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

Anonymous authors
Paper under double-blind review

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 on 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., 2024) 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 to this, 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 utilize 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 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.

Our main contributions are as follows:

- We extend the Two-Sided Stochastic Interpolants framework to simulate the dynamics of three-dimensional representations. By training on trajectory data, our approach directly leverages the close relationship between consecutive timesteps.

- We introduce a novel four-track SO(3)-equivariant neural architecture to implement a generative transport operator, which we apply to model the dynamics of *all heavy atoms* in protein monomers.
- We learn *transferable* simulators for capturing the dynamics of 12 fast-folding proteins while performing large jumps in time, demonstrating the effectiveness of EquiJump in reproducing long-term dynamics and ensemble distributions.

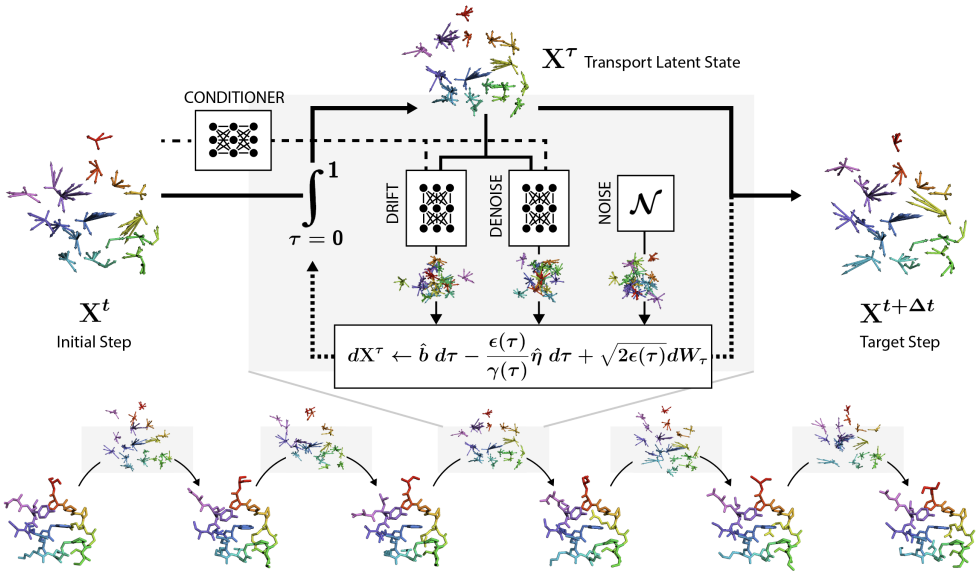


Figure 1: **Direct bridging of 3D all-atom simulation time steps**: EquiJump runs an stochastic interpolants-based transport process on coordinates and 3D geometric representations to generate future time frames from an initial state. Gray boxes depict transport across the learned latent space, which takes Gaussian noise perturbations and uses noise ($\hat{\boldsymbol{\eta}}$) and drift ($\hat{\boldsymbol{b}}$) predictions to directly transform all-atom proteins across time and 3D space.

2 Related Work

Recent advancements in protein modeling through deep learning have led to the development of several models capable of replicating molecular dynamics (MD) trajectories. (Wang et al., 2019; Husic et al., 2020) introduced supervised models trained through direct force matching, demonstrating their ability to transfer simulations across different proteins. (Majewski et al., 2023) implements a unified transferable model for multiple proteins. Their approach relies on force matching, which approximates atomistic forces. In contrast, EquiJump eliminates the need for force data by using a two-sided stochastic transport framework that directly generates next steps of configurations, enabling time stepping that is orders of magnitude larger than those required by this method. We show that our model outperforms force-matching approaches by maintaining kinetic consistency and replicating dynamical observables more precisely.

(Fu et al., 2023) learns to predict accelerated, coarse grained dynamics of polymers with GNNs. (Köhler et al., 2023) utilizes Normalizing Flows (Gabrié et al., 2022) for coarse grained force-matching, while (Arts et al., 2023) builds upon Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) through Graph Transformers (Shi et al., 2020; Costa, 2021). These approaches focus heavily on coarse-grained representations, which limit their ability to simulate the full complexity of protein dynamics at the all-atom level. In contrast, EquiJump operates directly at the all-atom scale, achieving efficiency through SO(3)-equivariant neural networks and residue representation (Costa et al., 2024). This allows EquiJump to provide a more accurate and comprehensive simulation of protein dynamics, combining the precision of all-atom modeling with the computational efficiency typically seen in coarse-grained approaches.

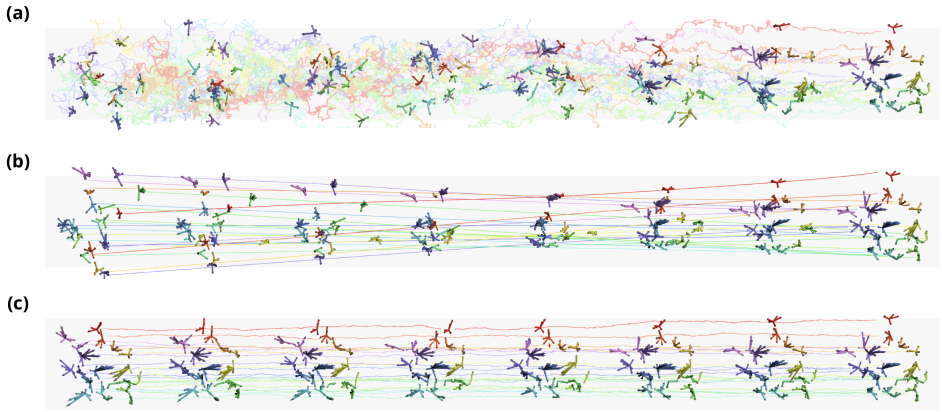


Figure 2: **Neural Transport of Tensor Clouds**. **(a) DDPM** defines an SDE for denoising samples from a Gaussian prior, while standard **(b) Flow Matching** traces a velocity field-based ODE for moving the Gaussian samples. **(c) Two-Sided Stochastic Interpolants** instead enable transporting through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

Recent models have proposed generating samples from a prior distribution while conditioning on an initial configuration. Timewarp (Klein et al., 2023) enhances MCMC sampling with conditional normalizing flows, while ITO (Schreiner et al., 2023) uses a PaiNN-based network (Schütt et al., 2021) to learn a conditional diffusion model for next-step prediction. Similarly, F^{3}low (Li et al., 2024) employs FramePred (Yim et al., 2023) and Optimal Transport Guided Flow Matching (Zheng et al., 2023). Finally, (Jing et al., 2024) applies one-sided stochastic interpolants (Ma et al., 2024) to interpolate or extrapolate molecular configurations. These approaches rely on transforming Gaussian priors via stochastic (SDE) or ordinary differential equations (ODE), where the prior often lies far from the true data distribution. Instead, EquiJump uses two-sided stochastic interpolants to directly bridge trajectory snapshots, leveraging the configuration proximity of consecutive timesteps and enabling a transport that stays close to physical states (Figure 2; Appendix A.1).

3 Methods

3.1 Stochastic Interpolants

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). 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 α 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 ρ_0 and ρ_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 γ 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:

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

We can similarly define the noise term $\eta(\tau, \mathbf{X})$ as:

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

In practice, the exact forms of b and η 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:

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

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

We can then sample $\mathbf{X}_{\tau=1} \sim p(\tau = 1, \mathbf{X}_1)$ through an ordinary differential equation (ODE), or a stochastic differential equation (SDE):

$$d\mathbf{X}_\tau = \hat{b}(\tau, \mathbf{X}_\tau) d\tau \tag{8}$$

$$d\mathbf{X}_\tau = \left(\hat{b}(\tau, \mathbf{X}_\tau) - \frac{\epsilon(\tau)}{\gamma(\tau)}\hat{\eta}(\tau, \mathbf{X}_\tau)\right) d\tau + \sqrt{2\epsilon(\tau)} dW_\tau \tag{9}$$

where W_τ is the Weiner process. Once we have learned 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 $(\tau = 1, \mathbf{X}_1 \sim p_1)$. Furthermore, following from eqs. (8) and (9) the probability $p(\tau, \mathbf{X}_\tau)$ is SO(3)-equivariant when $\hat{b}$ and $\hat{\eta}$ are SO(3)-equivariant and dW_τ is isotropic. We provide more details on interpolant parameterization in Appendix A.2.

3.2 Two-Sided Stochastic Interpolants for Dynamics Simulation

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 $\rho_1 = \rho(\mathbf{X}^{t+1} \mid \mathbf{X}^t)$. 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 $\mathbf{X}^t$. We apply this approach to simulating all-atom protein dynamics, as depicted in Figure 1. In this context, $\mathbf{X}^t$ represents a 3D all-atom protein conformation at time t, which is provided as input to our model. We frame it as the source distribution, and set $\mathbf{X}_{\tau=0} = \mathbf{X}^t$. We then employ an iterative process governed by the integration of eqs. (8) and (9) from $\tau = 0$ to $\tau = 1$. This produces a sample $\mathbf{X}_{\tau=1}$, which follows the distribution $\mathbf{X}_{\tau=1} \sim \rho_1$, generating a next step in the simulation $\mathbf{X}^{t+1}$.

3.3 Multimodal Interpolants of Geometric Representations

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 eqs. (8) and (9) to the multi-modal type $\mathbf{X}_i = (\mathbf{V}_i, \mathbf{P}_i)$ by integrating geometric features and coordinate components 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 define the Tensor Cloud dot product as $\mathbf{X}_i \cdot \mathbf{X}_j = \mathbf{V}_i \cdot \mathbf{V}_j + \mathbf{P}_i \cdot \mathbf{P}_j$. In general, treating the feature and coordinate components independently allows for different parameterizations of the interpolant. In this work, we use the same interpolant form for both components, and only adjusting the scaling $\gamma(t)$ of the variable $\mathbf{Z}$. For further details see Appendix A.2.

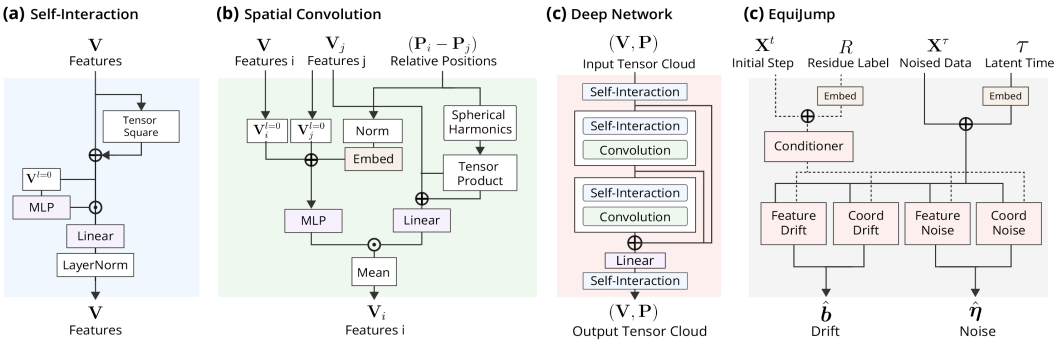


Figure 3: **EquiJump Architecture**: **(a)** The **Self-Interaction** Layer updates geometric features independently, mixing $\mathbf{V}^l$ of different degrees into new features through a Tensor Square operation. **(b)** The **Spatial Convolution** layer updates representations by aggregating the tensor 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 two-sided stochastic interpolant process.

3.3.1 Protein Structure Representation

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_α 3D coordinate $\mathbf{P}_i^\alpha \in \mathbb{R}^3$, and a geometric feature of order $l = 1$ with multiplicity 13, $\mathbf{V}_i^A \in \mathbb{R}^{13\times 3}$. This feature encodes the relative 3D vector from the C_α to all other heavy atoms in the residue, following a canonical ordering. For residues with fewer than 13 non-C_α heavy atoms, we pad the atom vectors. This modeling approach, based on (King and Koes, 2020; Costa et al., 2024), allows for the direct representation of all heavy atoms in 3D, while maintaining a coarse-grained representation anchored on the C_α.

3.3.2 Neural Network Architecture and Training

To efficiently process 3D data, we utilize Euclidean-equivariant neural networks (Geiger and Smidt, 2022; Miller et al., 2020). We design a neural network 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, conditioned on the sequence $\mathbf{R}$, the source structure $\mathbf{X}^t$, the latent transport structure $\mathbf{X}_\tau^t$, and the latent time τ (Figure 3).

The EquiJump layer is built from two SO(3)-equivariant modules: the Self-Interaction (Figure 3.a) module for updating features $\mathbf{V}^l$ indepedently from coordinates, based on (Costa et al., 2024; Batatia et al., 2022); and the Spatial Convolution (Figure 3.b) module for sharing information between neighbors, based on Tensor Field Networks (Thomas et al., 2018). At each layer, we also employ residual connections and the SO(3)-equivariant layer norm from (Liao and Smidt, 2022). We build a deep neural network (DNN) by stacking L times the above blocks (Figure 3.c). Refer to Appendix A.2 for additional details.

We use 5 DNNs in our model (Figure 3.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. 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 τ, and only the prediction headers are used in the integration loop of the latent transport. We train and sample these networks 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 $\hat{b}_\mathbf{V}$, $\hat{b}_\mathbf{P}$, $\hat{\eta}_\mathbf{V}$, $\hat{\eta}_\mathbf{P}$, $\mathbf{f}_{\text{cond}}$
$t \sim \mathcal{U}(1, T-1)$
$\tau \sim \mathcal{U}(0, 1)$
$\mathbf{Z}^\tau \sim \mathcal{N}(0, \mathbb{I})$
$\tilde{\mathbf{X}}^t = \mathbf{f}_{\text{cond}}(\mathbf{R}, \mathbf{X}^t)$
$\mathbf{X}_\tau^t \leftarrow (1-\tau)\cdot \mathbf{X}^t + \tau \cdot \mathbf{X}^{t-1} + \gamma(\tau)\mathbf{Z}^\tau$
$\hat{\eta} \leftarrow \big(\hat{\eta}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{\eta}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
$\hat{b} \leftarrow \big(\hat{b}_\mathbf{V}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau), \hat{b}_\mathbf{P}(\tilde{\mathbf{X}}^t, \mathbf{X}_\tau^t, \tau)\big)$
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 $d\tau$
$\mathbf{X}_{\tau=0}^t = \mathbf{X}^t$
$\tilde{\mathbf{X}}^t = \mathbf{f}_{\text{cond}}(\mathbf{R}, \mathbf{X}^t)$
for $(\tau = 0\,;\, \tau < 1\,;\, \tau = \tau + d\tau)$ **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 $\mathbf{X}_{\tau=1}^t$

4 Experiments and Results

4.1 Fast-Folding Proteins

To evaluate the capability of our model to reproduce protein dynamics, we leverage the 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 100 ps spaced snapshots of MD for 12 proteins ranging from 10 to 80 residues. The trajectories are made up of several NVT runs (20 to 100 ns) at T=350 K from different starting configurations sampling the phase space, for an aggregated simulation time of hundreds to thousands of μs per protein. We refer to the original work and Appendix A.3 for more details on the dataset.

While training on the whole dataset is effective in producing configurations within the original phase space, it is not optimal to correctly reconstruct the potential energy surface. This is because when performing MD the relative probability of different configurations is extremely sensitive to the correct description of the dynamics of the transition states. However, for the slowest modes of the system these states are by their own nature very high in free energy and therefore heavily underrepresented in our training dataset. For this reason, 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 that will enable our model to better learn the complex dynamics of transition states. To achieve this, we define collective variables and add a bias to the sampling probability in training, to compensate the free energy of the system. We first find the relevant degrees of freedom through TICA analysis (Pérez-Hernández et al., 2013) (see below) and then construct a small number of clusters through k-means in this simplified space. Since we are interested in the long-time dynamics of the system, this provides a reasonable basis to consider for reweighing the trajectories. If our data came from a long NVT dynamics, the free energy F_i of cluster i would be proportional to the number of samples: $N_i \propto e^{-\beta F_i}$ (β being the inverse temperature). In the discrete basis, the reweighing with $e^{\beta F_i}$ would thus naturally correspond to a uniform distribution over the clusters, as desired. Since our dataset was obtained through a biased procedure to enhance variety (Majewski et al., 2023), the relation between probability and free energy does not hold true before reweighing. However, we still propose to sample uniformly on the clusters, as this is a simple procedure that does not depend on the population of each clusters, if not for its small effect on the definition of the clusters themselves. We show the cluster centers and distributions of population sizes for each protein in Figure 8. Refer to Appendix A.4 for more details.

4.2 Model Training and Sampling

We train EquiJump models for time-evolving any of the 12 fast-folding proteins following Algorithm 1. We use the Adam optimizer (Kingma and Ba, 2017) with linearly decreasing learning rate from

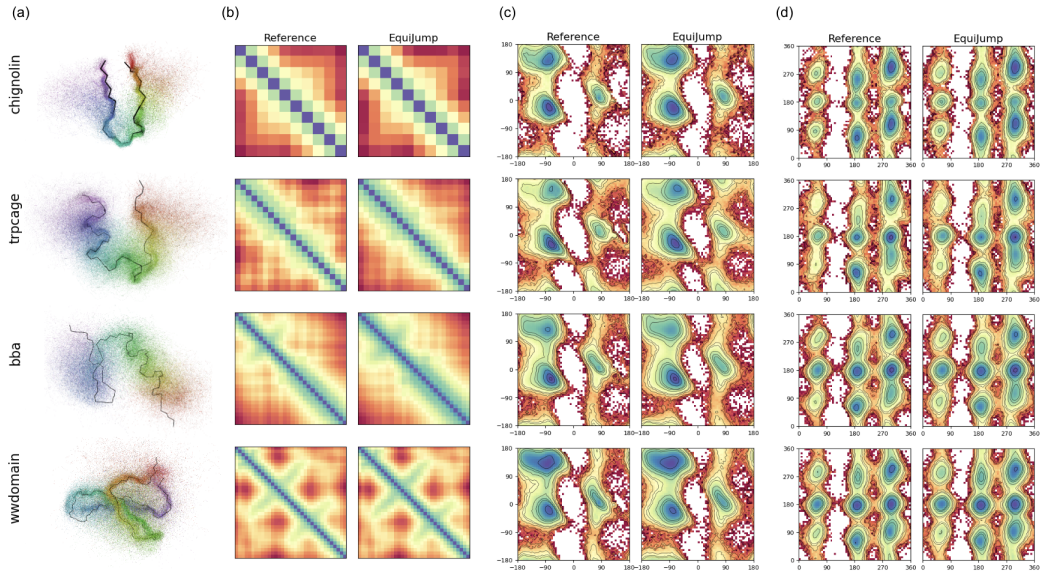


Figure 4: **EquiJump Samples**: **(a)** We visualize the distribution in 3D of 1500 backbone random samples of EquiJump trajectories. We align samples to the crystal backbone (shown in black) and verify that our model stays close to the native state basin. We show **(b)** mean pairwise $\mathbf{C}_\alpha$ distance matrices, **(c)** Ramachandran plots of backbone dihedrals and **(d)** Janin plots of sidechain dihedrals of EquiJump samples against reference trajectory data.

1×10^{-2} to 1×10^{-3} over 150k steps, with batch size of 128. We train all models for 500k steps. We perform all our experiments on NVIDIA A100 machines with 2-4 GPUs. Additional details on model paramterization can be found in Appendix A.2.

For sampling, we start from a configuration of the enhanced dataset and iteratively apply the model. We perform 500 simulations of 500 steps. We employ 100 steps of integration to obtain the next configuration. While each evaluation is slower than the original model used for MD, our timesteps are effectively 25000 times longer. Moreover, the simulations can be very efficiently parallelized on modern hardware.

4.3 Configuration Space and Free Energy Surface Analysis

In Figure 4, we present samples of the best performing EquiJump model and compare its generated densities of backbone and sidechain dihedral angles and pairwise $\mathbf{C}_\alpha$ distances to those of reference trajectory data. We provide plots for the additional fast-folding proteins in Appendix A.5. By accurately recovering these metrics across the sampled conformations, we demonstrate that EquiJump remains within the physical configuration space of the original dataset and effectively avoids sampling nonphysical states.

In order to study the long-term dynamical behavior of the model, we leverage Time-lagged Independent Component Analysis (TICA) (Pérez-Hernández et al., 2013), which offers a reduced dimensional space that highlights the slow macroscopic modes of the system. We obtain reference TICA components from the original trajectories by considering a similarity based on the Euclidean distance between C_α and a lagtime of 2ns. To estimate long-term probabilities after equilibration, we reweigh the the density of sampled configurations. We first cluster the configuration using K-means in the first 4 TIC dimensions with 100 clusters. We then build a Markov State Model (MSM) on the basis 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 we use to reweigh them and approximate the Boltzmann distribution. Note that this reweighing is extremely sensitive to the correct description of the transition states, as a wrong sampling of the transition probability will exponentially affect the relative probability of different basins. As such, the correct description of these probability distributions is a good indicator of the faithfulness of the

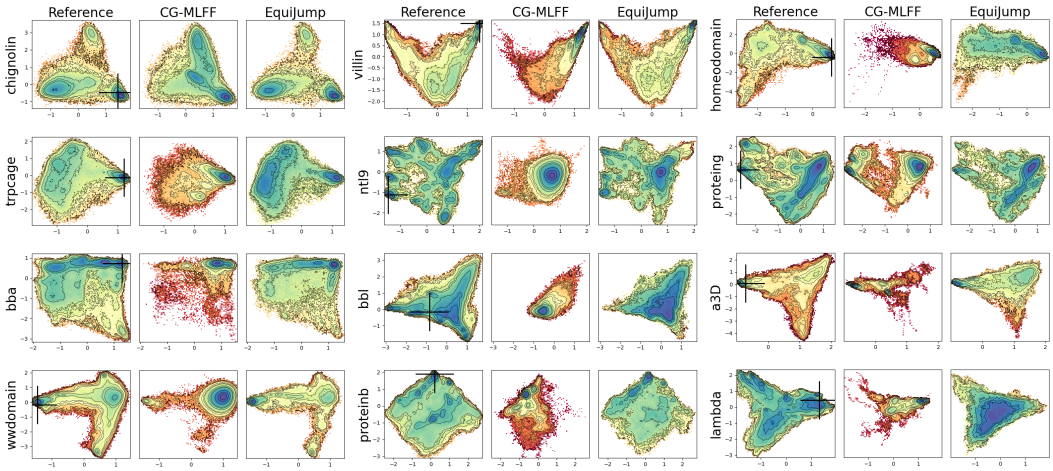


Figure 5: **Free Energy on TICA components for the 12 Fast-Folding Proteins**. We compare the free energy of EquiJump against that of the reference and that of available model CG-MLFF. The free energy for each plot is set to 0 at the minimum and the color map is in units of $k_{\mathrm{B}}T$ at the MD temperature of 350 K. EquiJump succesfully recovers the dynamics of the proteins, covering the phase space and stabilizing the basin of most (shown as + in reference profile).

long-term dynamics. The density plot in the first two TIC dimensions for these reweighed samples is shown in Figure 5. In these plots, different regions represent different conformational states and their proximity indicates time autocorrelation. The logarithmic color scale represents the free energy in units of k_BT, and it is shifted to the minimum of each plot. We find that EquiJump sucessfully captures the dynamics of the 12 fast-folding proteins by accurately recovering the free energy curve in that describes long-term behavior. While performing significant steps of 100ps, our model is able to cover the phase space and accurate reproduce the dynamical profiles of observables.

4.4 Model Comparison and Ablation

In order to assess the quality of EquiJump trajectories, we compare its performance across model capacities $H = \{32, 64, 128, 256\}$, where H is the per residue dimension of the latent representation, and against available transferable model CG-MLFF (Majewski et al., 2023), which is based on coarse-graining and force matching, and uses Langevin sampling for dynamical generation. To the best of our knowledge, this is the only other multi-protein model that covers the 12 fast-folding proteins.

We assess the dynamics of EquiJump and compare it to CG-MLFF by estimating their free energies on TICA space as shown in Figures 5. We additionally provide free energy profiles across different EquiJump model capacities in Appendix A.6. We note that while CG-MLFF remains stable within most native basins, EquiJump covers more of the phases space and reveals stronger bias to less likely and disordered states, which are attenuated through increased model capacity (Appendix A.6). Nevertheless, our model reveals better reconstruction of the slow components and more accurate profiling of the free energy TICA maps across the considered proteins.

In Figure 6, 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 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. In Appendix A.7 we provide free energy curves for additional ensemble observables. Through these plots, we observe that EquiJump precisely replicates the energy curves in comparison to reference, correctly modeling basin states while providing a more extensive profile in comparison to CG-MLFF.

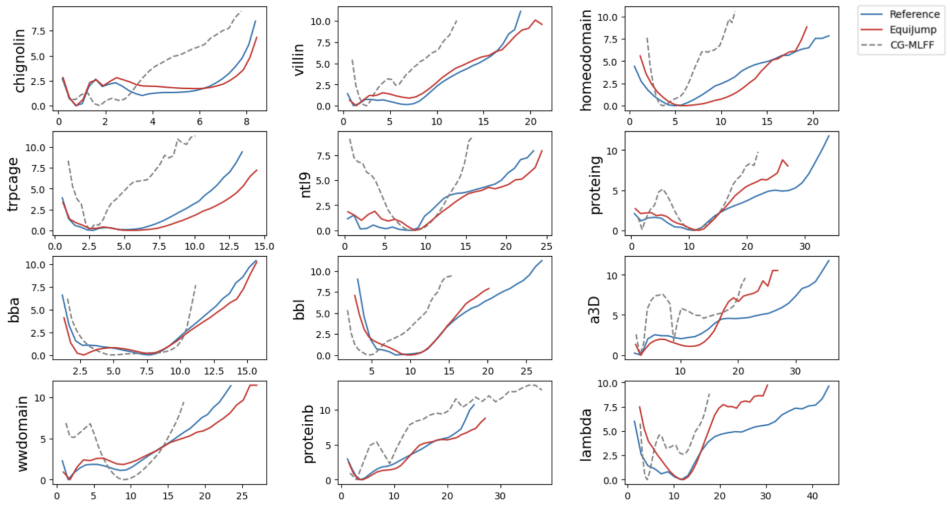


Figure 6: **Free Energy on C_α-RMSD for the 12 Fast-Folding Proteins**. We align trajectory samples to the reference crystal, and measure C_α-RMSDs (x-axis). Using Markov State Model (MSM) weights based on our TICA-based clusters, we reweight C_α-RMSD counts to obtain free energy estimates (y-axis). We find that EquiJump successfully approximates the free energy curves of reference trajectories.

		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 1: **Jensen-Shannon Divergence** of ensemble observables of 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 2: **Percent Error in Predicting Ensemble Averages** of fast-folding proteins observables

In Tables 1 and 2, we investigate model performance in reproducing the long-time (MSM-reweighted) distribution of ensemble observables and in recovering the average values of these observables. In both tables, metrics are averaged over the twelve proteins. We additionally provide protein-specific results for Jensen-Shannon divergence comparisons in Appendix A.8. Our tables 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 accurately reproduce long-term equilibrium behavior, it successfully navigates this manifold across model sizes (Appendix A.6), while achieving overall superior performance.

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 3 we

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

Table 3: **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 GPUs of 80G.

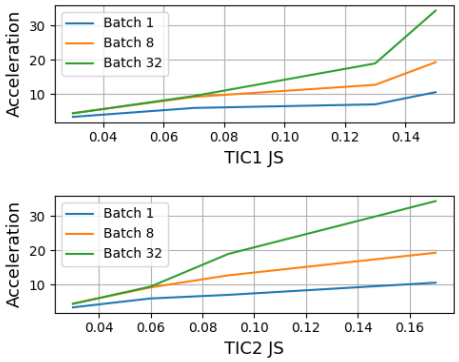


Figure 7: **Quality against Acceleration**. We plot estimated acceleration factors against Jensen-Shannon divergence (JS) for the distribution of long-term TIC components of EquiJump samples against reference trajectories. We display positive performance across batch sizes.

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 7, 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, while MACE-OFF (Kovács et al., 2023) represents the state-of-the-art in machine learned force-fields, it 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 would amount to 860s per 100ps step, representing 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 hyperparameter optimization, exploration of more efficient differential equation solvers, and application of distillation and sampling acceleration techniques (Luhman and Luhman, 2021; Salimans and Ho, 2022).

5 Conclusion

In this work, we introduced EquiJump for learning the dynamics of 3D protein simulations. EquiJump extends Two-Sided Stochastic Interpolants for 3D dynamics through SO(3)-equivariant neural networks, and is implemented through a novel four-track architecture that handles all-atom structures. We validated our approach through a series of experiments on large-scale dataset of fast-folding proteins, where we demonstrated a unified model that accurately reproduces complex dynamics across different proteins. We ablated model capacities and compared our model to existing approaches, outperforming state-of-the-art methods in terms of accuracy and efficiency. Our results suggest EquiJump provides a stepping stone for future research in modeling and accelerating protein dynamics simulation. Future work will focus on architecture and parameter efficiency, on transferability across data, and on generalization.

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.

Costa, A. d. S. (2021). Chaperonet : distillation of language model semantics to folded three-dimensional protein structures.

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*.

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*.

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., Stark, H., Jaakkola, T., and Berger, B. (2024). 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.

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

Liao, Y.-L. and Smidt, T. (2022). Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *arXiv preprint arXiv:2206.11990*.

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.

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.

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.

Schütt, K. T., Unke, O. T., and Gastegger, M. (2021). Equivariant message passing for the prediction of tensorial properties and molecular spectra.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2020). Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

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.

Yim, J., Trippe, B. L., Bortoli, V. D., Mathieu, E., Doucet, A., Barzilay, R., and Jaakkola, T. (2023). Se(3) diffusion model with application to protein backbone generation.

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:

$$\mathbf{M}\ddot{\mathbf{X}}(t) = -\nabla E(\mathbf{X}(t)) - \mathbf{R}\dot{\mathbf{X}}(t) + \mathbf{W}(t), \tag{10}$$

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:

$$\langle \mathbf{W}(t) \rangle = 0, \quad \langle \mathbf{W}(t)\mathbf{W}(t')^T \rangle = 2k_B T \mathbf{R}\delta(t - t'), \tag{11}$$

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. 10 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. 10 simplifies to the Brownian dynamics form:

$$\dot{\mathbf{X}}(t) = -\mathbf{R}^{-1}\nabla E(\mathbf{X}(t)) + \mathbf{R}^{-1}\mathbf{W}(t). \tag{12}$$

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. (12) 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. (9), while also following the form of eq. (12), 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 ARCHITECTURE DETAILS

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.

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
1: $(\tilde{\mathbf{P}}, \tilde{\mathbf{V}})_{1:k} \leftarrow k\text{NN}(\mathbf{P}_i, \mathbf{P}_{1:N})$
2: $R_{1:k} \leftarrow \text{Embed}(||\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i||_2)$,
3: $\phi_{1:k} \leftarrow \text{SphericalHarmonics}(\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i)$
4: $\tilde{\mathbf{V}}_{1:k} \leftarrow \text{MLP}\big(R_k \oplus \mathbf{V}^{l=0}_{1:k} \oplus \mathbf{V}^{l=0}\big) \cdot \text{Linear}\big(\tilde{\mathbf{V}}_{1:k} \otimes \phi_{1:k}\big)$
5: $\mathbf{V} \leftarrow \text{Linear}\left(\mathbf{V} + \frac{1}{k}\big(\sum_k \tilde{\mathbf{V}}_k\big)\right)$
6: **return** $(\mathbf{V}, \mathbf{P})$

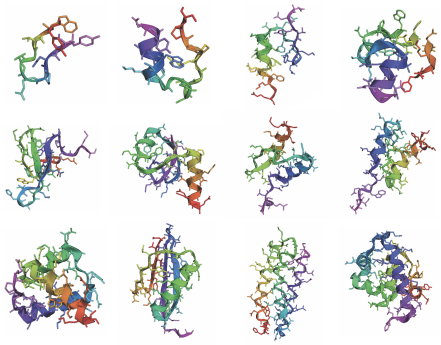
Algorithm 5 EquiJump Deep Network

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

A.3 Dataset Details

We adapt the dataset produced by (Majewski et al., 2023). The dataset consists of trajectories of 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
λ-repressor	80	15697



A.4 Details on Enhanced Sampling Clusters

To choose clusters that are diverse and dynamically relevant for enhanced sampling in training, we perform K-means clustering on 2D TICA components and find 200 clusters for each protein. Figure 8 visualizes cluster centers and the distribution of population sizes. Our enhanced dataset first samples a cluster, then from the cluster a configuration and its transition.

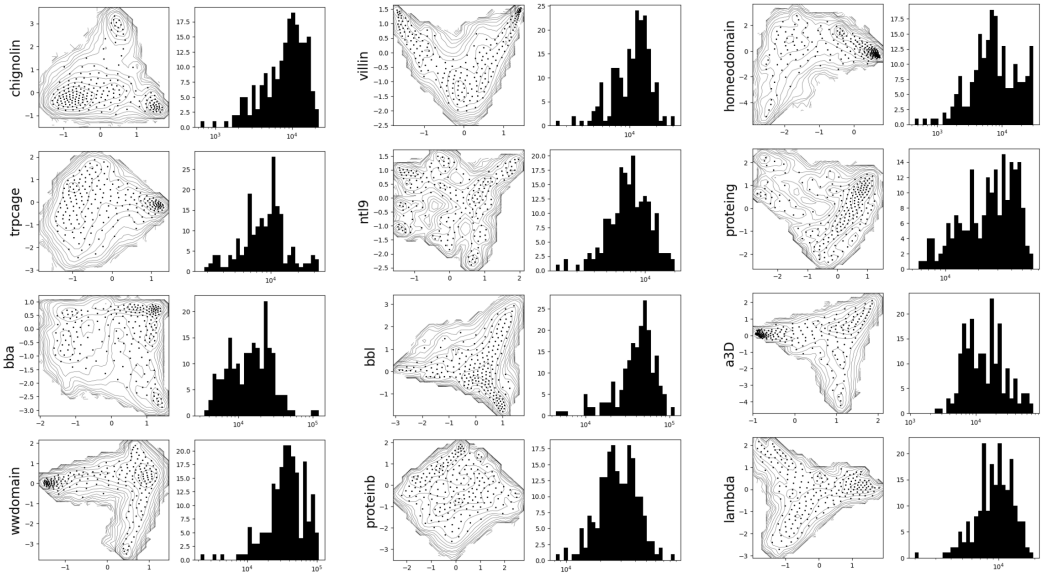


Figure 8: **Cluster Centers and Distribution of Cluster Population Sizes**.

A.5 Additional Visualization of Samples

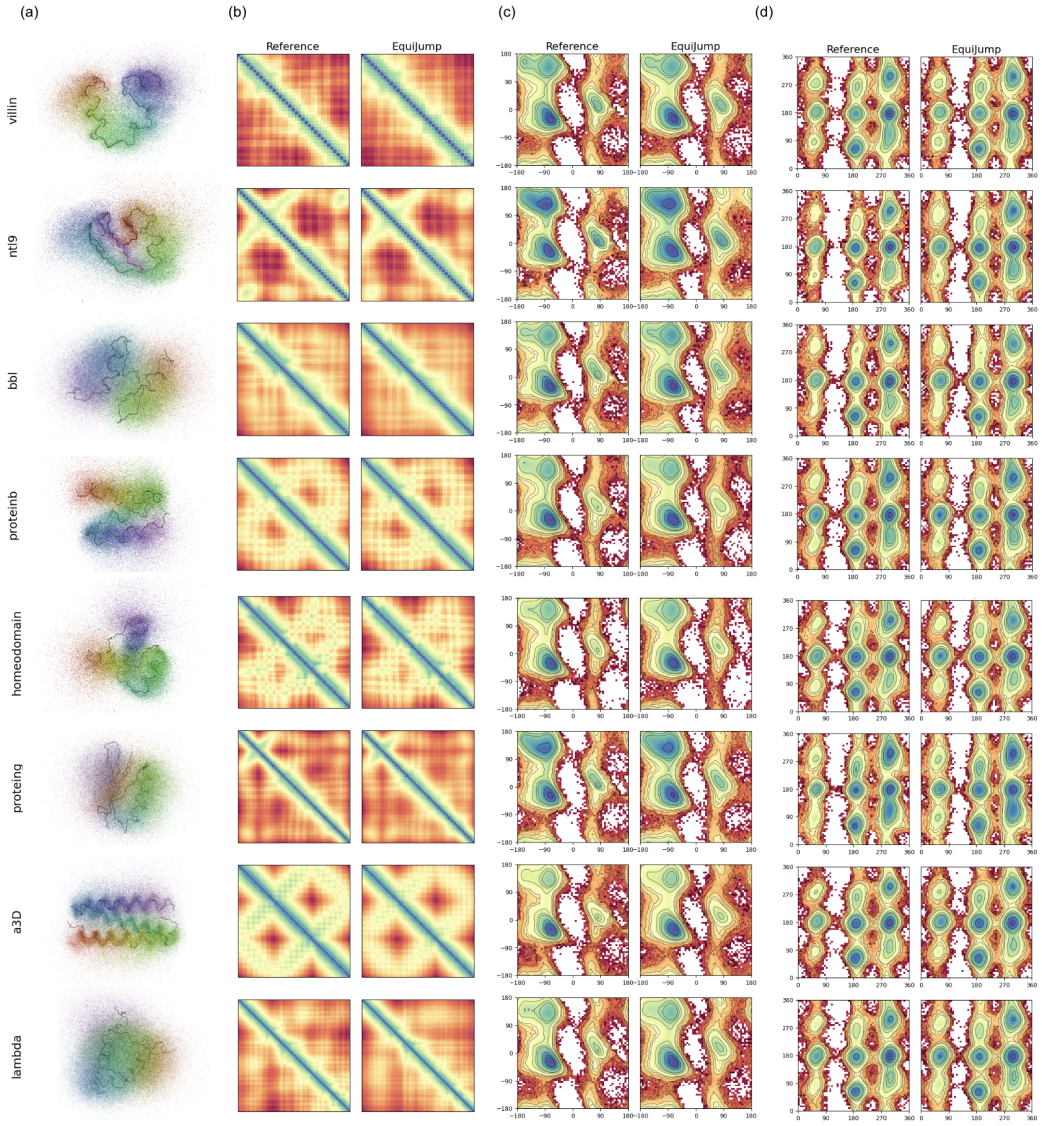


Figure 9: **EquiJump Samples**: **(a)** We visualize the performance of EquiJump on additional fast-folding proteins by superposing 1500 backbone random samples of EquiJump trajectories. We align samples to the crystal backbone (shown in black). We further present **(b)** mean pairwise $\mathbf{C}_\alpha$ distance matrices, **(c)** Ramachandran plots of backbone dihedrals and **(d)** Janin plots of sidechain dihedrals of EquiJump samples against reference trajectory data.

A.6 Additional TICA Free Energy Profiles

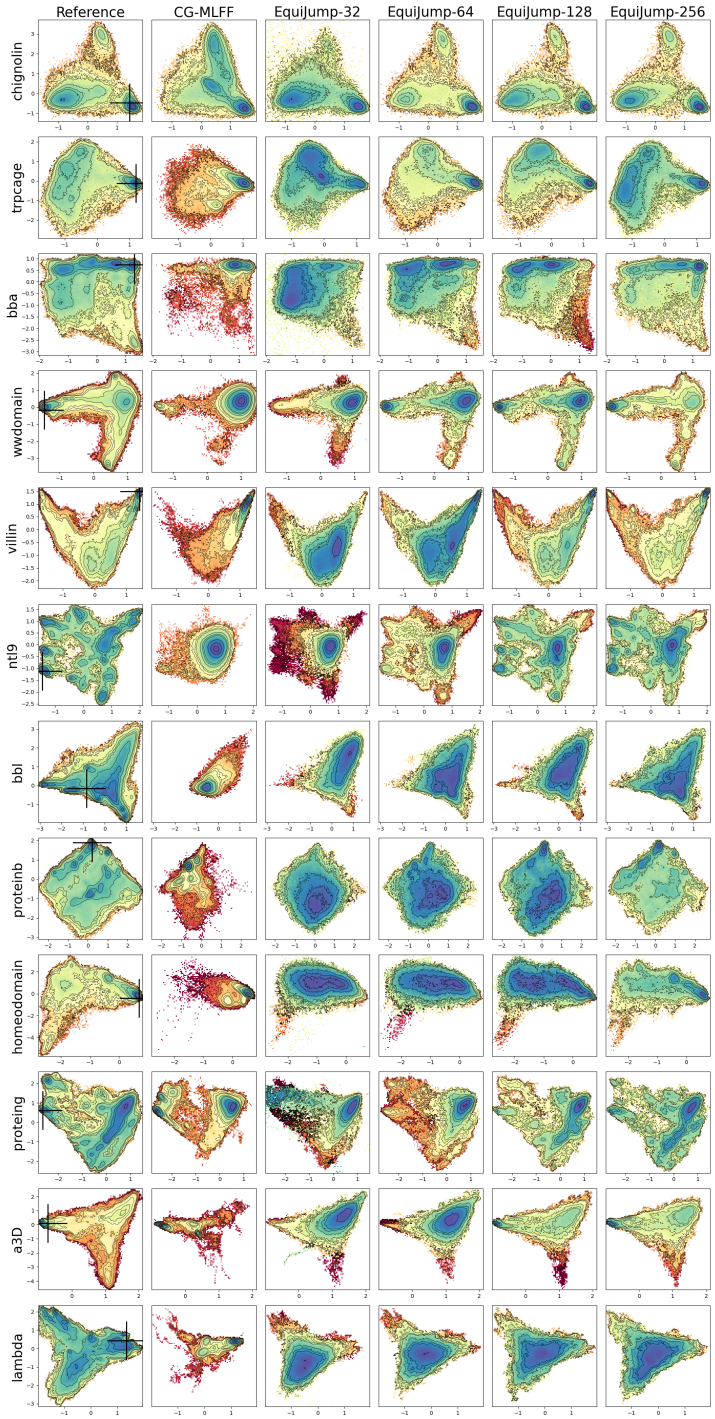


Figure 10: **From disorder to order: Free Energy profiles on TIC1 and TIC2 for comparison model and EquiJump models with increasing capacity.** While the MLFF model remains close to basin states, EquiJump is biased to less ordered regions despite staying in the manifold, and instead becomes more stable with increasing capacity.

A.7 Additional Free Energy Curves

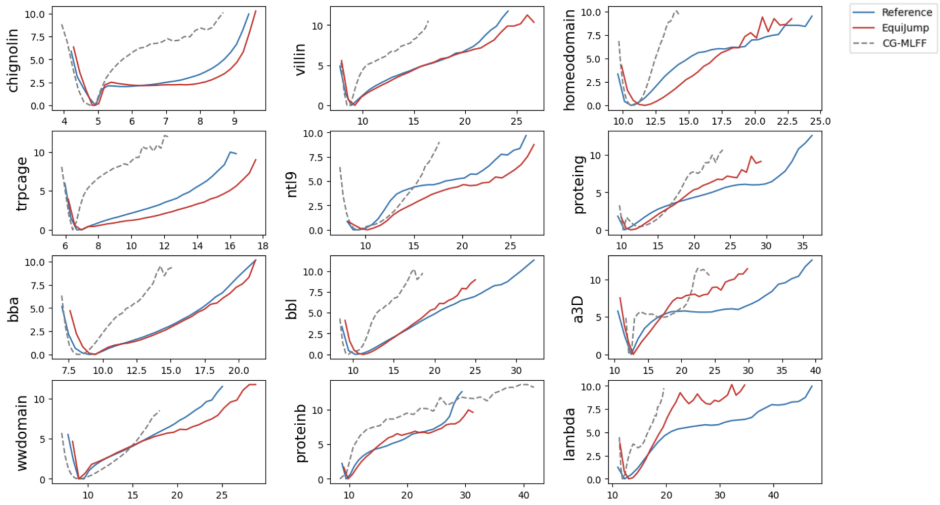


Figure 11: **Free Energy on $\mathbf{C}_\alpha$ Radius of Gyration for the 12 Fast-Folding Proteins**. We bin and reweigh counts of $\mathbf{C}_\alpha$ gyradii (x-axis) based on MSM weights to obtain free energy estimates (y-axis).

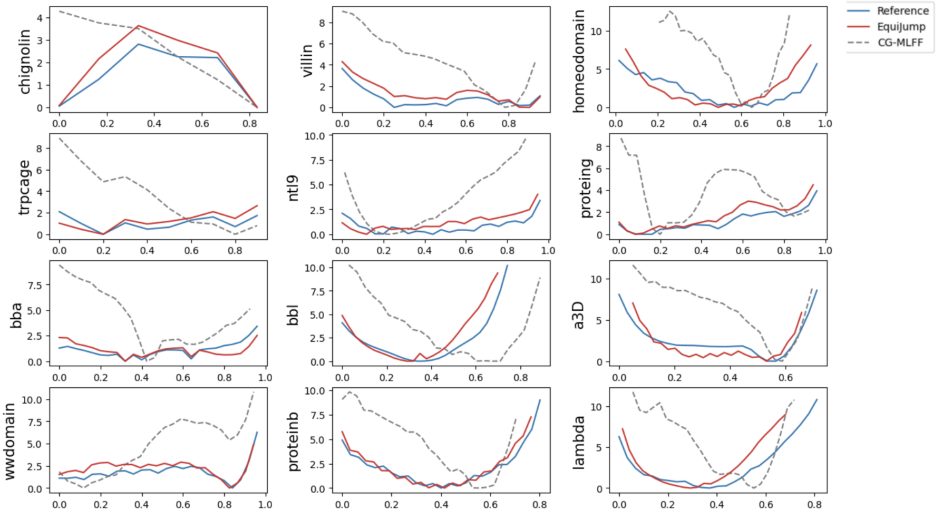


Figure 12: **Free Energy on Fraction of Native Contacts of $\mathbf{C}_\alpha$ atoms for the 12 Fast-Folding Proteins**. We bin and reweigh Fraction of Native Contacts (FNC) (x-axis) of $\mathbf{C}_\alpha$ atoms based on MSM weights to obtain free energy estimates (y-axis). We only consider residues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

A.8 PROTEIN-SPECIFIC ABLATION RESULTS

We provide below Jensen-Shannon divergece measures against reference trajectories for (reweighted) ensemble observables of each protein. We compare CG-MLFF (Majewski et al., 2023) and EquiJump models at different capacities.

			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