and **TIC2**: first two TICA components. **RMSD**: Root Mean Square Deviation of $\mathbf{C}_\alpha$ atoms to crystal reference. **GDT**: Global Distance Test (Total Score) of $\mathbf{C}_\alpha$ atoms to crystal reference. **RG**: Radius of Gyration. **FNC**: Fraction of Native Contacts. Please refer to Appendix A.5 for detailed metrics descriptions.

### 4.2 Fast-Folding Transferable Model

We investigate the capabilities of our method on the challenging task of learning a stable and accurate dynamics simulator for all of the 12 fast-folding proteins using a single, transferable model. We systematically vary model capacity across latent dimensionalities $H = \{32, 64, 128, 256\}$ to assess the impact of model complexity on performance, further comparing our model to available transferable force-field model CG-MLFF (Majewski et al., 2023). To the best of our knowledge, this

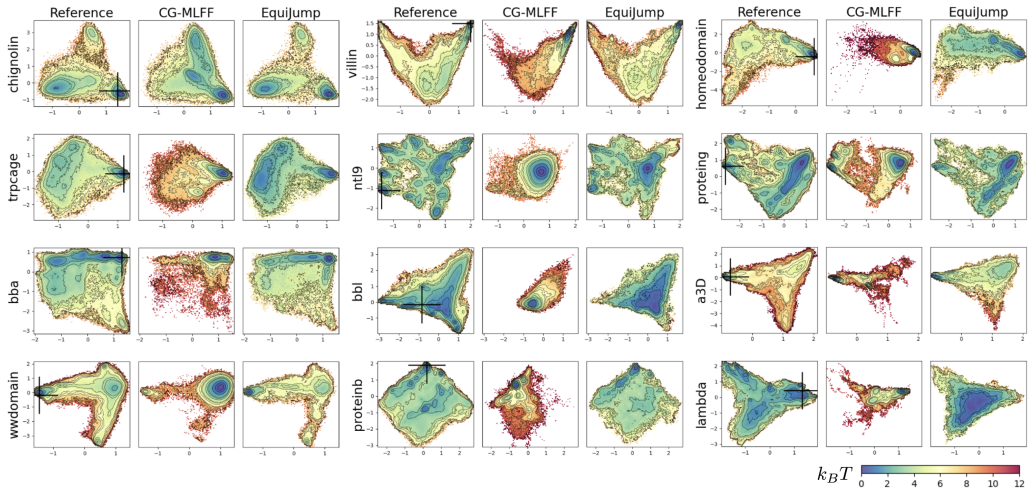


Figure 4: **Free Energy on TICA components for the 12 Fast-Folding Proteins**. We compare the free energy of EquiJump-256 against that of the reference and that of available model CG-MLFF. EquiJump succesfully recovers the dynamics of the proteins, covering the phase space and stabilizing the basin of most (shown as + in reference profile).

is the only other transferable model that covers these proteins. In Tables 2 and 3 and Figure 4, we investigate model performance in reproducing the long-time (MSM-reweighted, 3.4) distribution of observables and find that despite its large steps, our model shows better reconstruction of the slow components and more accurate profiling of the free-energy TICA maps across the studied proteins. We provide protein-specific results in Appendix A.8. We additionally verify the chemical validity (Appendix A.11) and analyze performance improvements of our approach (Appendix A.12).

| | | EquiJump | | | |
|---|---|---|---|---|---|
| | **CG-MLFF** | 32 | 64 | 128 | 256 |
| **TIC1** | 0.30 | 0.15 | 0.13 | 0.07 | **0.03** |
| **TIC2** | 0.23 | 0.17 | 0.09 | 0.06 | **0.03** |
| **RMSD** | 0.20 | 0.18 | 0.12 | 0.11 | **0.03** |
| **GDT** | 0.21 | 0.25 | 0.13 | 0.11 | **0.02** |
| **RG** | 0.18 | 0.14 | 0.08 | 0.12 | **0.04** |
| **FNC** | 0.27 | 0.25 | 0.13 | 0.08 | **0.03** |

Table 2: **Jensen-Shannon Divergence** of ensemble observables averaged over the 12 fast-folding proteins.

| | | EquiJump | | | |
|---|---|---|---|---|---|
| | **CG-MLFF** | 32 | 64 | 128 | 256 |
| **RMSD** | 34.7 | 51.2 | 46.9 | 43.6 | **15.2** |
| **GDT** | 51.5 | 57.1 | 42.7 | 38.0 | **18.3** |
| **RG** | 9.4 | 13.8 | 11.4 | 18.7 | **4.3** |
| **FNC** | 45.2 | 48.8 | 32.8 | 23.7 | **15.7** |

Table 3: **Percent Error in Predicting Averages** of ensemble observables for the 12 fast-folding proteins

## 5 Conclusion

In this paper we introduced EquiJump for learning the dynamics of 3D protein simulations. EquiJump extends Two-Sided Stochastic Interpolants for learning 3D dynamics through SO(3)-equivariant neural networks. We validated our approach on the large-scale dataset of fast-folding proteins, in which we compared generative frameworks and demonstrated a unified model that can accurately reproduce complex dynamics across the different proteins. Our experiments suggest EquiJump provides a stepping stone for future research in modeling and accelerating protein dynamics simulation.

## Acknowledgments

We thank NVIDIA, the Eleven Eleven Foundation, the Center for Bits and Atoms, and the MIT Media Lab Consortium for their essential support in this research.

## REFERENCES


Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. (2023a). Stochastic interpolants: A unifying framework for flows and diffusions.

Albergo, M. S., Goldstein, M., Boffi, N. M., Ranganath, R., and Vanden-Eijnden, E. (2023b). Stochastic interpolants with data-dependent couplings.

Albergo, M. S. and Vanden-Eijnden, E. (2023). Building normalizing flows with stochastic interpolants.

Arts, M., Garcia Satorras, V., Huang, C.-W., Zügner, D., Federici, M., Clementi, C., Noé, F., Pinsler, R., and van den Berg, R. (2023). Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. *Journal of Chemical Theory and Computation*, 19(18):6151–6159.

Batatia, I., Kovacs, D. P., Simm, G., Ortner, C., and Csányi, G. (2022). Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35:11423–11436.

Best, R. B., Hummer, G., and Eaton, W. A. (2013). Native contacts determine protein folding mechanisms in atomistic simulations. *Proceedings of the National Academy of Sciences*, 110(44):17874–17879.

Costa, A. D. S., Mitnikov, I., Geiger, M., Ponnapati, M., Smidt, T., and Jacobson, J. (2024). Ophiuchus: Scalable modeling of protein structures through hierarchical coarse-graining SO(3)-equivariant autoencoders. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*.

Daigavane, A., Vani, B. P., Saremi, S., Rackers, J. A., and Kleinhenz, J. (2024). JAMUN: Transferable molecular conformational ensemble generation with walk-jump sampling. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*.

Durumeric, A. E., Charron, N. E., Templeton, C., Musil, F., Bonneau, K., Pasos-Trejo, A. S., Chen, Y., Kelkar, A., Noé, F., and Clementi, C. (2023). Machine learned coarse-grained protein force-fields: Are we there yet? *Current Opinion in Structural Biology*, 79:102533.

Ermak, D. L. and McCammon, J. A. (1978). Brownian dynamics with hydrodynamic interactions. *The Journal of chemical physics*, 69(4):1352–1360.

Exxact Corp. (2024). Amber 24 nvidia gpu benchmarks. `https://www.exxactcorp.com/blog/molecular-dynamics/amber-molecular-dynamics-nvidia-gpu-benchmarks`.

Fu, X., Xie, T., Rebello, N. J., Olsen, B., and Jaakkola, T. S. (2023). Simulate time-integrated coarse-grained molecular dynamics with multi-scale graph networks. *Transactions on Machine Learning Research*.

Gabrié, M., Rotskoff, G. M., and Vanden-Eijnden, E. (2022). Adaptive monte carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10):e2109420119.

Geiger, M. and Smidt, T. (2022). e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*.

Han, J., Xu, M., Lou, A., Ye, H., and Ermon, S. (2024). Geometric trajectory diffusion models. *arXiv preprint arXiv:2410.13027*.

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

Hollingsworth, S. A. and Dror, R. O. (2018). Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143.

Husic, B. E., Charron, N. E., Lemm, D., Wang, J., Pérez, A., Majewski, M., Krämer, A., Chen, Y., Olsson, S., de Fabritiis, G., et al. (2020). Coarse graining molecular dynamics with graph neural networks. *The Journal of chemical physics*, 153(19).

Jing, B., Berger, B., and Jaakkola, T. (2024a). Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*.

Jing, B., Stark, H., Jaakkola, T., and Berger, B. (2024b). Generative modeling of molecular dynamics trajectories. In *ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications*.

Karplus, M. and Kuriyan, J. (2005). Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences*, 102(19):6679–6685.

King, J. E. and Koes, D. R. (2020). Sidechainnet: An all-atom protein structure dataset for machine learning.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Klein, L., Foong, A. Y. K., Fjelde, T. E., Mlodozeniec, B., Brockschmidt, M., Nowozin, S., Noé, F., and Tomioka, R. (2023). Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. In *NeurIPS 2023*.

Kovács, D. P., Moore, J. H., Browning, N. J., Batatia, I., Horton, J. T., Kapil, V., Witt, W. C., Magdău, I.-B., Cole, D. J., and Csányi, G. (2023). Mace-off23: Transferable machine learning force fields for organic molecules.

Köhler, J., Chen, Y., Krämer, A., Clementi, C., and Noé, F. (2023). Flow-matching: Efficient coarse-graining of molecular dynamics without forces. *Journal of Chemical Theory and Computation*, 19(3):942–952.

Laio, A. and Parrinello, M. (2002). Escaping free-energy minima. *Proceedings of the national academy of sciences*, 99(20):12562–12566.

Lane, T. J., Shukla, D., Beauchamp, K. A., and Pande, V. S. (2013). To milliseconds and beyond: challenges in the simulation of protein folding. *Current opinion in structural biology*, 23(1):58–65.

Lewis, S., Hempel, T., Jiménez Luna, J., Gastegger, M., Xie, Y., Foong, A. Y., García Satorras, V., Abdin, O., Veeling, B. S., Zaporozhets, I., et al. (2024). Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, pages 2024–12.

Li, S., Wang, Y., Li, M., Zhang, J., Shao, B., Zheng, N., and Tang, J. (2024). $F^3$low: Frame-to-frame coarse-grained molecular dynamics with se(3) guided flow matching.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Lindorff-Larsen, K., Piana, S., Dror, R. O., and Shaw, D. E. (2011). How fast-folding proteins fold. *Science*, 334(6055):517–520.

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

Liu, X., Gong, C., and Liu, Q. (2023). Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.

Luhman, E. and Luhman, T. (2021). Knowledge distillation in iterative generative models for improved sampling speed.

Luo, S., Xu, Y., He, D., Zheng, S., Liu, T.-Y., and Wang, L. (2024). Bridging geometric states via geometric diffusion bridge.

Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. (2024). Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*.

Majewski, M., Pérez, A., Thölke, P., Doerr, S., Charron, N. E., Giorgino, T., Husic, B. E., Clementi, C., Noé, F., and De Fabritiis, G. (2023). Machine learning coarse-grained potentials of protein thermodynamics. *Nature Communications*, 14(1):5739.

Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. (2020). Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*.

Noé, F., De Fabritiis, G., and Clementi, C. (2020). Machine learning for protein folding and dynamics. *Current opinion in structural biology*, 60:77–84.

Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G., and Noé, F. (2013). Identification of slow molecular order parameters for markov model construction. *The Journal of chemical physics*, 139(1).

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models.

Saremi, S. and Hyvärinen, A. (2019). Neural empirical bayes. *Journal of Machine Learning Research*, 20(181):1–23.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR.

Schlick, T. (2010). *Molecular modeling and simulation: an interdisciplinary guide*, volume 2. Springer.

Schreiner, M., Winther, O., and Olsson, S. (2023). Implicit transfer operator learning: Multiple time-resolution surrogates for molecular dynamics.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. (2018). Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*.

Torrie, G. M. and Valleau, J. P. (1977). Nonphysical sampling distributions in monte carlo free-energy estimation: Umbrella sampling. *Journal of computational physics*, 23(2):187–199.

Wang, J., Olsson, S., Wehmeyer, C., Pérez, A., Charron, N. E., De Fabritiis, G., Noé, F., and Clementi, C. (2019). Machine learning of coarse-grained molecular dynamics force fields. *ACS central science*, 5(5):755–767.

Zemla, A. (2003). Lga: a method for finding 3d similarities in protein structures. *Nucleic acids research*, 31(13):3370–3374.

Zheng, Q., Le, M., Shaul, N., Lipman, Y., Grover, A., and Chen, R. T. (2023). Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*.

# A Appendix / supplemental material

## A.1 Brownian Dynamics in Protein-Solvent Systems and Connection to Stochastic Interpolants

In the study of molecular dynamics of proteins immersed in solvents, it is crucial to account for the interactions between the proteins and the surrounding fluid. Proteins in a solvent experience random collisions with solvent molecules, leading to stochastic behavior that can be effectively modeled using Brownian dynamics (Ermak and McCammon, 1978). This approach captures the random motion arising from thermal fluctuations and solvent effects, providing a realistic depiction of protein behavior in biological environments.

Generalized frictional interactions among the particles can be incorporated into the Langevin equation through a friction tensor $\mathbf{R}$ (Schlick, 2010). This tensor accounts for the action of the solvent on the particles and modifies the Langevin equation to:

$$\mathbf{M}\ddot{\mathbf{X}}(t) = -\nabla E(\mathbf{X}(t)) - \mathbf{R}\dot{\mathbf{X}}(t) + \mathbf{W}(t), \tag{9}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{X}(t)$ represents the particle positions at time $t$, $E(\mathbf{X}(t))$ is the potential energy, and $\mathbf{W}(t)$ is a random force representing thermal fluctuations from the solvent. The mean and covariance of the random force $\mathbf{W}(t)$ are given by:

$$\langle \mathbf{W}(t) \rangle = 0, \quad \langle \mathbf{W}(t)\mathbf{W}(t')^T \rangle = 2k_B T \mathbf{R}\delta(t - t'), \tag{10}$$

where $k_B$ is Boltzmann's constant, $T$ is the temperature, and $\delta(t - t')$ is the Dirac delta function. This relation is based on the fluctuation-dissipation theorem, a fundamental result that connects the friction experienced by a particle to the fluctuations of the random force acting upon it, assuming the particle is undergoing random motion around thermal equilibrium.

This description ensures that the ensemble of trajectories generated from Eq. 9 is governed by the Fokker-Planck equation, a partial differential equation that describes the evolution of the probability density function of a particle's position and momentum in phase space during diffusive motion. In this context, the random force $\mathbf{W}(t)$ is modeled as white noise with no intrinsic timescale. The inertial relaxation times, given by the inverses of the eigenvalues of the matrix $\mathbf{M}^{-1}\mathbf{R}$, define the characteristic timescale of the thermal motion. When these inertial relaxation times are short compared to the timescale of interest, it is appropriate to neglect inertia in the governing equation, effectively discarding the acceleration term by assuming $\mathbf{M}\ddot{\mathbf{X}}(t) \approx 0$. Under this approximation, Eq. 9 simplifies to the Brownian dynamics form:

$$\dot{\mathbf{X}}(t) = -\mathbf{R}^{-1}\nabla E(\mathbf{X}(t)) + \mathbf{R}^{-1}\mathbf{W}(t). \tag{11}$$

This simplification reflects that solvent effects are sufficiently strong to render inertial forces negligible, resulting in motion that is predominantly Brownian and stochastic in nature. This description is particularly effective for modeling very large, dense systems whose conformations in solution are continuously and significantly altered by the fluid flow in their environment. To stably evolve Brownian dynamics eq. (11) over time, small integration steps are usually required due to the stiffness of the physical manifold. Molecular systems exhibit a wide range of timescales: fast atomic motions such as bond vibrations and thermal fluctuations occur on the order of femtoseconds, while slower conformational shifts and larger-scale rearrangements may take place over much longer periods. These necessitate small time steps to accurately capture the system's rapid changes without numerical instability. In contrast, Stochastic Interpolants eq. (8), while also following the form of eq. (11), enable smoothing of the data manifold by convolution with small Gaussian perturbations. This leads to a latent representation that is robust to noise, allowing for larger integration steps. The smoother manifold helps overcome local energy barriers and navigate the broader conformational landscape more efficiently, making it possible to simulate molecular dynamics on extended timescales without losing stability.

## A.2 EQUIJUMP MODEL

### A.2.1 MODEL ARCHITECTURE

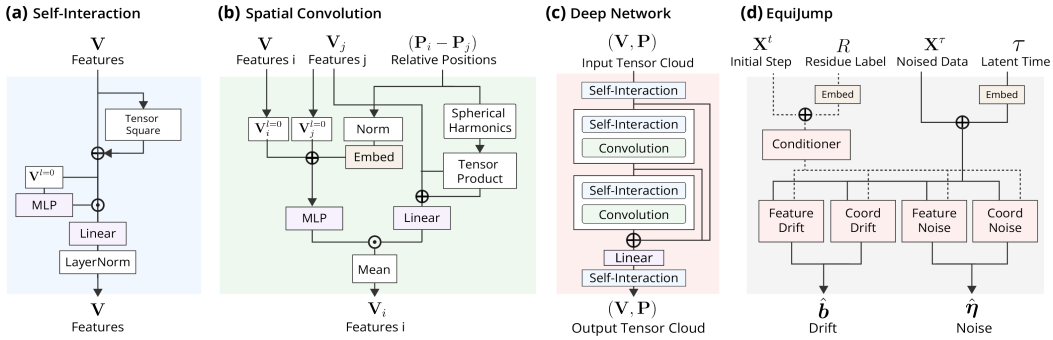


Figure 5: **EquiJump Architecture**: **(a)** The **Self-Interaction** Layer updates geometric features independently, mixing $\mathbf{V}^l$ of different degrees into new features through a Tensor Square operation. **(b)** The **Spatial Convolution** layer updates representations by aggregating the tensor product of neighbors messages with the spherical harmonics embedding of the relative 3D vector between the positions of those neighbors. **(c)** We stack the above modules to form a block, and build a base network out of $L$ blocks for making predictions. **(d)** A shared conditioner and 4 headers are built from the base network. The conditioner processes sequence and the current simulation step, producing latent embeddings that are fed to the prediction headers. The headers independently predict features and coordinates updates for drift and noise components of the stochastic process.

For training all models we use the Adam optimizer (Kingma and Ba, 2017) with linearly decreasing learning rate from $1 \times 10^{-2}$ to $1 \times 10^{-3}$ over 150k steps. We perform all experiments on NVIDIA A100 machines with 2-4 GPUs.

The EquiJump block is built from two SO(3)-equivariant networks: the Self-Interaction (Figure 5.a) for updating features $\mathbf{V}^l$ independently, based on MACE (Batatia et al., 2022); and the Spatial Convolution (Figure 5.b) for sharing information across neighbors, based on Tensor Field Networks (Thomas et al., 2018). We build a deep neural network (DNN) by stacking $L$ of these blocks (Figure 5.c). We use 5 DNNs in our model (Figure 5.d): 1 conditioner network $\mathbf{f}_{\text{cond}}$ and 4 header networks for predicting each of $\hat{b}_{\mathbf{V}}$, $\hat{b}_{\mathbf{P}}$, $\hat{\eta}_{\mathbf{V}}$, $\hat{\eta}_{\mathbf{P}}$ independently.

Algorithms 3, 4, 5 describe the components of EquiJump. The Tensor Square operation in Alg. 3 (line 1) is applied independently within each channel. The residual sum of Alg. 5 (line 5) is only performed on the geometric features, since positions are fixed. Tested models employ irreps of `0e + 1e` across multiplicities $\{32, 64, 128, 256\}$. We only test conditional number of layers $L_{\text{cond}} = 6$, and header number of layers $L_{\text{header}} = 4$. Our experimentation indicates further scaling is a promising direction of research.

For interpolant parameterization we use $I(\tau, \mathbf{X}_0, \mathbf{X}_1) = (1-\tau)\mathbf{X}_0 + \tau\mathbf{X}_1$, $\gamma(\tau) = \sigma \cdot \tau \cdot (1-\tau)$ and fixed time dependent diffusion coefficient $\epsilon(\tau) = 1.0$ in sampling. Where $\sigma = 3.0, 1.0$ for the coordinates and geometric features, respectively. In future work we will investigate how different interpolant parameterizations affect the performance of our model. Independent treatment of feature and coordinate components enables different interpolants for each term. In this work, we use the same interpolant for both, only adjusting the variances $(\sigma^2_{\mathbf{V}}, \sigma^2_{\mathbf{P}})$ in sampling the variable $\mathbf{Z}$. For training the transferable model, we use $\sigma^2_{\mathbf{P}} = 3.0$ and $\sigma^2_{\mathbf{V}} = 1.0$.

**Algorithm 3** Self-Interaction

**Require:** Tensor Cloud $(\mathbf{P}, \mathbf{V})$
1: $\mathbf{V} \leftarrow \mathbf{V} \oplus (\mathbf{V})^{\otimes 2}$
2: $\mathbf{V} \leftarrow \text{MLP}(\mathbf{V}^{l=0}) \cdot \mathbf{V}$
3: $\mathbf{V} \leftarrow \text{Linear}(\mathbf{V})$
4: **return** $(\mathbf{P}, \mathbf{V})$

**Algorithm 4** Spatial Convolution

**Require:** Tensor Cloud $(\mathbf{V}, \mathbf{P})$
**Require:** Output Node Index $i$
$(\tilde{\mathbf{P}}, \tilde{\mathbf{V}})_{1:k} \leftarrow k\text{NN}(\mathbf{P}_i, \mathbf{P}_{1:N})$
$R_{1:k} \leftarrow \text{Embed}(||\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i||_2)$,
$\phi_{1:k} \leftarrow \text{SphericalHarmonics}(\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i)$
$\tilde{\mathbf{V}}_{1:k} \leftarrow \text{MLP}\big(R_k \oplus \mathbf{V}^{l=0}_{1:k} \oplus \mathbf{V}^{l=0}\big) \cdot \text{Linear}\big(\tilde{\mathbf{V}}_{1:k} \otimes \phi_{1:k}\big)$
$\mathbf{V} \leftarrow \text{Linear}\left(\mathbf{V} + \frac{1}{k}\big(\sum_k \tilde{\mathbf{V}}_k\big)\right)$
**return** $(\mathbf{V}, \mathbf{P})$

**Algorithm 5** EquiJump Deep Network

**Require:** Tensor Cloud $\mathbf{X} = (\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
$\mathbf{H}^0 \leftarrow \text{Self-Interaction}(\mathbf{X})$
**for** $l$ in [0, L) **do**
    $\mathbf{H}^{l+1} \leftarrow \text{Self-Interaction}(\mathbf{H}^l)$
    $\mathbf{H}^{l+1} \leftarrow \text{SpatialConvolution}(\mathbf{H}^{l+1})$
    $\mathbf{H}^{l+1} \leftarrow \text{LayerNorm}(\mathbf{H}^{l+1} + \mathbf{H}^l)$
$\mathbf{H}^{\text{agg}} \leftarrow \text{Linear}\big(\bigoplus_{l=0}^{L-1} \mathbf{H}^l\big)$
$\mathbf{H}^{\text{out}} \leftarrow \text{Self-Interaction}(\mathbf{H}^{\text{agg}})$
**return** $\mathbf{H}^{\text{out}}$

### A.2.2 EQUIVARIANCE

The features used in EquiJump are irreducible representations (irreps) of $\text{SO}(3)$. To prove that EquiJump is equivariant, we demonstrate that all operations within the network preserve the transformation properties of the irreps under rotation.

Scalars, which are irreps of degree $l = 0$, remain invariant under rotation. For a scalar $s \in \mathbb{R}$ and a rotation $R \in \text{SO}(3)$, we have:

$$R \cdot s = s. \tag{12}$$

Applying functions such as MLPs to scalars preserves this invariance:

$$f(R \cdot s) = f(s). \tag{13}$$

For irreps with $l > 0$, equivariance depends on the nature of the operations. Linear operations combine irreps of the same degree using scalar weights. Let $\mathbf{v}$ and $\mathbf{w}$ be irreps and $W$ a learnable weight matrix. Under a rotation $R$,

$$R \cdot (W\mathbf{v}) = W(R \cdot \mathbf{v}), \tag{14}$$

where $R \cdot \mathbf{v}$ represents the rotated vector. Since the weight matrix $W$ does not interfere with the transformation properties, linear operations are equivariant.

EquiJump also employs tensor products, which combine two irreps $\mathbf{v}$ and $\mathbf{w}$ of degrees $l_1$ and $l_2$, respectively, to produce new irreps of degrees $|l_1 - l_2|, \ldots, (l_1 + l_2)$. The tensor product transforms under rotation as:

$$R \cdot (\mathbf{v} \otimes \mathbf{w}) = (R \cdot \mathbf{v}) \otimes (R \cdot \mathbf{w}), \tag{15}$$

ensuring equivariance. In EquiJump, tensor products are used in two key cases: (1) between features and spherical harmonics $Y_{lm}(\mathbf{r})$ of relative positions, and (2) between features and themselves (tensor square). Spherical harmonics transform under rotation as:

$$R \cdot Y_{lm}(\mathbf{r}) = \sum_{m'} D^{(l)}_{mm'}(R) Y_{lm'}(\mathbf{r}), \tag{16}$$

where $D^{(l)}_{mm'}(R)$ are elements of the Wigner-D matrix. Combining features with spherical harmonics via tensor products preserves equivariance by construction.

The basic operation in EquiJump involves a tensor product followed by a linear transformation. Let $T$ represent this combined operation:

$$T(\mathbf{v}, \mathbf{w}) = W(\mathbf{v} \otimes \mathbf{w}), \tag{17}$$

where $W$ is a learnable weight tensor. Under a rotation $R$,

$$R \cdot T(\mathbf{v}, \mathbf{w}) = W(R \cdot (\mathbf{v} \otimes \mathbf{w})) = W((R \cdot \mathbf{v}) \otimes (R \cdot \mathbf{w})). \tag{18}$$

This demonstrates that this basic operation is equivariant. Since scalar transformations, linear layers, and tensor products all preserve equivariance, the entire EquiJump network is $\mathrm{SO}(3)$-equivariant. This ensures that outputs transform consistently with inputs under rotation, making EquiJump well-suited for modeling the rotationally symmetric dynamics of protein structures in 3D space.

### A.3 DATASET

We adapt the dataset produced by (Majewski et al., 2023). The trajectories are made up of several NVT runs (20 to 100 ns) at $T = 350$ K from different starting configurations spanning the phase space. The dataset consists of trajectories of $\approx 500$ steps at intervals of 100ps. Refer to the table below for the number of trajectories curated. To include all relevant residues, in addition to the standard vocabulary of residues, we also include a canonical form of Norleucine (NLE).

| Protein | Residues | Trajectories |
|---|---|---|
| Chignolin | 10 | 3744 |
| Trp-Cage | 20 | 3940 |
| BBA | 28 | 7297 |
| Villin | 34 | 17103 |
| WW domain | 35 | 2347 |
| NTL9 | 39 | 7651 |
| BBL | 47 | 18033 |
| Protein B | 47 | 6094 |
| Homeodomain | 54 | 1991 |
| Protein G | 56 | 11272 |
| a3D | 73 | 7113 |
| $\lambda$-repressor | 80 | 15697 |

### A.4 CLUSTER-ENHANCED DATASET SAMPLING

While sampling uniformly on the whole dataset is effective for learning transitions, it is not efficient due to slow modes and low frequency states. For the slowest modes of the system, states are very high in free energy and heavily under-represented in our training dataset. To address this issue, taking inspiration by classical sampling methods such as umbrella sampling (Torrie and Valleau, 1977) and metadynamics (Laio and Parrinello, 2002) we propose a reweighing of the training set. Notably, since our model learns $\rho(\mathbf{X}^{t+1} \mid \mathbf{X}^t)$, the target distribution remains unchanged as long as we fix the endpoints $(\mathbf{X}^t, \mathbf{X}^{t+1})$: this sampling only varies the speed at which different parts of the phase space are learned. We first find relevant degrees of freedom through TICA analysis (Pérez-Hernández et al., 2013), which offers a reduced dimensional space that highlights the slow macroscopic modes of the system. We then fit a small number of clusters through k-means in this simplified space using the first two TIC components. Our enhanced dataset first samples a cluster, then from the cluster a configuration and its transition. We fit 200 clusters for each protein. Figure 6 visualizes cluster centers and the distribution of population sizes.

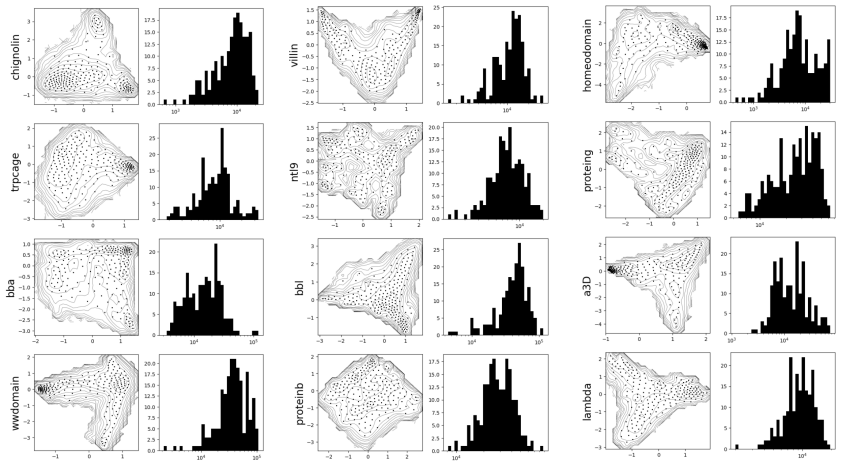


Figure 6: **Cluster Centers and Distribution of Cluster Population Sizes**.

### A.5 METRICS

We use the following metrics for assessment as commonly used for protein dynamics analysis:

**TICs (Time-lagged Independent Components):** Derived from Time-lagged Independent Component Analysis (TICA), TICs project molecular dynamics data into a reduced-dimensional space that emphasizes slow collective motions (Pérez-Hernández et al., 2013). We fit TICA on ground truth and project samples to make comparisons. We consider the two main TIC components for our metrics. For embedding our proteins for TICA, we use a distance matrix considering only $C_\alpha$ positions. We use lagtime of 2ns.

**RMSD (Root Mean Square Deviation):** RMSD measures the average deviation of atomic positions from a reference structure:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i^{\text{model}} - \mathbf{x}_i^{\text{ref}}\|^2}, \tag{19}$$

where $N$ is the number of atoms. We align $\mathbf{C}_\alpha$ backbone samples to reference crystal structure and measure the resulting RMSD for obtaining our metric.

**GDT (Global Distance Test):** GDT quantifies the fraction of residues within specific distance thresholds of the reference structure:

$$\text{GDT} = \frac{1}{4}\sum_{d\in\{1,2,4,8\}} \frac{\text{\# of residues within } d\,\text{Å}}{\text{total residues}}, \tag{20}$$

and is used in protein structure alignment (Zemla, 2003). We align the $\mathbf{C}_\alpha$ of our structures to reference crystal structure and measure GDT for obtaining our reported values.

**RG (Radius of Gyration):** The compactness of a protein structure is measured as:

$$R_g = \sqrt{\frac{\sum_{i=1}^{N} m_i \|\mathbf{x}_i - \mathbf{x}_{\text{COM}}\|^2}{\sum_{i=1}^{N} m_i}}, \tag{21}$$

where $m_i$ and $\mathbf{x}_i$ are the mass and position of atom $i$, and $\mathbf{x}_{\text{COM}}$ is the center of mass.

**FNC (Fraction of Native Contacts):** FNC evaluates the fraction of interatomic contacts preserved from the reference structure:

$$\text{FNC} = \frac{\text{\# of native contacts in the sample}}{\text{\# of native contacts in the reference crystal}}. \tag{22}$$

This metric highlights native structure preservation (Best et al., 2013).

## A.6 EXPERIMENTS

### A.6.1 GENERATIVE MODEL COMPARISON

For this evaluation, we vary the latent variable noise variance of positions $\sigma^2_{\mathbf{P}} = \{1, 3, 5\}$ while keeping the features noise variance fixed at $\sigma^2_{\mathbf{V}} = 1$. For comparison, we adapt our network and only use 2 headers (noise or drift) for DDPM and Flow Matching. We train all models with $H = 32$ for 200k steps and batch size 128. For each model, we sample 1000 trajectories of 500 steps (50 ns) with 100 steps of latent variable integration.

### A.6.2 12 FAST-FOLDERS TRANSFERABLE MODEL

We train all models for 500k steps with batch size 128. For collecting samples, we perform 500 simulations of 500 steps (50 ns) starting from states of the (enhanced) dataset. We employ 100 steps of integration to obtain the next configuration. Our results demonstrate that while the force field-based model is competitive in low-capacity regimes, our long-interval generative model significantly outperforms it in higher-capacity settings. This can be attributed to the fact that force-field methods are constrained to small time steps and only require short-term, local predictions which can be captured with fewer model parameters. In contrast, a model capable of large time steps must possess a deep understanding of the underlying data manifold, as the number of plausible transition states grows significantly with increasing time step size. While EquiJump requires substantial capacity to precisely reproduce equilibrium behavior, it successfully navigates the manifold across model sizes (Appendix A.9), while achieving overall superior performance.

### A.7 VISUALIZATION OF SAMPLES

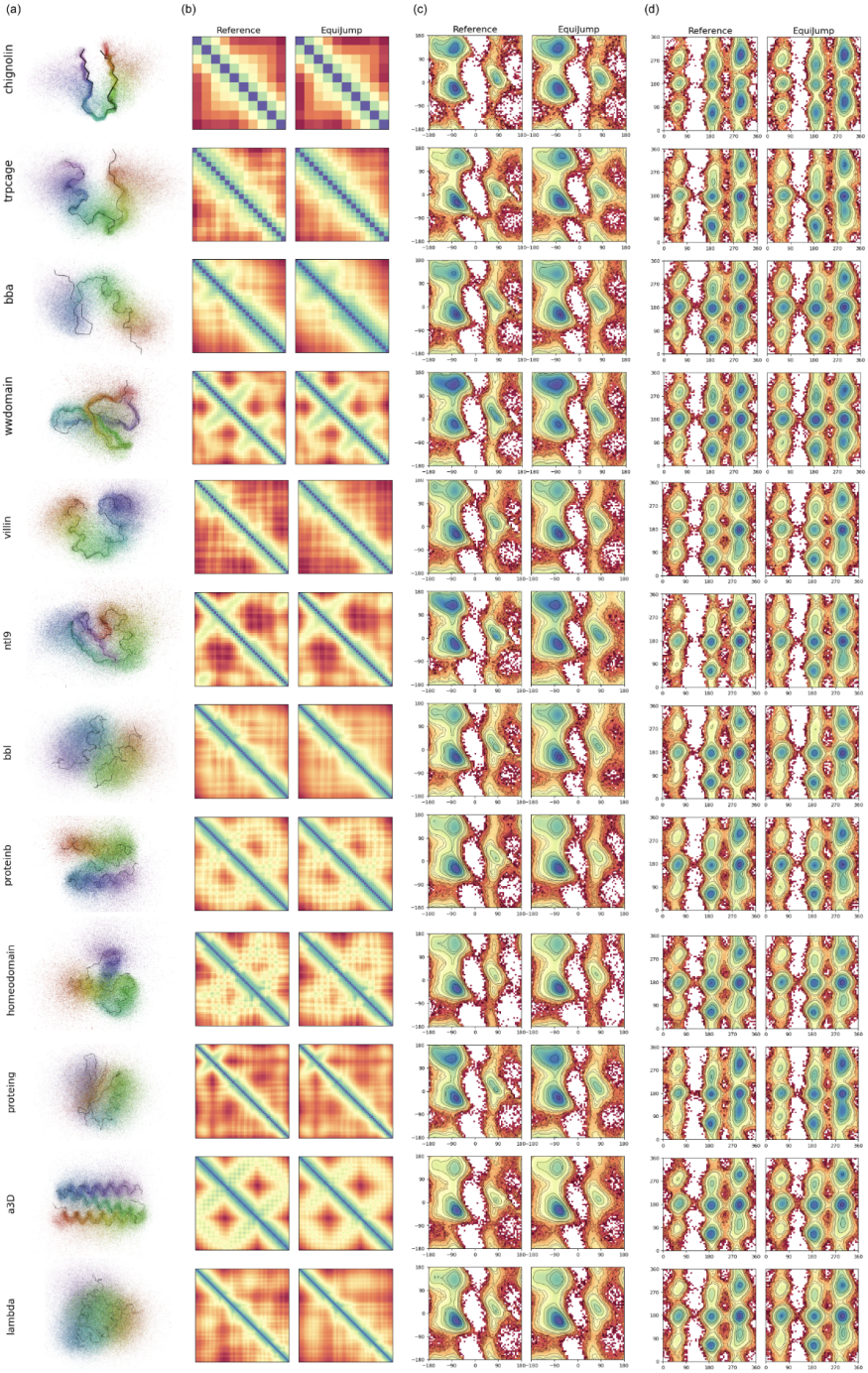


Figure 7: **EquiJump Samples**: **(a)** We visualize the performance of EquiJump on fast-folding proteins by superposing 1500 backbone random samples of EquiJump trajectories. We align samples to the crystal backbone (shown in black). We further present **(b)** mean pairwise $\mathbf{C}_\alpha$ distance matrices, **(c)** Ramachandran plots of backbone dihedrals and **(d)** Janin plots of sidechain dihedrals of EquiJump samples against reference trajectory data.

### A.8 PROTEIN-SPECIFIC ABLATION RESULTS

We report Jensen-Shannon Divergence (Lin, 1991) against reference trajectories for reweighted ensemble observables of each protein, comparing CG-MLFF (Majewski et al., 2023) and EquiJump at various capacities to assess their ability to replicate realistic structural and dynamic properties.

| | | | **EquiJump** | | | |
|---|---|---|---|---|---|---|
| | | **CG-MLFF** | 32 | 64 | 128 | 256 |
| chignolin | TIC1 | 0.221 | 0.026 | 0.069 | 0.009 | **0.006** |
| | TIC2 | 0.152 | 0.019 | 0.039 | **0.003** | 0.006 |
| | RMSD | 0.253 | 0.028 | 0.083 | **0.012** | 0.018 |
| | GDT | 0.231 | 0.018 | 0.080 | **0.010** | 0.013 |
| | RG | 0.191 | 0.029 | 0.061 | **0.008** | 0.020 |
| | FNC | 0.189 | 0.024 | 0.069 | **0.007** | 0.012 |
| trpcage | TIC1 | 0.372 | 0.115 | 0.107 | 0.048 | **0.019** |
| | TIC2 | 0.187 | 0.037 | 0.047 | 0.068 | **0.008** |
| | RMSD | 0.283 | 0.051 | 0.038 | **0.011** | 0.022 |
| | GDT | 0.302 | 0.066 | 0.042 | **0.013** | 0.021 |
| | RG | 0.438 | 0.045 | 0.028 | **0.007** | 0.035 |
| | FNC | 0.299 | 0.100 | 0.096 | 0.036 | **0.031** |
| bba | TIC1 | 0.334 | 0.110 | 0.067 | 0.114 | **0.044** |
| | TIC2 | 0.169 | 0.132 | **0.012** | 0.022 | 0.017 |
| | RMSD | **0.022** | 0.102 | 0.048 | 0.211 | 0.025 |
| | GDT | 0.037 | 0.339 | 0.045 | 0.248 | **0.029** |
| | RG | 0.185 | 0.086 | **0.024** | 0.271 | 0.026 |
| | FNC | 0.200 | 0.279 | 0.086 | 0.143 | **0.026** |
| wwdomain | TIC1 | 0.252 | 0.191 | 0.061 | 0.048 | **0.028** |
| | TIC2 | 0.072 | 0.065 | 0.047 | 0.037 | **0.014** |
| | RMSD | 0.246 | 0.226 | 0.091 | 0.139 | **0.022** |
| | GDT | 0.263 | 0.240 | 0.093 | 0.127 | **0.021** |
| | RG | 0.084 | 0.161 | 0.064 | 0.173 | **0.018** |
| | FNC | 0.264 | 0.294 | 0.100 | 0.090 | **0.021** |
| villin | TIC1 | 0.347 | 0.181 | 0.091 | 0.020 | **0.015** |
| | TIC2 | 0.340 | 0.162 | 0.078 | **0.016** | 0.019 |
| | RMSD | 0.253 | 0.149 | 0.079 | 0.035 | **0.016** |
| | GDT | 0.293 | 0.151 | 0.088 | 0.028 | **0.015** |
| | RG | 0.240 | 0.101 | 0.054 | 0.079 | **0.019** |
| | FNC | 0.300 | 0.149 | 0.064 | **0.015** | 0.020 |
| ntl9 | TIC1 | 0.251 | 0.270 | 0.207 | 0.072 | **0.045** |
| | TIC2 | 0.270 | 0.287 | 0.225 | 0.078 | **0.073** |
| | RMSD | 0.192 | 0.283 | 0.191 | 0.101 | **0.059** |
| | GDT | 0.170 | 0.242 | 0.156 | 0.069 | **0.044** |
| | RG | **0.019** | 0.187 | 0.130 | 0.121 | 0.050 |
| | FNC | 0.172 | 0.383 | 0.240 | 0.054 | **0.038** |
| bbl | TIC1 | 0.402 | 0.124 | 0.062 | 0.069 | **0.033** |
| | TIC2 | 0.229 | 0.224 | 0.063 | 0.137 | **0.036** |
| | RMSD | 0.378 | 0.053 | 0.016 | 0.135 | **0.011** |
| | GDT | 0.409 | 0.055 | **0.008** | 0.132 | **0.008** |
| | RG | 0.207 | 0.042 | **0.013** | 0.140 | 0.018 |
| | FNC | 0.445 | 0.237 | 0.057 | 0.134 | **0.029** |
| proteinb | TIC1 | 0.377 | 0.055 | 0.040 | 0.041 | **0.008** |
| | TIC2 | 0.332 | 0.115 | 0.062 | 0.054 | **0.008** |
| | RMSD | 0.214 | 0.265 | 0.156 | 0.178 | **0.007** |
| | GDT | 0.240 | 0.277 | 0.162 | 0.181 | **0.007** |

| | | | **EquiJump** | | | |
|---|---|---|---|---|---|---|
| | | **CG-MLFF** | 32 | 64 | 128 | 256 |
| | RG | 0.247 | 0.247 | 0.121 | 0.190 | **0.044** |
| | FNC | 0.313 | 0.189 | 0.095 | 0.095 | **0.004** |
| homeodomain | TIC1 | 0.250 | 0.308 | 0.260 | 0.203 | **0.081** |
| | TIC2 | 0.183 | 0.144 | 0.130 | 0.117 | **0.044** |
| | RMSD | 0.150 | 0.414 | 0.298 | 0.321 | **0.068** |
| | GDT | 0.189 | 0.468 | 0.349 | 0.355 | **0.078** |
| | RG | **0.051** | 0.288 | 0.195 | 0.313 | 0.137 |
| | FNC | 0.246 | 0.378 | 0.257 | 0.261 | **0.089** |
| proteing | TIC1 | 0.180 | 0.077 | 0.127 | 0.034 | **0.009** |
| | TIC2 | 0.212 | 0.602 | 0.154 | 0.037 | **0.012** |
| | RMSD | 0.075 | 0.175 | 0.101 | 0.079 | **0.019** |
| | GDT | 0.103 | 0.628 | 0.106 | 0.067 | **0.013** |
| | RG | 0.079 | 0.191 | 0.069 | 0.105 | **0.040** |
| | FNC | 0.241 | 0.386 | 0.155 | 0.039 | **0.011** |
| a3d | TIC1 | 0.348 | 0.336 | 0.356 | 0.095 | **0.072** |
| | TIC2 | 0.319 | 0.130 | 0.099 | 0.055 | **0.034** |
| | RMSD | 0.107 | 0.352 | 0.336 | 0.074 | **0.070** |
| | GDT | 0.112 | 0.355 | 0.339 | 0.072 | **0.057** |
| | RG | 0.371 | 0.234 | 0.173 | 0.099 | **0.038** |
| | FNC | 0.224 | 0.311 | 0.286 | 0.079 | **0.055** |
| lambda | TIC1 | 0.330 | 0.109 | 0.159 | **0.107** | 0.116 |
| | TIC2 | 0.338 | 0.210 | 0.144 | 0.100 | **0.091** |
| | RMSD | 0.311 | 0.129 | 0.109 | **0.033** | 0.046 |
| | GDT | 0.277 | 0.167 | 0.095 | **0.028** | 0.042 |
| | RG | 0.157 | 0.137 | 0.131 | **0.046** | 0.053 |
| | FNC | 0.382 | 0.277 | 0.116 | **0.044** | 0.060 |

### A.9 ADDITIONAL TICA FREE ENERGY PROFILES

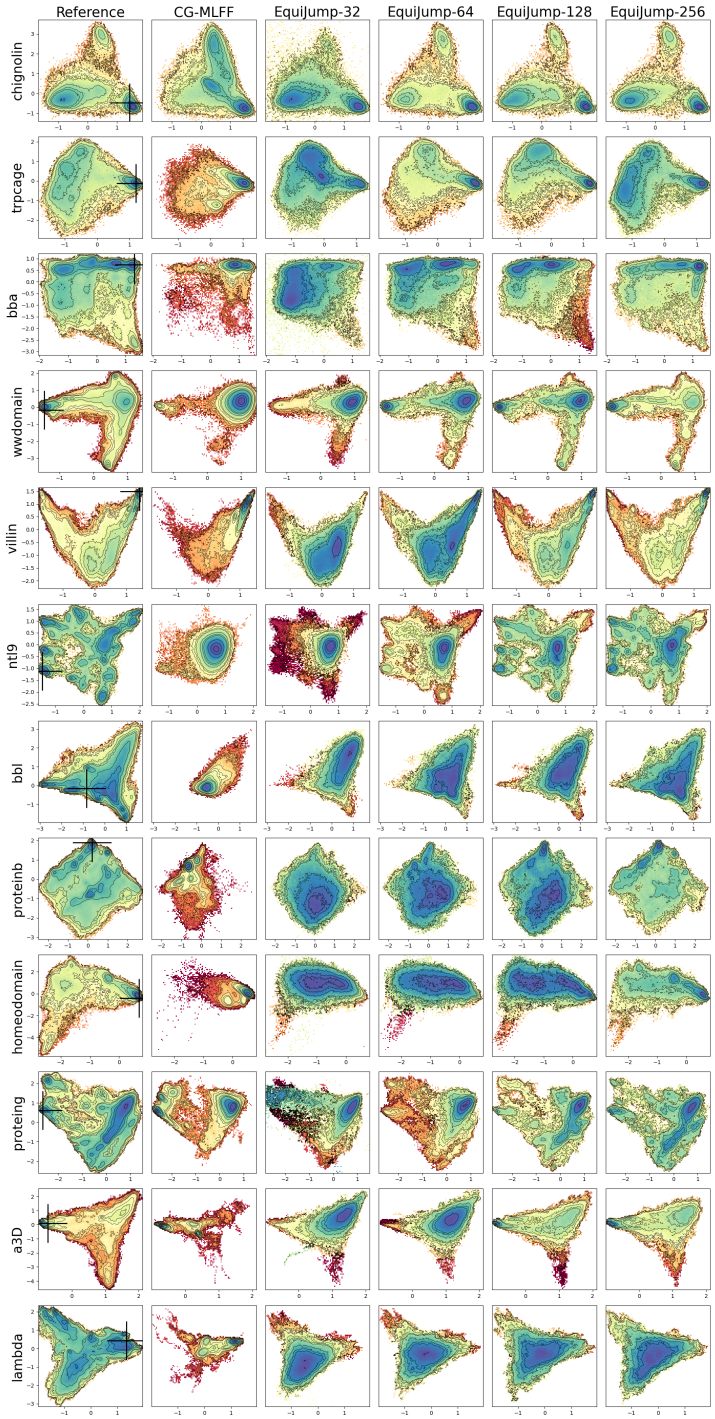


Figure 8: **From disorder to order: Free Energy profiles on TIC1 and TIC2 for comparison model and EquiJump models with increasing capacity.** While the MLFF model remains close to basin states, EquiJump is biased to less ordered regions despite staying in the manifold, and instead becomes more stable with increasing capacity.

### A.10 ADDITIONAL FREE ENERGY CURVES

In Figure 9, we show the estimated free energy of observable $\mathbf{C}_\alpha$ Root Mean Square Deviation (RMSD) from the reference crystal structure, following reweighting based on stationary distributions of fitted Markov Models. We compare the best performing EquiJump model against reference and CG-MLFF. These curves represent the energy distribution of $\mathbf{C}\alpha$-RMSD after the system reaches equilibrium and are highly sensitive to the accurate estimation of conformational transitions, making them a robust evaluation metric for dynamics models. We additionally provide free energy curve estimates for Radius of Gyration 10 and for Fraction of Native Contacts 11.

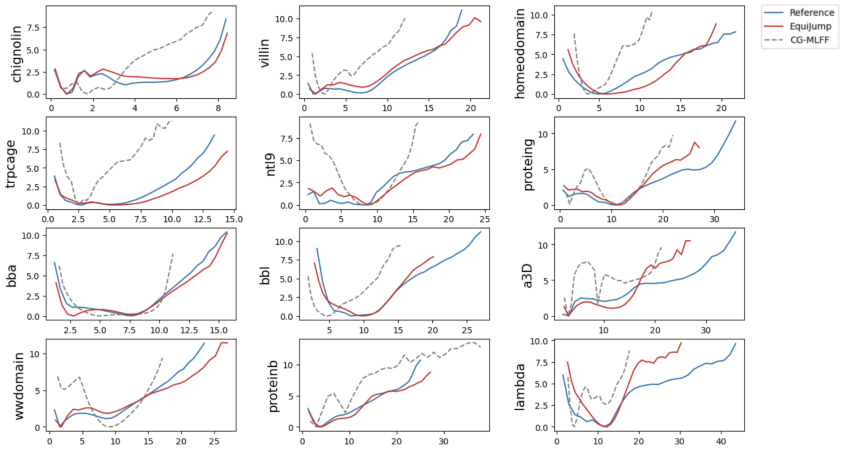


Figure 9: **Free Energy on $\mathbf{C}_\alpha$-RMSD for the 12 Fast-Folding Proteins**. We align trajectory samples to the reference crystal, and measure $\mathbf{C}_\alpha$-RMSDs (x-axis). Using Markov State Model (MSM) weights based on our TICA-based clusters, we reweight $\mathbf{C}_\alpha$-RMSD counts to obtain free energy estimates (y-axis). We find that EquiJump successfully approximates the free energy curves of reference trajectories.

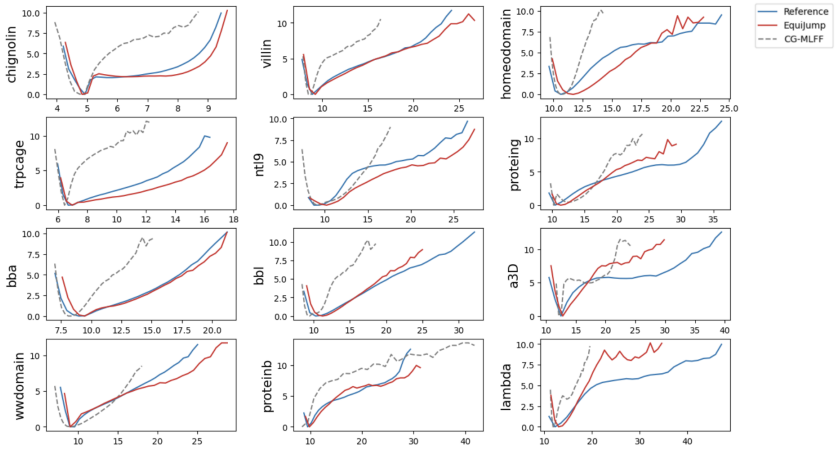


Figure 10: **Free Energy on $\mathbf{C}_\alpha$ Radius of Gyration for the 12 Fast-Folding Proteins**. We bin and reweigh counts of $\mathbf{C}_\alpha$ gyradii (x-axis) based on MSM weights to obtain free energy estimates (y-axis).

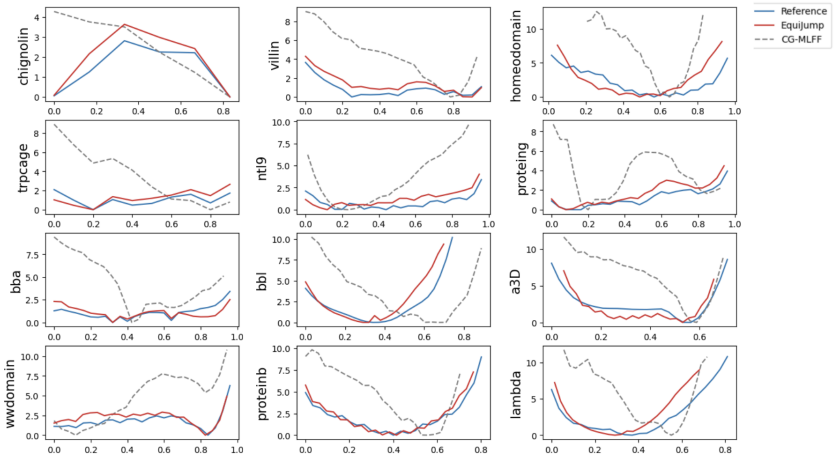


Figure 11: **Free Energy on Fraction of Native Contacts of $C_\alpha$ atoms for the 12 Fast-Folding Proteins**. We bin and reweigh Fraction of Native Contacts (FNC) (x-axis) of $C_\alpha$ atoms based on MSM weights to obtain free energy estimates (y-axis). We only consider residues at least 3 sequence positions apart, and use a cutoff of 8 Å for counting contacts.

### A.11 CHEMISTRY EVALUATION

We verify that EquiJump trajectories stay in a chemically valid manifold by plotting the distribution of bond lengths, bond angles and collisions of van der Waals radii against those in reference. In Figure 12, we plot those distributions for best peforming EquiJump model ($H = 256$) for fast-folding proteins Chignolin, Trp-cage, BBA and the WW domain. We observe that EquiJump accurately reproduces the distribution curves, interestingly revealing fewer counts for atomic clashes (despite larger dispersion) compared to reference. Our plots demonstrate that our model produces data that successfully stays within a chemically valid manifold across the different proteins.

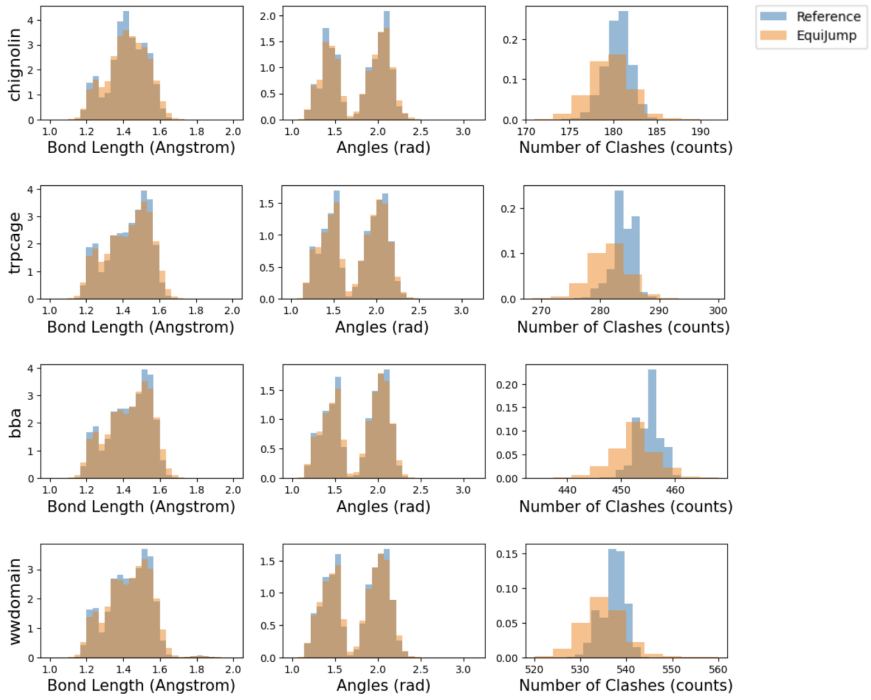


Figure 12: **Distribution of Chemical Measures**. We pick 3000 structures at random from reference trajectories and EquiJump. For each sample, we measure the distribution of bond lengths and bond angles considering all heavy atoms in the system. We estimate clashes by counting intersections of van der Waals radii for all pairs of non-bonded atoms.

### A.12 PERFORMANCE ANALYSIS

In order to study the performance of our model, we consider the largest protein (lambda) as reference. The classical MD simulation used to generate its trajectory uses explicit water and the total system has size around 12000 atoms (Lindorff-Larsen et al., 2011). Following Amber24 benchmarks, on the same hardware we use for our simulations (NVIDIA A100) a system twice as large (JAC) can reach a throughput of 1258 ns/day (Exxact Corp., 2024). Scaling linearly to the size of the lambda cell, this results in 3.6s for a single 100ps step. Drawing from this reference, in Table 5 we compare the performance of EquiJump models across different scales, where we observe positive acceleration factors for all model instances. Based on these metrics, we study the relation of EquiJump speedup against generation quality. In Figure 13, we identify the trade-off between estimated acceleration factors and accuracy in reproducing the distribution of TIC components for different simulation batch sizes. We observe that EquiJump models are able to accurately reconstruct TIC components (JS $< 0.1$) while accelerating by factors of 5-15$\times$ compared to Amber24, achieving a significant simulation throughput with minimal trade-off in precision. In comparison, CG-MLFF is estimated to be 1-2 orders of magnitude slower than the reference simulation (Majewski et al., 2023). Similarly, state-of-the-art neural force-field MACE-OFF (Kovács et al., 2023) is estimated to perform 2.5Msteps/day for its smallest model on the same hardware (Kovács et al., 2023). With a time step of 4 fs, this corresponds to 860s per 100ps step, or a $0.004\times$ slowdown in comparison to reference. In contrast, despite its already promising acceleration, the performance of EquiJump is likely to be further enhanced through additional network architecture optimization, exploration of more efficient differential equation solvers, and application of distillation and sampling acceleration techniques (Luhman and Luhman, 2021; Salimans and Ho, 2022). In Appendix A.13, we study the impact of sampling hyperparameters and find promising results for more acceleration through further tuning of noise scaling for fewer integration steps.

| | | Batch Size | | |
|---|---|---|---|---|
| Model (# Params) | | 1 | 8 | 32 |
| 32 (6.5M) | Time (s) | 0.34 | 1.49 | 3.35 |
| | Accel. ($\times$) | 10.60 | 19.33 | 34.39 |
| 64 (25.4M) | Time (s) | 0.51 | 2.26 | 6.07 |
| | Accel. ($\times$) | 7.05 | 12.74 | 18.98 |
| 128 (100.8M) | Time (s) | 0.60 | 3.12 | 12.17 |
| | Accel. ($\times$) | 6.00 | 9.23 | 9.47 |
| 256 (391.1M) | Time (s) | 1.05 | 6.40 | 26.09 |
| | Accel. ($\times$) | 3.40 | 4.50 | 4.42 |

Table 5: **Performance Metrics and Estimates**. We measure the time of transport for a step of 100 ps using different model capacities when integrating with 100 latent time steps for simulating the largest protein considered (lambda), and estimate the acceleration factor from representative classical MD with explicit solvent that generated the training dataset. All results are reported on NVIDIA A100 machines with single GPU of 80G.

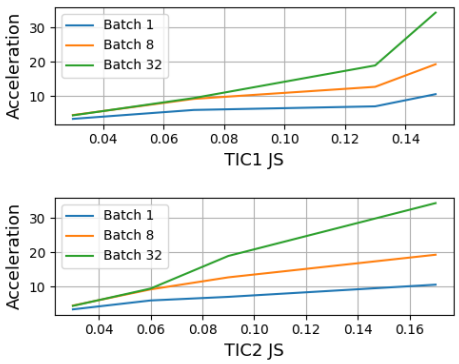


Figure 13: **Quality against Acceleration**. We plot estimated acceleration factors against Jensen-Shannon divergence (JS) for the distribution of long-term TIC components of EquiJump samples against reference trajectories. We display positive performance across batch sizes.

### A.13 SAMPLING PARAMETERS ABLATION

We study the impact of changing sampling hyperparameters $\epsilon(\tau)$ and $d\tau$ in the quality of long-term dynamics generation. For that, we train a model for simulating the dynamics of fast-folding protein Chignolin. We parameterize our model with $H = 32$ and train it for $120k$ steps with batch size 512. We consider scale of noising parameter $\epsilon(\tau) = \{0.1, 0.3, 0.5, 1.0, 2.0, 3.0, 10.0\}$ and number of steps in integration $(1/d\tau) = \{30, 50, 75, 100, 150, 200, 300\}$. For each, we sample 300 trajectories of 500 steps (50 ns). As above, we reweight metrics through MSM based on TICA-clusters to estimate values at equilibrium. In Figure 14, we show the Jensen-Shannon Divergence (JS) between ground truth and samples obtained at different parameterizations of $\epsilon$ and $d\tau$. In Figure 15, we show the effects of hyperparamter variation on the (long-term reweighted) density of the first TIC components of samples. We observe that large variation $(= 0.1, 10.0)$ on $\epsilon$ yields underperforming models. Notably, we observe that even with few steps $(30, 50, 75)$ higher quality can be obtained through smaller $\epsilon$ $(= 0.5, 0.75)$. Our results suggest that further investigation about the noising schedule $\epsilon(\tau)$ is a promising direction for increasing acceleration of high-quality simulation through fewer integration steps.

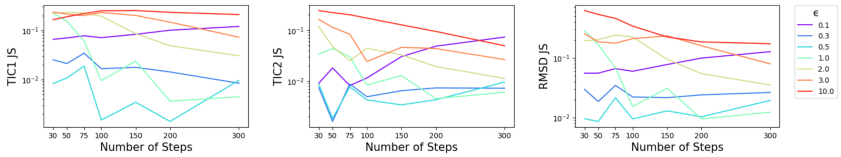


Figure 14: **Effect of Sampling Parameters on Generation Quality**. We measure Jensen-Shannon divergence (JS) from reference of (reweighted) observable distributions (TIC1, TIC2 and RMSD) across different sampling parameters $\epsilon(\tau)$ and $(1/d\tau)$.

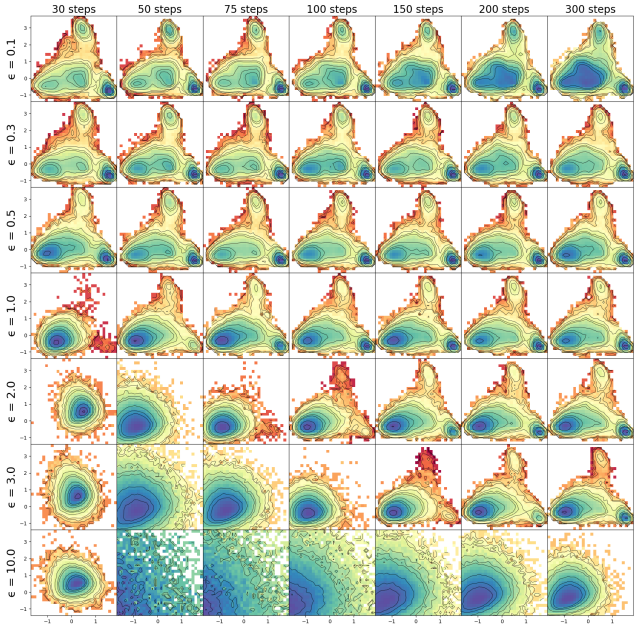


Figure 15: **Effect of Sampling Parameters on TIC profile**: we plot the density of the first TIC components of Chignolin with varying noise factor $\epsilon(\tau)$ and integration number of steps $(1/d\tau)$.