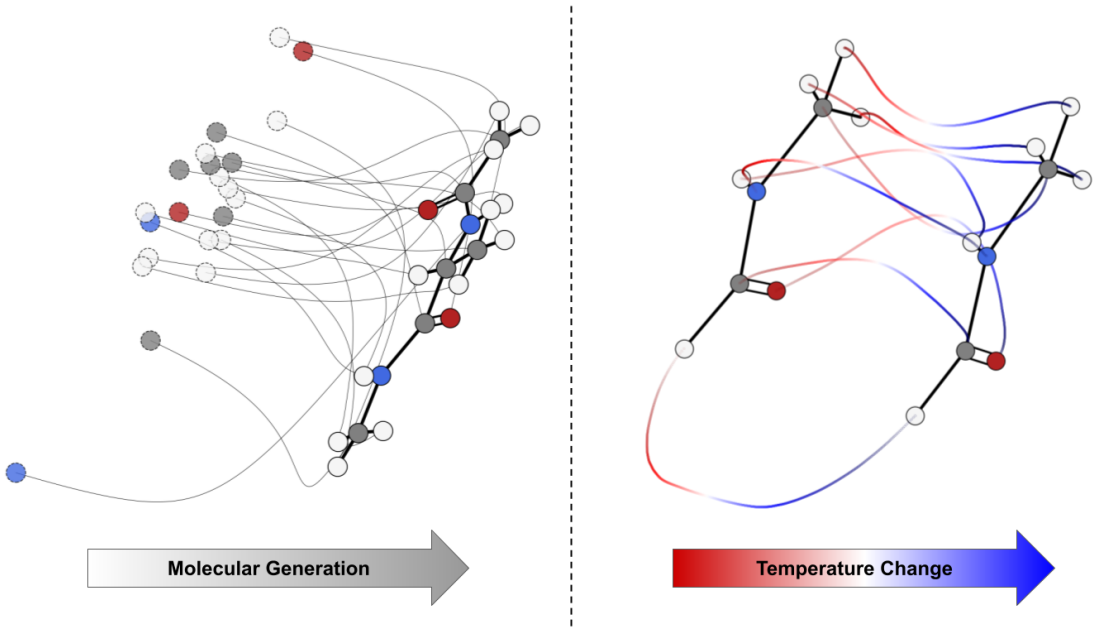
CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG



Generative AI for Molecular Simulations

Speeding Up Molecular Simulations With Stochastic Interpolation

Master's thesis in Computer Science and Engineering

Weilong Chen and Selma Moqvist

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Generative AI for Molecular Simulations

Speeding Up Molecular Simulations With Stochastic Interpolation

Weilong Chen
Selma Moqvist

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Generative AI for Molecular Simulations
Speeding Up Molecular Simulations With Stochastic Interpolation
Weilong Chen and Selma Moqvist

Supervisor: Simon Olsson, Division of Data Science and AI
Examiner: Simon Olsson, Division of Data Science and AI

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Generative AI for Molecular Simulations
Speeding Up Molecular Simulations With Stochastic Interpolation
Weilong Chen and Selma Moqvist
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In statistical mechanics, computing the average behavior of microscopic states is crucial, for example, in estimating observables for equilibrium distributions in molecular systems. The challenge lies in the difficulty of sampling, as the density is known but hard to sample from. Typically, sampling of molecular conformations is performed using molecular dynamics, which faces challenges in obtaining iid samples due to the problem of rare events.

Various enhanced sampling methods have been proposed to tackle this issue. Machine learning, specifically continuous-time generative models, offers a new perspective for tackling this problem. In our thesis, we propose two generative models using the recent Stochastic Interpolants framework. The first learns to transform between equilibrium distributions with different temperatures, which can be further applied with the current replica exchange method. The second model learns transition probability densities across time scales, which can be used as a surrogate model to accelerate MD simulations.

We highlight the ability of Stochastic Interpolants to design efficient sampling methods for many-body systems in different ways, making it a powerful tool for advancing molecular simulation. Our results are two-fold. First, we present our Stochastic Interpolant ITO model and show how it reduces the VAMP-2 score gaps when benchmarked against the original ITO architecture. Next, we showcase our Thermodynamic Interpolant model, that to some extent manages to perform temperature transformations in a setting where it has to generalize beyond the training data. Our advancements show potential and could benefit various fields such as drug discovery, material science, catalysis, and green chemistry.

Keywords: Machine Learning, Deep Generative Models, Molecular Simulations, Message Passing Neural Networks, Stochastic Interpolants.

Acknowledgements

We are grateful to many people who have supported us throughout our journey of completing this master's thesis.

It all began with a course given by Simon, called Structured Machine Learning, which we both took. The course was very well prepared and up-to-date on every subject, something that awakened our interest in Simon's research area. Afterwards, we continued on with a project course in his research group on similar topics, and we are very thankful for being well-informed and cared for right from the beginning. While we were only masters students, we always felt like a part of the team. Throughout the entire process, Simon has spent a lot of time with us and on our work. He provided assistance as much as he could, from computational resources to workspace, from daily discussions to practical collaboration, from theoretical ideas to concrete feedback. He is the best supervisor we could have had, and we highly value the amount of time he has invested in our thesis work.

In addition, we want to thank every member of our group. It has been our pleasure to work with you all. Thank you for the discussions during group meetings and fika sessions, and for always being helpful and open to meetings whenever we have had questions. We would especially like thank Juan, Ross, Mathias, Chris, and Flemming for their support related to our topic. We also want to express our gratitude to Rocío for her assistance and organization. Thank you for sharing your joy and kindness!

We also wish to thank our friends for making our every-day life here at Chalmers fun despite all challenges. Thank you to all the teachers and classmates who built an inspiring learning environment for us. Lastly, we thank our families for their unwavering support and sacrifice for our growth.

Weilong Chen and Selma Moqvist, Gothenburg, 2024-06-07

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

For a long time, studies of molecular systems have been of interest in a multitude of scientific disciplines. Many questions remain, since such systems usually are complex with many constituents in high dimensions, rendering analytical calculations infeasible in a majority of cases. Accurate predictions of molecular properties are still highly relevant, since they often are an important component in applications such as materials design and drug discovery. Thus, efficient ways of studying these systems are desirable, both from a societal and from a research perspective.

Consider a molecule with a given number of atoms, each of them in a three-dimensional space. In order to compute properties of the molecule, we need to compute expectation values. These are taken over the probability distribution that describes the molecule's possible microscopic states. For a molecular system, this distribution is called the Boltzmann distribution, and it describes the probability of finding that system in a certain state. Still, analytical methods often face difficulties due to the intractability of the distribution's normalization constant, the partition function. As an example, consider the Boltzmann distribution of a system at energy E

$$\rho(\mathbf{x}) = \frac{1}{Z} \exp\left(-\beta E(\mathbf{x})\right), \quad Z = \int \exp\left(-\beta E(\mathbf{x})\right) d\mathbf{x},$$

where the partition function Z is obtained by integrating over each dimension of the microscopic states $\mathbf{x}$. Now remember that each atom in a molecule exists in three dimensions, implying that the final dimension of the molecular system will be three times the number of atoms in the molecule. Clearly there is no analytical solution to this integral, at least not in general. However, through sampling of the underlying distribution we can build a Monte Carlo-estimator of the integral and at least obtain a reasonable estimate of the quantity of interest. While, in theory, this seems like an appealing approach, a major issue still remains. The validity of a Monte Carlo-estimator depends on the quality of the samples. More specifically, for the estimator to be valid, we need independent and identically distributed samples. Unfortunately, a general method for obtaining such samples in an efficient manner does not exist.

1.2 Context

While many techniques have been developed to the purpose, sampling molecular configurations with high sampling efficiency is not trivial. Traditional techniques are usually iterative in nature, meaning that a simulation is started from some initial condition from which a solution trajectory can be obtained iteratively.

An example of such a method is molecular dynamics, or MD. In an MD simulation, we define a differential equation describing the dynamics of the molecular system, such that the equilibrium distribution of the dynamics is the distribution we are trying to sample from (i.e. the Boltzmann distribution). Then, the differential equation is numerically integrated from an initial condition to obtain a solution trajectory. At every integration step, we obtain a new molecular microstate that can be regarded as a sample from the underlying distribution. In infinite time, the trajectory will have visited every state and the entire distribution will have been sampled [1].

The issue with such methods is that they do not fulfill the requirement of independence between the samples. Since the samples are obtained iteratively, every new sample will be correlated to the previous one, resulting in any Monte Carlo-estimator becoming biased. While there are methods to combat such correlation issues, the result is still very long simulation times. This issue is magnified in situations where the landscape of the energy function, defining the Boltzmann distribution, is intricate and contains energy barriers [1]. At low temperatures, the likelihood of a sampled trajectory passing such barriers is extremely low, leading to the trajectory becoming "stuck" on one side of the barrier. This, in turn, results in higher sample correlations and even longer run times [1]. Thus, there is a huge interest in faster sampling techniques.

Traditional methods, such as parallel tempering, can be used to speed up molecular simulations in situations similar to the ones previously described [2]. In a parallel tempering scheme, multiple Molecular Dynamics simulations or replicas are run in parallel. The replicas are identical in every aspect apart from the simulation temperature. Since there is an increased probability of passing energy barriers when the temperature is higher, samples from high-temperature replicas can be proposed to the lower-temperature replica that one in practice is trying to sample [1]. While this is just one example of an enhanced traditional sampling method, there are many others available, showing the topic's high relevance [3].

The parallel tempering method is still such that samples are generated by visiting one state after another. A conceptually different way to approach equilibrium sampling is given through the Boltzmann Generator [4]. Here, a normalizing flow model is trained to function as an invertible mapping between a complex system consisting of many particles, and a simpler Gaussian distribution, which is of matching dimension to the original system [5], [6]. When sampling latent coordinates from the simpler distribution, coordinates from the more complex system can be implicitly sampled by transforming the latent coordinates through the learned map [4]. Unfortunately, Boltzmann generators can suffer from mode collapse, leading them to only sample

the most stable state. Still, they are an important milestone in molecular equilibrium sampling through generative AI.

Another generative AI sampling approach, more focused on prediction of dynamical properties, is that of Implicit Transfer Operator (ITO) learning. The idea is that during training, a denoising diffusion probabilistic model (DDPM) is fed with tuples $(\mathbf{x}_t, \mathbf{x}_{t+N\tau}, N)$, where τ is a time lag and N is a natural number [7], [8]. In this manner, the model implicitly learns to reproduce the dynamics of the molecular system, and the distribution can be sampled with larger time steps than in a regular MD simulation. Through sampling with large strides, autocorrelation times can be reduced, resulting in a higher ratio of independent samples [7].

While the original ITO framework was built using DDPMs, there is no reason why another generative AI architecture, like normalizing flows, could not be used as long as it meets the relevant requirements. A subcategory to the normalizing flow is the continuous normalizing flow or CNF. Just as a normalizing flow, a CNF transforms samples from an initial density into samples from a target density, with the difference that the transformation is done in a continuous manner. This is achieved by learning a differential equation that, when solved from initial conditions sampled from the prior, generates samples from the target distribution [9], [10]. The differential equation can be learned in various ways, such as through Flow Matching or by using Stochastic Interpolants. Unlike diffusion models, CNFs allow for transformation between arbitrary densities in finite integration time [11]–[13].

1.3 Goals and Challenges

The overarching goal of this thesis is to speed up molecular simulations using generative AI models. We split this goal into five sub-goals.

1. Firstly, we investigate the usage of Stochastic Interpolants to learn a continuous normalizing flow model for transforming data from a Boltzmann distribution at an initial temperature into samples from a Boltzmann distribution at a target temperature. In this manner, we hope to obtain a generative model for sampling target configurations given an initial ensemble. Using this type of model, we want to investigate if it is possible to make transformations between two thermodynamic ensembles.

2. Next, it is interesting to explore how data-efficient such methods are. Training a map between two ensembles using Stochastic Interpolants requires data at both ensembles, which is rather inefficient. Therefore, it is interesting to see how well this type of map generalizes to sampling from ensembles outside of the model's training data. In that way, there would be a computational gain in using such a model. Thus, our second goal is to also investigate if the learned map generalizes to previously unseen thermodynamic states.

3. Since our main goal is to speed up molecular simulations, we must also investigate if using the thermodynamic interpolant models yield a computational speed up. As such, our third goal is to investigate whether there are any

computational advantages to using the aforementioned methods compared to explicit simulation.

4. Another framework tackling the same problem, but in a different manner, is that of Implicit Transfer Operator learning. Originally the ITO framework was built using diffusion models. However, an implementation of ITO through the Stochastic Interpolants framework would be interesting for comparison and improvement of the architecture. Extending ITO to the Stochastic Interpolant framework also opens up the possibility for extension across temperatures, given that the thermodynamic interpolants are working. Thus, we also investigate if we can implement ITO using Stochastic Interpolants.

5. Lastly, while working with low-dimensional toy systems is useful when developing methods, it is not so relevant for real-world applications. If we want a method or model to work for molecular systems, we also need to test in on molecular systems. Thus, our last goal is to investigate how the methods we have developed scale to higher-dimensional systems, such as molecules.

2

Theory

2.1 Concepts from Statistical Mechanics

Many ideas in generative machine learning for molecular systems have their roots in physics and statistical mechanics. To further the understanding of this thesis, we provide a brief summary of a few such concepts.

2.1.1 Statistical Ensembles

A fundamental concept in statistical mechanics is that of *statistical ensembles*. Consider a collection of identical copies of a system, such as a molecule or some other particle. The copies, described by the same microscopic laws while sharing the same macroscopic properties, define the statistical ensemble. Despite each copy evolving under the same rules in time, their initial conditions differ, resulting in a vast number of unique *microstates* [1].

A system can, on a macroscopic level, be completely characterized through a set of control variables, defining its *thermodynamic state*. If each system in an ensemble share fixed values of the particle number N, volume V and temperature T, the ensemble is referred to as the *canonical ensemble* [1].

2.1.2 Boltzmann Distribution

For a system described by a canonical ensemble, the *Boltzmann distribution* describes the probability of finding the system in a certain microstate. For instance, the distribution of microstates could describe possible 3D-coordinates of a molecule's atoms.

Given microstates $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ of dimension $d \in \mathbb{N}$, and an energy function $E(\mathbf{x}) : \Omega \to \mathbb{R}$, the Boltzmann distribution of E can be written as

$$\rho(\mathbf{x}) = \frac{1}{Z} \exp(-\beta E(\mathbf{x})), \quad Z = \int_{\Omega} \exp(-\beta E(\mathbf{x})) \, \mathrm{d}\mathbf{x}, \tag{2.1}$$

for $\beta^{-1} = kT$, where k is the Boltzmann constant and T is the system's temperature. The constant Z is often referred to as the *partition function* and is, as seen in Equation (2.1), obtained by integrating over all dimensions of $\mathbf{x}$ [1].

2.1.3 Observables

It is possible to gain insights about the properties of a molecular system by calculating *observables*. If a quantity can be measured on the microscopic level, then it could be averaged over the entire ensemble, providing us with an estimate of the corresponding observable. For example, say we are interested in the average bond length between two atoms in a molecule. In theory, one could then calculate the bond lengths for many microstates and average the values over the distribution corresponding to the ensemble to obtain the answer. There are two different types of observables, namely stationary and dynamic observables, which we describe in more detail below.

2.1.3.1 Stationary Observables

Given some quantity of interest that is described by a function $f : \Omega \to \mathbb{R}$, a *stationary observable* is defined as the expected value of f,

$$O_f = \mathbb{E}_{\mathbf{x}\sim\rho(\mathbf{x})}\left[f(\mathbf{x})\right] = \int_\Omega f(\mathbf{x})\rho(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{2.2}$$

where ρ is the Boltzmann distribution defined through Equation (2.1) [7].

2.1.3.2 Dynamic Observables

Unlike stationary observables, *dynamic observables* are quantities that relate the current state of a system to a previous state. As such, there is also a time-component included.

Consider the time-evolution of a molecular system described by the state vector $\mathbf{x}$. Assuming the process is described by a dynamical model that is integrated with a time step τ, the conditional probability density function $\rho_\tau(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)$ describes the time-discrete evolution of $\mathbf{x}$ at $\Delta t = N\tau$ steps into the future, given an initial state $\mathbf{x}_t$. Here, $N \in \mathbb{N}$ and $\tau, t \in \mathbb{R}$. A dynamic observable can then be written as

$$O_{f(t)h(t+\Delta t)} = \mathbb{E}_{\mathbf{x}_t\sim\rho(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}_{t+\Delta t}\sim\rho_\tau(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t)}\left[f(\mathbf{x}_t)h(\mathbf{x}_{t+\Delta t})\right]\right], \tag{2.3}$$

where f, $h : \Omega \to \mathbb{R}$ are two functions describing the microscopic observation processes corresponding to the dynamic observable of interest [7].

2.1.4 Helmholtz Free Energy

Helmholtz free energy is a measure of the amount of work that can be extracted from a thermodynamic system at constant particle number, volume and temperature. In terms of the partition function, Z, it can be written as

$$F = -\beta^{-1}\log Z,$$

with the inverse temperature β [1].

Measurements of *free energy differences* between two thermodynamic states S_0 and S_1 are important since they provide a way of determining whether a process will be favorable from a thermodynamic perspective or not. The free energy difference between states S_0 and S_1 can be written in terms of the partition functions Z_0 and Z_1 as

$$\Delta F = -\beta^{-1} \log \left(\frac{Z_0}{Z_1} \right),$$

at a shared inverse temperature β.

2.1.4.1 Targeted Free Energy Perturbation

Originating from the concept of free energy perturbation [14], the method of *targeted free energy perturbation* offer a way to calculate free energy changes between two thermodynamic states given a way of transforming between them [15]. Let S_0 and S_1 represent these two thermodynamic states, specified by potentials E_0 and E_1 respectively. Furthermore, let S_0 and S_1 be characterized by the canonical distributions

$$\rho_0(\mathbf{x}) = \frac{1}{Z_0} e^{-\beta E_0(\mathbf{x})}, \quad \rho_1(\mathbf{x}) = \frac{1}{Z_1} e^{-\beta E_1(\mathbf{x})}.$$

Assume that $\Omega \subset \mathbb{R}^d$ for some $d \in \mathbb{R}$. Say there exist a transformation of samples $\mathbf{x}_0 \in \Omega$ drawn from the initial density ρ_0 into samples $\mathbf{x}_1 \in \Omega$ from the target density ρ_1, so that $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ is related to $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$ via a differentiable and invertible mapping $\mathcal{M}$, transforming the configuration space onto itself as

$$\mathcal{M} : \mathbf{x}_0 \to \mathbf{x}_1(\mathbf{x}_0).$$

Samples from ρ_0 that have been mapped through $\mathcal{M}$ are then implicitly sampled from ρ_1 and can be expressed as $\mathcal{M}(\mathbf{x}_0) = \mathbf{x}_1 \sim \rho_1(\mathbf{x})$ [15].

Furthermore, let the two densities ρ_0 and ρ_1 be related as

$$\rho_1(\mathbf{x}_1) = \frac{\rho_0(\mathbf{x}_0)}{\det |J(\mathbf{x}_0)|}, \tag{2.4}$$

where J is the Jacobian of the mapping $\mathcal{M}$. With this setup, it is possible to define a function ϕ as

$$\phi(\mathbf{x}_0) \triangleq E_1\left(\mathcal{M}(\mathbf{x}_0)\right) - E_0(\mathbf{x}_0) - \beta^{-1} \log \det |J(\mathbf{x}_0)|. \tag{2.5}$$

Following the approach of targeted free energy perturbation [15], ϕ can be used to calculate the free energy difference ΔF between the two thermodynamic states S_0 and S_1. This is achieved by taking an exponentiated average over samples $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ as

$$\langle e^{-\beta\phi} \rangle_{S_0} = e^{-\beta \Delta F}. \tag{2.6}$$

This relation holds significant importance. Given a method for transforming samples $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ into samples $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$ and a way of calculating the Jacobian determinant of that transformation, it is possible to estimate the change in free energy ΔF between the two states S_0 and S_1 using the function ϕ.

2.1.5 Fokker-Planck Equation

While stationary quantities can provide a lot of insight about a molecular system, the same could be said for time-dependent quantities. By letting each microstate follow a prescribed set of dynamical rules, one can study how it evolves in time. Such dynamics can be specified through stochastic differential equations.

Given a domain $\Omega \subset \mathbb{R}^d$ of dimension $d \in \mathbb{N}$, the stochastic differential equation

$$\mathrm{d}\mathbf{x}_t = \mathbf{b}(\mathbf{x}_t, t)\mathrm{d}t + \boldsymbol{\sigma}(\mathbf{x}_t, t)\mathrm{d}W_t, \quad \mathbf{x}_{t=0} = \mathbf{x}_0, \tag{2.7}$$

with $\mathbf{b}, \boldsymbol{\sigma} \in \mathbb{R}^d$, describes the time evolution of some $\mathbf{x}_t \in \Omega$. Assuming $\mathbf{x}_t \sim \rho_t$ follows the SDE (2.7), then the time-dependent density ρ_t satisfies the partial differential equation

$$\partial_t \rho_t(\mathbf{x}) + \nabla \cdot \mathbf{b}\rho_t(\mathbf{x}) = \frac{1}{2}\Delta\left(\boldsymbol{\sigma}\boldsymbol{\sigma}^T \rho_t(\mathbf{x})\right), \tag{2.8}$$

given that $\rho_{t=0}(\mathbf{x}) = \rho_0(\mathbf{x})$. Equation (2.8) is more widely known as the *Fokker-Planck equation* and describes the time evolution of the density ρ_t [16]. In this way, the individual dynamics of each state can be related to the dynamics of the probability density of states.

2.1.5.1 Special Case 1: Continuity Equation

The SDE (2.7) describes the movements of a system that is put under the action of deterministic and random forces. If the random forces are excluded, so that the SDE (2.7) reduces to the ordinary differential equation

$$\mathrm{d}\mathbf{x}_t = \mathbf{b}(\mathbf{x}_t, t)\mathrm{d}t, \quad \mathbf{x}_{t=0} = \mathbf{x}_0, \tag{2.9}$$

then the Fokker-Planck equation (2.8) reduces to the partial differential equation

$$\partial_t \rho_t(\mathbf{x}) + \nabla \cdot \mathbf{b}\rho_t(\mathbf{x}) = 0, \tag{2.10}$$

with $\rho_{t=0}(\mathbf{x}) = \rho_0(\mathbf{x})$. Equation (2.10) is known as the *continuity equation* [16].

2.1.5.2 Special Case 2: Langevin and Brownian Dynamics

The dynamics of a molecular system can be modelled through the *Langevin dynamics*, a stochastic differential equation describing the movements of a particle or system of particles that is put under the action of deterministic and random forces [17]. In the limit where there is no average particle acceleration, the *Brownian dynamics* is retrieved.

For a particle at energy $E(\mathbf{x})$ that is acted upon by a random force, the Brownian dynamics SDE can be written as

$$\mathrm{d}\mathbf{x}_t = -\gamma^{-1}\nabla E(\mathbf{x}_t)\mathrm{d}t + \sqrt{2\gamma^{-1}\beta^{-1}}\mathrm{d}W_t, \tag{2.11}$$

where γ denotes the friction and β is the inverse temperature [16].

2.1.6 Transfer Operators

In the study of dynamical systems a crucial part is, as previously mentioned, to understand how observables or ensembles evolve continuously over time τ. This evolution can be formally characterized by the transfer operator $T_\tau : L^1(\Omega) \to L^1(\Omega)$

$$[T_\tau \circ \rho](\mathbf{x}_{t+\tau}) \triangleq \frac{1}{\mu(\mathbf{x}_{t+\tau})} \int_{x_t} \mu(\mathbf{x}_t)\, \rho(\mathbf{x}_t)\, \rho(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)\, \mathrm{d}\mathbf{x}_t \tag{2.12}$$

Here, $L^1(\Omega)$ denotes the space of integrable functions on the domain Ω. Specifically, a function $f : \Omega \to \mathbb{R}$ belongs to $L^1(\Omega)$ if the integral of its absolute value over Ω is finite:

$$\int_\Omega |f(x)|\, \mathrm{d}x < \infty.$$

This operator is commonly referred to as the *(Ruelle) Transfer Operator* [7]. Spectral form of this operator is expressed as:

$$T_\Omega(\tau) = \sum_{i=0}^{\infty} \lambda_i(\tau)\, |\psi_i\rangle \langle \phi_i|\,, \tag{2.13}$$

where the eigenvalues $\lambda_i(\tau) = \exp(-\tau\kappa_i)$ are the only elements dependent on the time-step τ. Here, κ_i represents the characteristic 'relaxation' rates associated with the pair of left and right eigenfunctions, ϕ_i and ψ_i,

Alternatively, two more conceptually intuitive definitions exist: the *Perron-Frobenius* operator or propagator $\mathcal{P}_\tau : L^1(\Omega) \to L^1(\Omega)$, defined as

$$[\mathcal{P}_\tau \circ p](\mathbf{x}_{t+\tau}) = \int_{x_t} \rho(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)\, \rho(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t.$$

and the *Koopman* operator $\mathcal{K}_\tau : L^\infty(\Omega) \to L^\infty(\Omega)$, defined by

$$[\mathcal{K}_\tau \circ f](\mathbf{x}_{t+\tau}) = \int_{x_t} \rho(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)\, f(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t = \mathbb{E}[f_t(\mathbf{x}_{t+\tau}) \mid \mathbf{x}_t = x]\,. \tag{2.14}$$

While the Perron-Frobenius operator describes the evolution of densities, the Koopman operator describes the evolution of observables. The relation between these operators can be found in [18]. Applications like *Dynamic Mode Decomposition* (DMD) and *Markov State Models* (MSM) together with their relations are discussed in [19].

2.2 Sampling Algorithms

In previous sections, we have assumed that it is possible to calculate expected values with respect to the Boltzmann distribution analytically. However, in high dimensions, the integrals in Equations (2.2) and (2.3) are generally intractable. Since many molecular systems consist of multiple atoms in three dimensions, analytic calculation of observables is often not feasible.

As an alternative, the distribution ρ can be sampled. The samples can then be used to build a Monte Carlo-estimator of the observable of interest. As an example, for

$M \in \mathbb{N}$ iid samples $\{\mathbf{x}_i\}_{i=1}^{M}$, $\mathbf{x}_i \sim \rho(\mathbf{x})$, such that $\mathbf{x}_i \in \Omega \subset \mathbb{R}^d$ and $d \in \mathbb{N}$, the integral from Equation (2.2) reduces to a sum, so that

$$\int_{\Omega} f(\mathbf{x})\rho(\mathbf{x})\mathrm{d}\mathbf{x} = \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}_i), \tag{2.15}$$

when $M \to \infty$ [1]. To exploit the estimator a large amount of samples is needed, making efficient sampling methods for molecular systems highly desirable.

2.2.1 Numerical Integration of the Langevin Dynamics

The Brownian dynamics SDE (2.11) can be solved iteratively to obtain a trajectory of the states the system has visited throughout time. The solution at each iteration step can be obtained as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma^{-1}\nabla E(\mathbf{x}_t)\Delta t + \sqrt{2\Delta t\gamma^{-1}\beta^{-1}}\mathbf{z}, \tag{2.16}$$

where $\Delta t \in \mathbb{R}$ is the integration step used in each iteration and $\mathbf{z} \sim \mathcal{N}(0, \mathrm{Id}) \in \mathbb{R}^d$ [20].

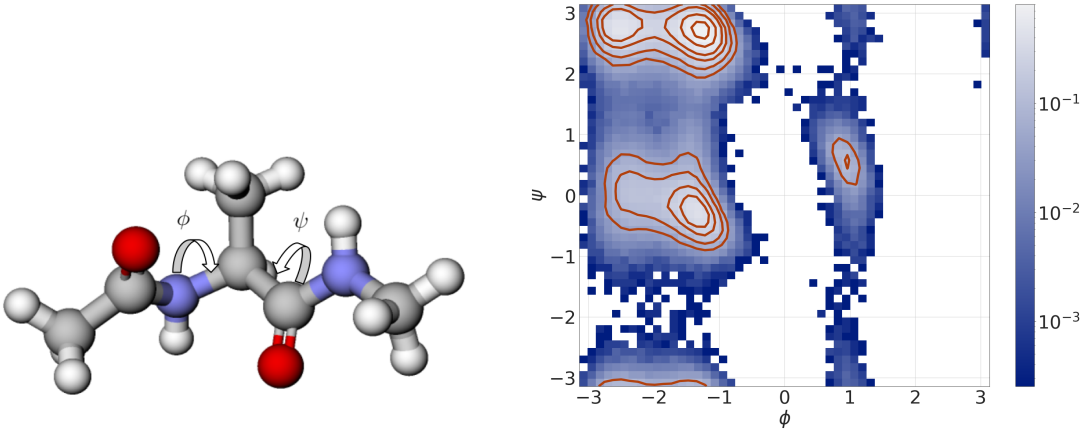
Since the equilibrium distribution of the stochastic process in Equation (2.16) is a Boltzmann distribution with energy function E, the states obtained from the iterative process can be considered to be sampled from this distribution. These samples will however not be drawn iid, since each new sample will be correlated to the previous one. This implies that the Monte Carlo-estimator (2.15) might not be valid for samples generated by the numerical integration scheme in Equation (2.16).

2.2.2 The Problem of Rare Events

In previous sections, it has been assumed that there is access to iid samples from the Boltzmann distribution that can be used for estimating observables. However, that is not always the case. Many molecules have complicated energy landscapes with barriers of very low transition probability between the modes of their corresponding probability distributions [1].

As an example, consider Figure 2.1, depicting a two-dimensional histogram of the backbone *dihedral angles* ϕ and ψ (also known as *torsion angles*) of an alanine dipeptide molecule. The molecule can exist in multiple stable conformations, characterized by the angles ϕ and ψ. This makes the molecule's energy landscape complicated, with multiple minima, and the corresponding probability distribution of the torsion angles is multimodal with low probability of transition between the modes [1].

Due to this, a simulated trajectory obtained through e.g. integration of the Langevin equation might not even pass through all modes in a feasible simulation time. The event that a random perturbation of the current state moves the simulated chain to a different distribution mode is so low that it barely happens - it is a *rare event*. If the full energy landscape cannot be explored properly, then it is not reasonable to expect that the Monte Carlo-estimator in Equation (2.15) will yield an accurate approximation either.



(a) Illustration of the torsion angles ϕ and ψ. (b) 2D-histogram of the torsion angles ϕ and ψ.

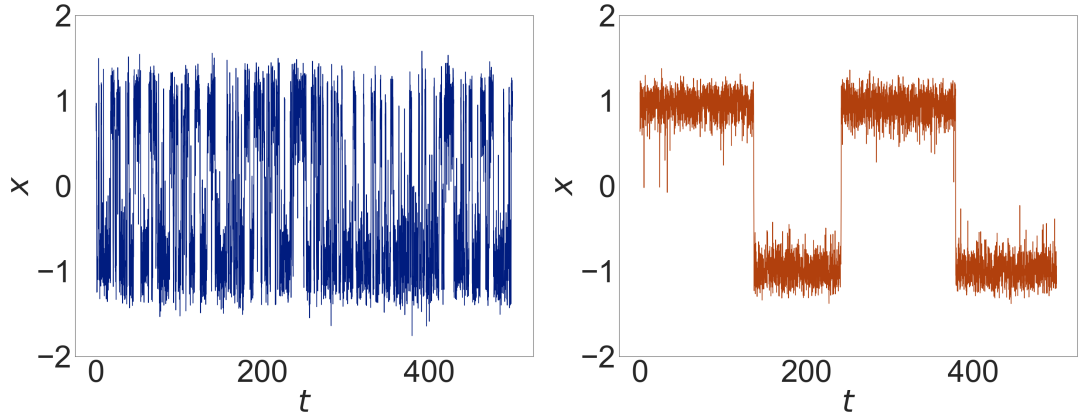
Figure 2.1: An illustration of the rare event problem for an alanine dipeptide molecule. Subfigure 2.1(a) shows a visualization of the torsion angles in the molecule, characterizing the stable conformations of the molecule. Subfigure 2.1(b) shows a log-scale 2D-histogram of the torsion angles for a single alanine dipeptide molecule, obtained through a long simulation trajectory of the system. As can be seen in Subfigure 2.1(b), the probability distribution is multimodal with low passage probability between the modes.

2.2.3 The Replica Exchange Method

At higher temperatures movement, or *mixing*, between the different modes occur faster. For example, consider Figure 2.2, which depicts the bimodal distribution of positions of a one-dimensional particle following an asymmetric double well potential and the corresponding sampled trajectories.

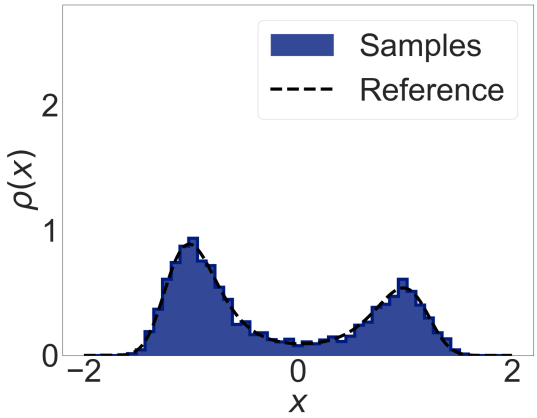
As can be seen in Figure 2.2, a higher temperature also implies a higher transition probability between the two modes. Thus, sampling can be done faster at higher temperatures, since the mixing is faster. Even more, the trajectory of samples will be less correlated compared to if the sampling had been done at a lower temperature.

This property is exploited in the *replica exchange* method. In a replica exchange scheme, simulations are run on a set of identical copies of a system. Each system is assigned a different value of a given physical control variable, for example the temperature, so that they each correspond to a different thermodynamic state. Then, it is possible to exchange coordinates of the different replicas by making Monte Carlo moves between them [1]. When the physical control variable is set to be the system's temperature, the replica exchange method is referred to as *parallel tempering* [1], [2].

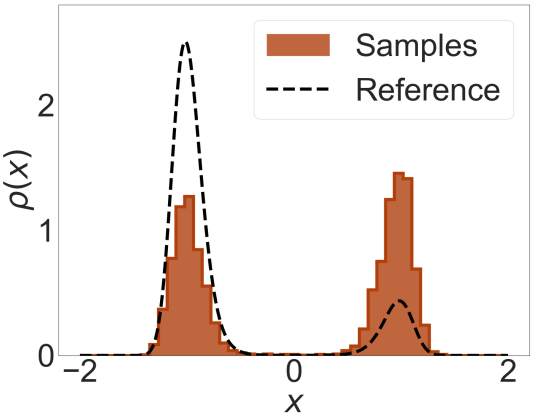


(a) Trajectory sampled at high temperature.

(b) Trajectory sampled at low temperature.



(c) Histogram of samples at high temperature.



(d) Histogram of samples at low temperature.

Figure 2.2: A comparison of trajectories and samples obtained at a high versus a low temperature. Subfigures 2.2.3 and 2.2.3 shows a sampled trajectories at high and low temperatures respectively. As can be seen in the plot of the trajectory simulated at high temperature, mixing is much higher compared to in the low-temperature trajectory. Further, as illustrated by comparison of Subfigures 2.2.3 and 2.2.3, the sample quality is much higher when simulating at a higher temperature.

An illustration of the parallel tempering idea is seen in Figure 2.3. The scheme is constructed so that there are N replicas of the system. Each replica is assigned a unique value of the temperature so that the temperatures $T_0, T_1, \ldots, T_N$, where $T_0 < T_1 < \ldots < T_N$, each correspond to one replica. The scheme is setup in such a way that the simulation temperature of interest is taken to be the lowest temperature, i.e. T_0.

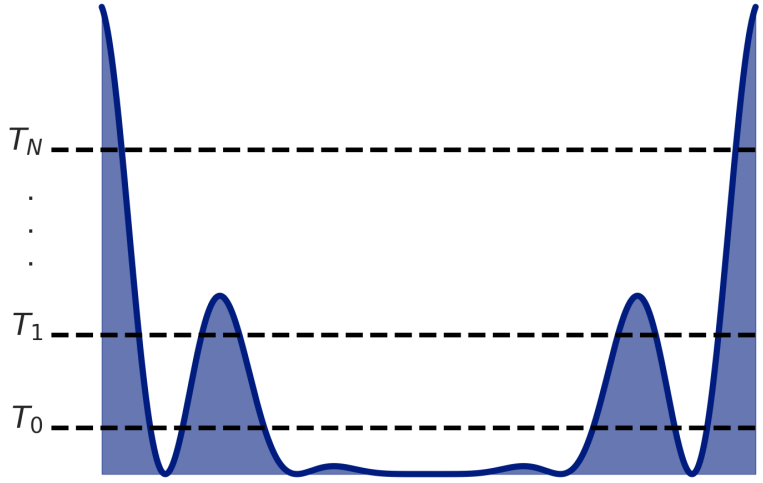


Figure 2.3: An illustration of a parallel tempering scheme in a complicated energy landscape, where the temperatures are chosen such that $T_0 < T_1 < \ldots < T_N$. The x-axis shows the possible states, while the y-axis shows the values of the energy function. As can be seen in the figure, the higher-temperature replicas can move through the majority of the energy landscape without encountering too many energy barriers since the noise level is higher. The low-temperature replicas can, however, sample the energy function in more detail but moves through the energy landscape with more difficulty.

This means that a replica can be simulated at a higher temperature, where mixing is faster due to higher transition probabilities. Then a Monte Carlo move can be made, so that the coordinates of the higher temperature replica are proposed to the lower temperature replica that one is interested in simulating.

As an example, denote the states of two neighboring replicas by $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$. A move from $(\mathbf{x}^{(k)}, \mathbf{x}^{(k+1)})$ to $(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})$ can then be made with the acceptance probability

$$A\left(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)} \mid \mathbf{x}^{(k)}, \mathbf{x}^{(k+1)}\right) = \min\left[1, \exp\left(-\Delta_{k,\,k+1}\right)\right], \tag{2.17}$$

where $\Delta = -(\beta_k - \beta_{k+1})\left[E(\mathbf{x}^{(k)}) - E(\mathbf{x}^{(k+1)})\right]$. By making many such moves, simulations in lower temperatures can be significantly sped up. Note however that in a real Replica Exchange Molecular Dynamics (REMD) simulation, one would have to account for momenta as well as positional coordinates. This complicates the simulation slightly, since for Equation (2.17) to hold the momenta needs to be updated separately [21].[1]

[1]Note that to maintain simplicity and relevancy in the text, we do not discuss this in more depth here and instead refer the reader to the original source [21].

2.3 Sampling with Generative AI

Generative AI provides a different perspective on sampling. Using neural networks, it is possible to learn a map from some easy-to-sample distribution, e.g. a standard Gaussian, to some target data-generating distribution of interest. In this way, one can build a model that approximates the data generating process and enable sampling of the target distribution. Note that given independent samples from the initial distribution, the transformed samples will also be independent. Thus, such *surrogate models* ρ_θ of the data density ρ could be designed to not suffer from the same correlation issues as many traditional sampling methods.

When modelling the data density in this manner, one should also account for *approximation errors* in the model itself. By using *importance weights* $w_i \in \mathbb{R}$, $i = 1, \dots M$, defined through

$$w_i = \frac{\hat{\rho}(\mathbf{x}_i)}{\rho_\theta(\mathbf{x}_i)}, \tag{2.18}$$

where $\mathbf{x}_i \sim \rho_\theta$, ρ_θ is the surrogate distribution and $\hat{\rho}$ is the unnormalized density, one can compensate for the approximation error when estimating observables. For example, we can use the importance weights to calculate observables through

$$O_f = \mathbb{E}[f(\mathbf{x})] \approx \frac{1}{M} \sum_{i=1}^{M} w_i f(\mathbf{x}_i),$$

to obtain a more accurate estimate of the observable O_f. Therefore, it is important to use a surrogate model that enables evaluation of the sample probabilities. In this section, we describe how one can use generative AI surrogate models to draw samples from the target distribution and obtain their corresponding sample probabilities.

2.3.1 Normalizing Flows

An important class of neural network model architectures, employing such a scheme, is that of *normalizing flows*. In a normalizing flow, an initial distribution ρ_0 is transformed into a target distribution ρ_1 by exploiting neural network models and the *change-of-variables formula*. If a learned map $\mathcal{M}_\theta : \mathbb{R}^d \to \mathbb{R}^d$, with $d \in \mathbb{N}$, is trained to map initial samples $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ into target samples $\mathbf{x}_1 = \mathcal{M}_\theta(\mathbf{x}_0) \sim \rho_1(\mathbf{x})$, then the endpoint density can be evaluated as

$$\rho_1(\mathbf{x}) = \rho_0(\mathcal{M}_\theta^{-1}(\mathbf{x}_1)) \det \left| \frac{\partial \mathcal{M}_\theta^{-1}(\mathbf{x}_1)}{\partial \mathbf{x}_1} \right| = \frac{\rho_0(\hat{\mathbf{x}}_0)}{\det \left| \frac{\partial \mathcal{M}_\theta(\hat{\mathbf{x}}_0)}{\partial \hat{\mathbf{x}}_0} \right|}, \tag{2.19}$$

where $\hat{\mathbf{x}}_0 = \mathcal{M}_\theta^{-1}(\mathcal{M}_\theta(\mathbf{x}_0))$. In this way a neural network map can be learned that allow for both sample transformation and evaluation of the sample probabilities. However, in order to apply Equation (2.19), the map $\mathcal{M}_\theta$ needs to be both differentiable and invertible. In some architectures, e.g. RealNVP, this is done by applying

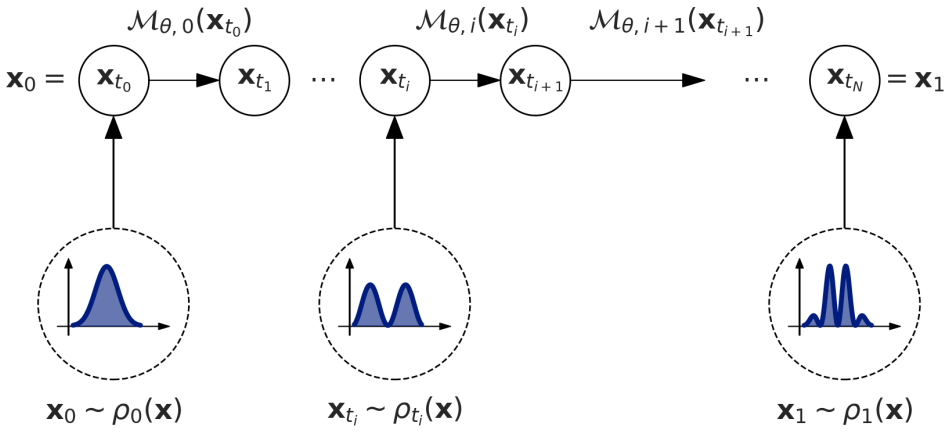


Figure 2.4: An illustration of a discretized map $\mathcal{M}_\theta = \mathcal{M}_{\theta,N} \circ \mathcal{M}_{\theta,N-1} \circ \ldots \circ \mathcal{M}_{\theta,0}$ when applied to an initial standard normal distribution.

a series of N partial transformations $\mathcal{M}_\theta = \mathcal{M}_{\theta,N} \circ \mathcal{M}_{\theta,N-1} \circ \ldots \circ \mathcal{M}_{\theta,0}$ to the input, such that their composition corresponds to $\mathcal{M}_\theta$ as is shown in Figure 2.4 [22].

However, discretizing the map $\mathcal{M}_\theta$ quickly gets complicated as the number of partial transformations increase and preferably one would rather model $\mathcal{M}_\theta$ as one continuous map.

2.3.2 Continuous Normalizing Flows

A possible way of modelling continuous density transformations is to exploit ordinary differential equations [10]. By defining an ODE that describes the flow of a time-dependent probability density, such that it at an initial time corresponds to the prior distribution and at an endpoint time corresponds to the data distribution, samples can be transformed in a continuous manner [10]. Figure 2.5 below shows an example of a continuous transformation of an initial density ρ_0 into an endpoint density ρ_1.

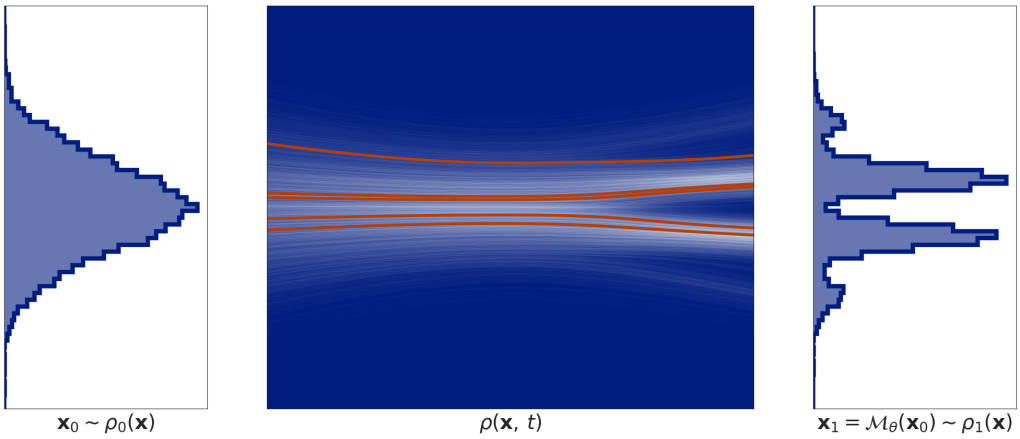


Figure 2.5: An illustration of how the initial density ρ_0 in the left histogram continuously transforms into a target density ρ_1 such that $\mathbf{x}_1 = \mathcal{M}(\mathbf{x}_0) \sim \rho_1(\mathbf{x})$ (see the right histogram). The red lines are example trajectories, showing the transformation on a per-sample level.

A way to achieve a density transformation similar to the one in Figure 2.5 is through *continuous normalizing flows*. Assume initial states $\mathbf{x}_0 \in \Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, which follow an initial distribution ρ_0 and time-dependent states $\mathbf{x}_t \in \Omega$, which follow a time-dependent distribution ρ_t. Further, assume that each $\mathbf{x}_t$ obeys the ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_t = \mathbf{b}(t,\, \mathbf{x}_t), \quad \mathbf{x}_{t=0} = \mathbf{x}_0, \tag{2.20}$$

where the vector field $\mathbf{b} : [0,\, 1] \times \Omega \to \mathbb{R}^d$, $d \in \mathbb{R}$, describes the velocity of $\mathbf{x}_t$. Then samples $\mathbf{x}_0 \sim \rho_0$ can be transformed into samples $\mathbf{x}_t \sim \rho_t(\mathbf{x})$ by learning a velocity $\mathbf{b}_\theta$ and integrating

$$\mathbf{x}_t = \mathbf{x}_0 + \int_0^t \mathbf{b}_\theta(s, \mathbf{x})\mathrm{d}s,$$

with a neural ODE solver [10]. In this way, a map that allows for time-continuous sample transformations is obtained. Interestingly, such continuous transformations also allow for evaluation of sample probabilities through a *continuous change of variables*. Note that the ODE (2.20) is the same as Equation (2.9). Thus, the time-evolution of the density ρ_t will be described through the continuity equation from Section 2.1.5.1, so that

$$\frac{\partial}{\partial t}\rho_t + \nabla \cdot (\rho_t \mathbf{b}) = 0. \tag{2.21}$$

The differential equation can be solved more easily in log-space, yielding the expression

$$\log \rho_t(\mathbf{x}) = \log \rho_0(\mathbf{x}) - \int_0^t \nabla \cdot \mathbf{b}_\theta(s,\, \mathbf{x})\mathrm{d}s, \tag{2.22}$$

for a learned velocity $\mathbf{b}_\theta$. Equation (2.22) can also be integrated using a neural ODE solver, providing us with a way of computing sample probabilities. This type of flow model is called a continuous normalizing flow.

2.3.3 Stochastic Interpolants

Still, the question remains of how to learn the velocity field $\mathbf{b}_\theta$. A neat solution is provided in the Stochastic Interpolant framework [12]. There, the velocity $\mathbf{b}_\theta$ is learned by interpolating between individual samples from the prior distribution ρ_0 and the data distribution ρ_1. A significant advantage is that the framework allows for the prior to be an arbitrary distribution, so ρ_0 could be anything, even another data distribution [12].

2.3.3.1 Definition of the Interpolant

The idea behind the framework is to interpolate between individual samples $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ and $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$, where the initial distribution ρ_0 and the target distribution ρ_1 are such that $\rho_0,\ \rho_1 : \mathbb{R}^d \to \mathbb{R}^+$. The *stochastic interpolant* is defined as a stochastic process $\mathbf{x}_t$ such that

$$\mathbf{x}_t = I(t,\, \mathbf{x}_0,\, \mathbf{x}_1) + \gamma(t)\mathbf{z}, \quad t \in [0,\, 1]. \tag{2.23}$$

where $I(0,\, \mathbf{x}_0,\, \mathbf{x}_1) = \mathbf{x}_0$, $I(1,\, \mathbf{x}_0,\, \mathbf{x}_1) = \mathbf{x}_1$, $\gamma(0) = \gamma(1) = 0$ and $\mathbf{z} \sim \mathcal{N}(0,\, \mathrm{Id})$[2].

In many cases interpolation can be done in a spatially linear manner, so that the interpolant takes the form

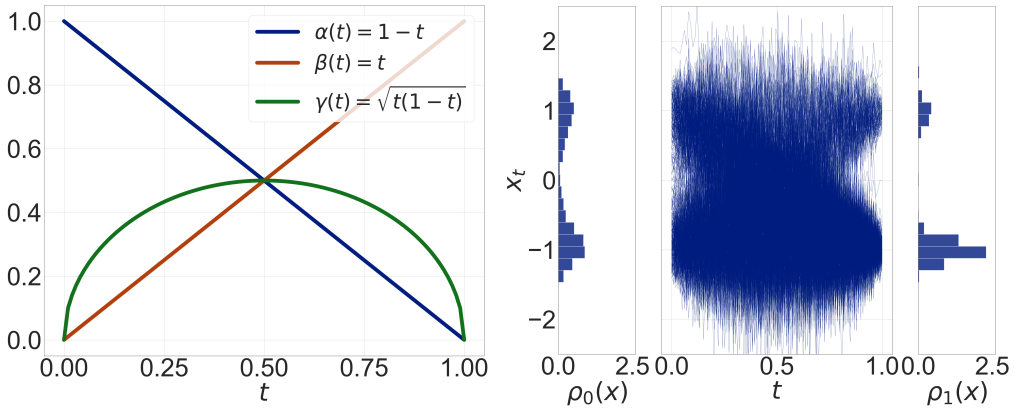
$$\mathbf{x}_t = \alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{x}_1 + \gamma(t)\mathbf{z},$$

where the functions α, β and γ can be chosen in any way that respects the constraints defined above (noting that the β in this context should not be confused with the inverse temperature of the Boltzmann distribution). As an example, consider the simple choice of functions

$$\alpha(t) = 1 - t, \quad \beta(t) = t, \quad \gamma(t) = \sqrt{t(1-t)}.$$

This choice results in a "noised" linear interpolation, defining a path of transportation from the prior to the end-point distribution on a per-sample level. Figure 2.6(a) shows the behaviour of the three functions α, β and γ on the time interval $t \in [0,\, 1]$. Figure 2.6(b) shows an example how an initial distribution ρ_0 of one-dimensional states x_0 transforms into a target distribution ρ_1 of one-dimensional states x_1 by interpolating between individual samples from the two distributions.

[2]Note that since the formal definition of the stochastic interpolant is quite long and complicated, we only define the major properties here. Furthermore, we refer the reader to Appendix A.1.1 for the full definition.



(a) A linear choice of the three interpolant functions α, β and γ

(b) Initial and final distributions along with in-between interpolated coordinates x_t.

Figure 2.6: Subfigure 2.6(a) shows an example with a linear choice of the three interpolant functions α, β and γ. Subfigure 2.6(b) shows histograms of the initial and final distributions ρ_0 and ρ_1, together with 500 x_t:s evaluated on a grid of interpolation times t ranging from 0 to 1 for the linear interpolant choice in Subfigure 2.6(a). The example distributions ρ_0 and ρ_1 correspond to the distribution of states for a one-dimensional particle following an asymmetric double well potential at two different temperatures.

2.3.3.2 Velocity field

As previously mentioned, an interpolant on the form of Equation (2.23) can be used to learn a *velocity field*, denoted $\mathbf{b}_\theta$. Furthermore, the velocity can be used to define a continuity equation, similar to Equation (2.21), for the time-dependent probability distribution $\rho_t(\mathbf{x})$ of the stochastic interpolant.

Let C^k denote the space of $k \in \mathbb{N}$ times continuously differentiable functions. In terms of the stochastic interpolant (2.23), the velocity field $\mathbf{b}$ is then defined as the expected value of the interpolated sample velocities, e.g.

$$\mathbf{b}(t, \mathbf{x}) = \mathbb{E}\left[\dot{\mathbf{x}}_t \middle| \mathbf{x}_t = \mathbf{x}\right], \tag{2.24}$$

and is in $C^0([0, 1]; (C^p(\mathbb{R}^d))^d)$ for $p \in \mathbb{N}$ [12].

2.3.3.3 Score field

In addition to the velocity field, the stochastic interpolants framework allow the possibility to learn another vector field; the *score* of the probability density of the stochastic interpolant. The score is in $C^1\left([0, 1]; (C^p(\mathbb{R}^d))^d\right)$ for any $p \in \mathbb{N}$, and is given as

$$\mathbf{s}(t, \mathbf{x}) = \nabla \log \rho(t, \mathbf{x}) = \gamma^{-1}(t)\mathbb{E}\left[\mathbf{z} \middle| \mathbf{x}_t = \mathbf{x}\right], \tag{2.25}$$

for all $(t, \mathbf{x}) \in (0, 1) \times \mathbb{R}^d$ [12].

2.3.3.4 Objective functions

Both the velocity (2.24) and the score (2.25) are learned quantities, approximated through neural networks. A velocity field $\mathbf{b}_\theta$ can be learned by minimizing the quadratic objective

$$\mathcal{L}_{\mathbf{b}}[\mathbf{b}_\theta] = \int_0^1 \mathbb{E}\left[\frac{1}{2}|\mathbf{b}_\theta(t,\, \mathbf{x}_t)|^2 - (\partial_t I(t,\, \mathbf{x}_0,\, \mathbf{x}_1) + \dot{\gamma}(t)\mathbf{z}) \cdot \mathbf{b}_\theta(t,\, \mathbf{x}_t)\right] \mathrm{d}t, \tag{2.26}$$

for a stochastic interpolant defined through Equation (2.23), with the expectation being taken independently over $(\mathbf{x}_0,\, \mathbf{x}_1) \sim \nu$ and $\mathbf{z} \sim \mathcal{N}(0,\, \mathrm{Id})$. Furthermore, $\mathbf{b}_\theta$ is the unique minimizer of the objective in Equation (2.26) in $C^0\left([0,\, 1];\, (C^1(\mathbb{R}^d))^d\right)$ [12]. As with Flow Matching, various forms of optimal transport can be employed during training to further optimize the integration path lengths [23], [24].

Similarly to the velocity field, an approximation $\mathbf{s}_\theta$ can be learned of the original score field $\mathbf{s}$ by training a neural network model to minimize a quadratic objective function. The score field $\mathbf{s}$ is the unique minimizer in $C^1\left([0,\, 1];\, (C^1(\mathbb{R}^d))^d\right)$ of the quadratic objective defined as

$$\mathcal{L}_{\mathbf{s}}\left[\mathbf{s}_\theta\right] = \int_0^1 \mathbb{E}\left[\frac{1}{2}|\mathbf{s}_\theta(t,\, \mathbf{x}_t)|^2 + \gamma^{-1}(t)\mathbf{z} \cdot \mathbf{s}_\theta(t,\, \mathbf{x}_t)\right] \mathrm{d}t, \tag{2.27}$$

where the expectation is taken independently over $(\mathbf{x}_0,\, \mathbf{x}_1) \sim \nu$ and $\mathbf{z} \sim \mathcal{N}(0,\, \mathrm{Id})$ [12].

2.3.3.5 Special Case 1: Mirror Interpolant

The definition of the stochastic interpolant (2.23) allows for an interesting special case, namely the *mirror interpolant*. When the initial distribution is chosen to be identical to the end-point distribution, i.e. $\rho_0(\mathbf{x}) = \rho_1(\mathbf{x})$, it results in an identity map. A stochastic interpolant that fulfills this choice of distributions is

$$\mathbf{x}_t = \mathbf{x}_0 + \gamma(t)\mathbf{z}, \quad t \in [0,\, 1], \tag{2.28}$$

with $\mathbf{x}_0 \sim \rho_0(\mathbf{x}) = \rho_1(\mathbf{x})$ and $\mathbf{z} \sim \mathcal{N}(0,\, \mathrm{Id})$. The interpolant defined through Equation (2.30) is called a mirror interpolant. Interestingly, for such an interpolant it holds

$$\mathbf{b}(t,\, \mathbf{x}) = -\dot{\gamma}(t)\gamma(t)\mathbf{s}(t,\, \mathbf{x}), \tag{2.29}$$

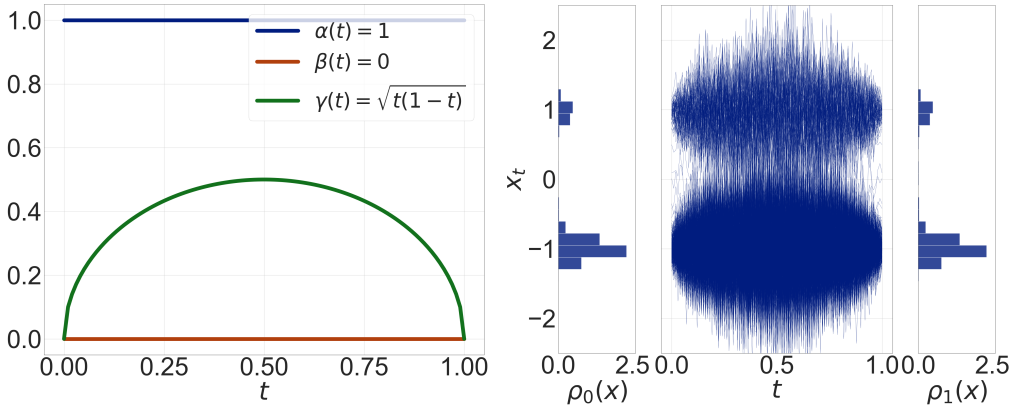
so that the velocity field $\mathbf{b}$ is directly related to the score field [12].

Just as the regular stochastic interpolant (2.23), the mirror interpolant (2.30) can be linearly decomposed into three functions α, β and γ

$$\alpha(t) = 1, \quad \beta(t) = 0, \quad \gamma(t) = \sqrt{t(1-t)}.$$

This choice results in a "noised" identity interpolation, defining a path of transportation from a prior distribution to itself, again on a per-sample level. Figure 2.7(a)

shows the behaviour of the three functions α, β and γ on the time interval $t \in [0, 1]$. Figure 2.6(b) shows an example how an initial distribution ρ_0 of one-dimensional states x_0 transforms into itself through mirror interpolation. In this sense, the mirror interpolant defines an identity map.



(a) A linear choice of the three interpolant functions α, β and γ

(b) Initial and final distributions along with in-between interpolated coordinates x_t.

Figure 2.7: Subfigure 2.7(a) shows an example choice of the three functions α, β and γ that results in a mirror interpolant. Subfigure 2.7(b) shows histograms of the initial and final distributions ρ_0 and ρ_1, together with 500 x_t:s evaluated on a grid of interpolation times t ranging from 0 to 1 for the mirror interpolant choice in Subfigure 2.7(a). The example distributions ρ_0 and ρ_1 are equal and correspond to the distribution of states for a one-dimensional particle following an asymmetric double well potential at one single temperature.

2.3.3.6 Special Case 2: One-sided Interpolant

The definition of the stochastic interpolant (2.23) allows for another special case, namely the *one-sided interpolant*. This interpolant employs a Gaussian prior, which is a common choice in generative modelling considering the absence of prior information. In this setting, we can group the effect of the latent variable $\mathbf{z}$ with $\mathbf{x}_0$, since they are both normally distributed.

The initial distribution is chosen to be identical to the Gaussian, i.e. $\rho_0(\mathbf{x}) \sim \mathcal{N}(0, \mathrm{Id})$. A stochastic interpolant that fulfills this choice of distributions is

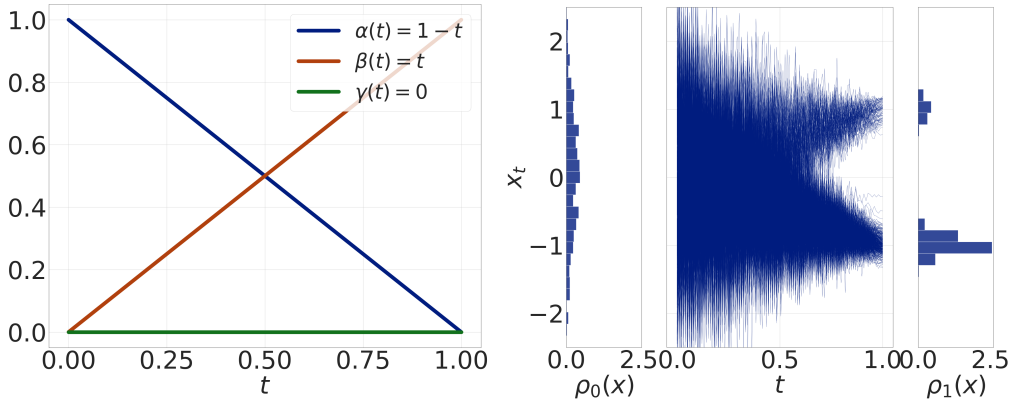
$$\mathbf{x}_t = J(t, \mathbf{x}_1) + \alpha(t)\mathbf{z}, \quad t \in [0, 1], \tag{2.30}$$

with $J(0, \mathbf{x}_1) = 0$, $J(1, \mathbf{x}_1) = \mathbf{x}_1$ and $\alpha(0) = 1, \alpha(1) = 0$. The interpolant defined through Equation (2.30) is called a one-sided interpolant.

Just as the regular stochastic interpolant (2.23), the one-sided interpolant (2.30) can be built through simple linear interpolant with $\alpha(t) = 1 - t$, $\beta(t) = t$ and $\gamma(t) = 0$ so that it takes the form

$$\mathbf{x}_t = t\mathbf{x}_1 + (1 - t)\mathbf{z}.$$

This choice results in linear interpolation, defining a linear transformation path from a Gaussian prior to the target distribution. Figure 2.8(a) shows the behaviour of the three functions α, β and γ on the time interval $t \in [0, 1]$. Figure 2.8(b) shows an example of how an initial distribution ρ_0 of one-dimensional Gaussian samples x_0 transforms into the target distribution ρ_1.



(a) A linear choice of the three interpolant functions α, β and γ

(b) Initial and final distributions along with in-between interpolated coordinates x_t.

Figure 2.8: Subfigure 2.8(a) shows an example choice of the three functions α, β and γ that results in a one-sided interpolant. Subfigure 2.8(b) shows histograms of the initial and final distributions ρ_0 and ρ_1, together with 500 x_t:s evaluated on a grid of interpolation times t ranging from 0 to 1 for the one-sided linear choice in Subfigure 2.8(a). $\rho_0 \sim \mathcal{N}(0, \mathrm{Id})$ and ρ_1 corresponds to the distribution of states for a one-dimensional particle following an asymmetric double well potential at one single temperature.

2.3.4 Denoising Diffusion Probabilistic Models

Score-based diffusion models are based on the Ornstein-Uhlenbeck process, for which the corresponding SDE is shown as

$$\mathrm{d}\mathbf{x}_t = -\mathbf{x}_t \mathrm{d}t + \sqrt{2}\mathrm{d}W_t, \quad \mathbf{x}_{t=0} \sim \rho_1(\mathbf{x}) \tag{2.31}$$

This process has the property that as t tends toward infinity, the marginal density of its values converges to a normal distribution. By learning the score $s(t, \mathbf{x}_t)$, the associated backward SDE for the SDE in Equation (2.31) can be written, which can be further used as a generative model. One typical learning method for the score is through (denoising) score matching, which offers a new perspective on training generative models. Instead of matching distributions, we can match scores by solving a regression problem.

The solution of Equation (2.31) with initial condition $\mathbf{x}_{t=0} = \mathbf{x}_1 \sim \rho_1(\mathbf{x})$ can be written as

$$\mathbf{x}_t = \mathbf{x}_1 e^{-t} + \sqrt{2} \int_0^t e^{-t+s} \mathrm{d}W_s.$$

This is also the law of the process

$$\mathbf{y}_t = \mathbf{x}_1 e^{-t} + \sqrt{1 - e^{-2t}}\mathbf{z}, \quad \mathbf{z} \sim N(0, \mathrm{Id}), \quad t \in [0, \infty). \tag{2.32}$$

Note that in diffusion model, $\mathbf{y}_t$ converges to a Gaussian when $t \to \infty$. In score-based diffusion models, the typical way of achieving this is to limit the evolution of $\mathbf{x}_t$ to a finite interval $[0, T]$ with $T < \infty$ and then used backward SDE restricted within $[0, T]$. This introduces a bias which is not shown in stochastic interpolants with fixed time interval $[0, 1]$. However, by defining $t' = e^{-t}$ and by choosing $\alpha(t')$ and $\beta(t')$, we can turn Equation (2.32) into stochastic interpolants. It is shown that casting SBDM into a one-sided stochastic interpolant (2.33) allows the construction of unbiased generative models that operate on $t \in [0, 1]$ [12].

$$\mathbf{y}_{t'=-\log t} = \sqrt{1 - t'^2}\mathbf{z} + t'\mathbf{x}_1 \quad \text{for} \quad \alpha(t') = \sqrt{1 - t'^2}, \quad \beta(t') = t'. \tag{2.33}$$

2.4 Implicit Transfer Operators

As described in Section 2.1.5.2, a common approach in molecular dynamics simulations is to model a system using the Brownian dynamics. A numerical method, such as Euler-Maruyama, can be put to serve as a time-discretization scheme with a time-step denoted by τ. Consequently, simulating the dynamics of the system, represented by Equation (2.11), corresponds to simulating a Markov chain with transition probability density in the following format:

$$\rho\left(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t, \tau\right) = \mathcal{N}\left(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t - \tau \nabla E\left(\mathbf{x}_t\right) \gamma^{-1}, \tau\sqrt{2D}\,\mathrm{Id}_{3M}\right). \tag{2.34}$$

where $\mathcal{N}$ denotes the Normal distribution, $D = \gamma^{-1}\beta^{-1}$, and Id_{3M} is the $3M$-dimensional identity matrix. Stability of the entire system can be ensured if τ is sufficiently small, requiring a large number of steps for iteration over long time scales, which is important for calculating the observables (as described in Section 2.1.3). Hence, single-step sampling of long-time-scale dynamics to approximate the transition probability $\rho(\mathbf{x}_{N\tau} \mid \mathbf{x}_0)$ can be very helpful. The spectral representation of the transfer operator, as indicated in Equation (2.12), allows the generating transition probability to be expressed in the following form, as proved in [7].

$$\rho\left(\mathbf{x}_{N\tau} \mid \mathbf{x}_0\right) = \sum_{i=1}^{\infty} \underbrace{\lambda_i^N(\tau)}_{\text{time-variant}} \underbrace{\alpha_i\left(\mathbf{x}_{N\tau}\right)\beta_i\left(\mathbf{x}_0\right)}_{\text{time-invariant}}$$

Here, α_i and β_i are the time-invariant projection coefficients of the state variables onto the left and right eigenfunctions ϕ_i and ψ_i of the transfer operator, respectively, and $\lambda_i(\tau)$ is its i'th eigenvalue.

Instead of learning this transfer operator through explicit numerical simulation, another approach is to build a surrogate model using deep generative models. This approach is thus called *Implicit Transfer Operator* (ITO) learning.

2.5 Machine Learning for Molecular Systems

Training machine learning models, more specifically neural networks, for molecular systems involves learning in high dimensional spaces. A neural network can be regarded as an estimated function, $f_\theta \in \mathcal{F}$ for some class of functions $\mathcal{F}$, mapping the input data to some output value. However, function estimation quickly gets complicated as the dimension is increasing. As an example, take the Lipschitz class of functions, e.g. functions $f : \Omega_1 \to \Omega_2$ such that for $\Omega_1 \subset \mathbb{R}^{d_1}$, $\Omega_2 \subset \mathbb{R}^{d_2}$,

$$\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| = c\|\mathbf{x}_1 - \mathbf{x}_2\|, \tag{2.35}$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \Omega_1$ [25]. Assuming an approximation f_θ of f, it is possible to measure the accuracy of f_θ on previously unseen data through the *generalization error*. Furthermore, the generalization error, denoted by ε, is bounded as

$$\varepsilon \lesssim n^{-1/d}, \tag{2.36}$$

for n data points of dimension d [25]. Thus, the error of the approximated function f_θ grows exponentially with the dimension. However, as will be described in this section, this phenomena can partly be counteracted by incorporating geometric knowledge about the data into the neural network, i.e. the approximated function f_θ.

2.5.1 Molecular Symmetries

Molecular systems are subject to a number of *symmetries*. By forcing the neural network, again viewing it as a function approximator, to only consider functions that respect these symmetries, the available data can be handled more efficiently [25]. Regard the alanine dipeptide molecule discussed in Section 2.2.2. As can be seen in Figure 2.9, *rotation* around the center of mass will not alter the molecule. Neither will a *translation* of its center of mass or *reflection* around a given coordinate axis do so. Despite these transformations, the molecule will remain the same.

These molecular symmetries can be described mathematically through the concept of *group theory*.

2.5.2 Group Theory Basics

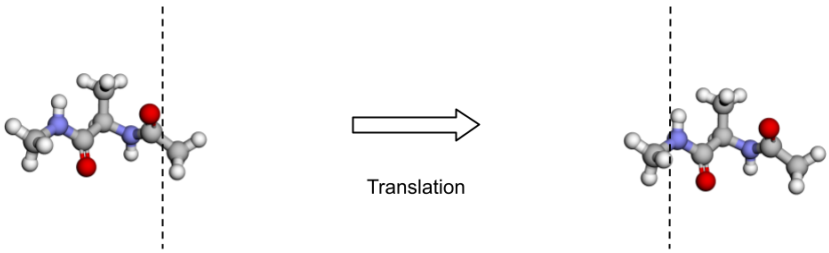
Formally, a *group* is defined as a set $\mathfrak{G}$ along with a binary operation $\circ : \mathfrak{G} \times \mathfrak{G} \to \mathfrak{G}$ called *composition*, satisfying the following axioms

1. *Associativity*, so that

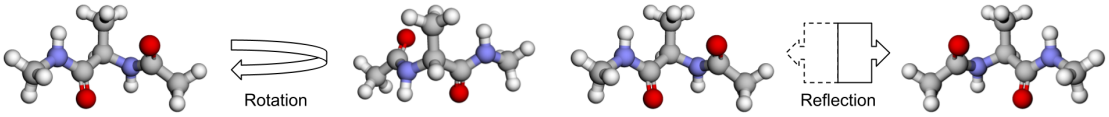
$$(\mathfrak{g}\mathfrak{h})\mathfrak{k} = \mathfrak{g}(\mathfrak{h}\mathfrak{k}) \text{ for all } \mathfrak{g},\ \mathfrak{h},\ \mathfrak{k} \in \mathfrak{G}.$$

2. *Identity*, so that there exists a unique $\mathfrak{e} \in \mathfrak{G}$ satisfying

$$\mathfrak{e}\mathfrak{g} = \mathfrak{g}\mathfrak{e} = \mathfrak{g} \text{ for all } \mathfrak{g} \in \mathfrak{G}.$$



(a) Translational symmetry.



(b) Rotational symmetry.

(c) Reflection symmetry.

Figure 2.9: An illustration of symmetries in an alanine dipeptide molecule.

3. *Inverse*, so that for each $\mathfrak{g} \in \mathfrak{G}$ there is a unique inverse $\mathfrak{g}^{-1} \in \mathfrak{G}$ such that
$$\mathfrak{g}\mathfrak{g}^{-1} = \mathfrak{g}^{-1}\mathfrak{g} = \mathfrak{e}.$$
4. *Closure*, so that the group is closed under composition, i.e. for every $\mathfrak{g}$, $\mathfrak{h} \in \mathfrak{G}$ it is also true that $\mathfrak{g}\mathfrak{h} \in \mathfrak{G}$.

The molecular symmetries discussed in Section 2.5.1 form a group [25].

2.5.2.1 Group Action

A group can act on data through *group actions*. A group action of $\mathfrak{G}$ on a set $\Omega \in \mathbb{R}^d$ is defined as a mapping $(\mathfrak{g}, \mathbf{u}) \to \mathfrak{g}.\mathbf{u}$ associating a group element $\mathfrak{g} \in \mathfrak{G}$ and a point $\mathbf{u} \in \Omega$ with some other point on Ω in a way that is consistent with the group operations [25].

Furthermore, assuming a space of signals $\mathcal{X}(\Omega)$ on Ω and a group $\mathfrak{G}$ acting on Ω, an action of $\mathfrak{G}$ on $\mathcal{X}(\Omega)$ is also obtained as
$$(\mathfrak{g}.\mathbf{x})(\mathbf{u}) = \mathbf{x}(\mathfrak{g}^{-1}\mathbf{u}),$$
where $\mathbf{x} \in \mathcal{X}(\Omega)$ and $\mathbf{u} \in \Omega$ [25].

2.5.2.2 Group Representation

In order to work with groups in a more tangible way, it is possible to use the *group representation*. A d-dimensional real representation of a group $\mathfrak{G}$ is a map $R : \mathfrak{G} \to \mathbb{R}^{d\times d}$, assigning to each $\mathfrak{g} \in \mathfrak{G}$ an invertible matrix $R(\mathfrak{g})$, while satisfying the condition
$$R(\mathfrak{g}\mathfrak{h}) = R(\mathfrak{g})R(\mathfrak{h})$$
for all $\mathfrak{g}$, $\mathfrak{h} \in \mathfrak{G}$ [25].

2.5.2.3 Invariance

Applying the group action to the input of a function might result in it not changing the output at all. This property is called *invariance* to the group action. More formally, a function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is $\mathfrak{G}$-invariant if

$$f(R(\mathfrak{g})\mathbf{x}) = f(\mathbf{x}),$$

for all $\mathfrak{g} \in \mathfrak{G}$ and $\mathbf{x} \in \mathcal{X}(\Omega)$, i.e. the output of f is not affected when applying the group action to the input [25].

2.5.2.4 Equivariance

Another property that some functions have is *equivariance* to a group action. A function $f : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$ is $\mathfrak{G}$-equivariant if

$$f(R(\mathfrak{g})\mathbf{x}) = R(\mathfrak{g})f(\mathbf{x}),$$

for all $\mathfrak{g} \in \mathfrak{G}$. Thus, applying the group action to the input affects the output in an equivalent manner [25].

2.6 Equivariant Flow Models

As a motivation for incorporating this type of geometrical knowledge, formally known as *geometric priors*, into a neural network, the Lipschitz class of functions can be regarded again. Given a group $\mathfrak{G}$, of cardinality $|\mathfrak{G}|$, it is possible to restrict the generalization error as

$$\varepsilon \lesssim (|\mathfrak{G}|n)^{-1/d}.$$

Thus, if symmetries of a large group can be taken into account then the error can be reduced without needing more data points [25], [26]. Furthermore, concepts from group theory can be incorporated in a neural network flow model, such as a stochastic interpolant flow [27].

More specifically, let ρ be a density on $\Omega_1 \subset \mathbb{R}^d$ that is invariant to actions of a group $\mathfrak{G}$. Further, let $\mathfrak{G}$ be such that $\mathfrak{G} > \mathfrak{H}$ for another group $\mathfrak{H}$. If $f : \Omega_1 \to \Omega_2$ is a $\mathfrak{H}$-equivariant diffeomorphism, then ρ_f is $\mathfrak{H}$-invariant [27]. So, if the continuity equation (2.21) is defined through an $\mathfrak{H}$-equivariant vector field and the initial density ρ_0 is invariant w.r.t. action of $\mathfrak{G}$, then the resulting endpoint distribution ρ_1 will be invariant with respect to action of $\mathfrak{H}$ [27].

2.6.1 Equivariant Flow Models for Molecular Systems

Translating the molecular symmetries described in Section 2.5.1 to group theory, one specific group can be identified. The *Euclidean group* in three dimensions, denoted E(3), consists of all spatial transformations that preserves the Euclidean distance

between two points, e.g. translations, rotations and reflections. This implies that rotations around and translations of the molecule's center of mass, as well as reflections, are part of the E(3) group. Therefore, when working with molecular systems, there is need for a neural network architecture that respects these symmetries in an equivariant manner.

For a task such as sampling of molecular configurations, it might be more fitting to use an SE(3)-equivariant architecture rather than an E(3)-equivariant one. Datasets of molecular samples often consist of one single molecule that has been generated through a long MD-simulation. In general, this molecule will not change parity during the simulation. This property is described by the SE(3)-group, rather than the E(3)-group. The *special Euclidean group* SE(3) is a subgroup to the Euclidean group. It is comprised of only rotations and translations, hence excluding reflections. By using an SE(3)-equivariant setup the model will not sample mirror copies of molecules (i.e. chiral molecules), which is desirable in the context of molecular simulations [7].

2.6.2 PaiNN and ChiroPaiNN Architectures

ChiroPaiNN is an SE(3)-equivariant neural network architecture for molecular systems [7]. Similarly to the E(3)-equivariant PaiNN architecture [28], it has a *message passing* type of neural network architecture. Each atom in the molecule becomes a node in a "graph-like" structure. Then, through a series of neural network layers, every node can send and receive messages from its neighboring atoms, which can then be used to update the node's current state.

2.6.2.1 Inputs and Internal Feature Representations

Inputs to the original PaiNN architecture are spatial coordinates of all atoms i in the molecule, $\mathbf{r}_i \in \mathbb{R}^3$, and their respective atom numbers, $Z_i \in \mathbb{N}$. The architecture is designed in such a way that features are divided into two categories, *invariant features* and *equivariant features*, with the symmetry being regarded with respect to the E(3)-group. The invariant features are treated in a way that respects their symmetries, so that they remain E(3)-invariant. In an equivalent manner, the equivariant features are treated in a way that respects their E(3)-equivariance.

The invariant representations are denoted by $\mathbf{s}_i^m$, and initialized as learned embeddings of the atom numbers as $\mathbf{s}_i^0 = \mathbf{a}_{Z_i} \in \mathbb{R}^{F\times 1}$ for a fixed number of features $F \in \mathbb{N}$. The equivariant representations are denoted by $\mathbf{v}_i^m$ and initialized to zero as $\mathbf{v}_i^0 = \mathbf{0} \in \mathbb{R}^{F\times 3}$ for, again, a fixed number of features $F \in \mathbb{N}$ [28].

2.6.2.2 Constructing Messages

Construction of messages is done through continuous-filter convolutions [28], [29]. For the invariant features, this convolution is defined through

$$\Delta\mathbf{s}_i^m = (\phi_s(\mathbf{s}) * \mathcal{W}_s)_i = \sum_{j\in\mathcal{N}(i)} \phi_s(\mathbf{s}_j) \circ \mathcal{W}_s\left(\|\mathbf{r}_{ij}\|\right), \tag{2.37}$$

for a neural network function ϕ_s that is applied in an atom-wise manner, the neighborhood of atom i is given through $\mathcal{N}(i) = \{j : \|r_{ij}\| \leq r_{\text{cutoff}}\}$ for a cutoff radius r_{cutoff} and where $\mathcal{W}_s$ is a rotationally invariant filter. In the case of the original PaiNN architecture, the filters $\mathcal{W}_s$ are constructed through linear combinations of radial basis functions defined as

$$\text{RBF} = \frac{\sin\left(\frac{n\pi}{r_{\text{cutoff}}}\|\mathbf{r}_{ij}\|\right)}{\|\mathbf{r}_{ij}\|}, \quad 1 \leq n \leq 20, \tag{2.38}$$

and where the cutoff radius r_{cutoff} defines the neighborhood $\mathcal{N}(i)$ for atom i. Thereafter, a cosine cutoff is applied to the filters [28].

In the ChiroPaiNN architecture, the construction of the filters $\mathcal{W}_s$ is handled in a slightly different way. There, the distances are encoded into a positional encoding as

$$\Lambda^n_{\text{pos}}(\|\mathbf{r}_{ij}\|) = \begin{cases} \cos\left(\frac{\pi}{l_0}\left(1 + \frac{n}{2}\right)\|\mathbf{r}_{ij}\|\right) \\ \sin\left(\frac{\pi}{l_0}\left(1 + \frac{n-1}{2}\right)\|\mathbf{r}_{ij}\|\right), \end{cases} \tag{2.39}$$

and then linearly combined into rotationally invariant filters. In the positional encoding (2.39), $l_0 \in \mathbb{R}$ is a hyperparameter [7].

Regarding equivariant features, continuous-filter convolutions are yet again employed for construction of messages. In the case of PaiNN, the message function is given as

$$\Delta\mathbf{v}^m_i = \sum_{j \in \mathcal{N}(i)} \mathbf{v}_j \circ \phi_{vv}(\mathbf{s}_j) \circ \mathcal{W}_{vv}\left(\|\mathbf{r}_{ij}\|\right) + \sum_{j \in \mathcal{N}(i)} \phi_{vs}(\mathbf{s}_j) \circ \mathcal{W}'_{vs}\left(\|\mathbf{r}_{ij}\|\right) \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}, \tag{2.40}$$

where ϕ_{vv} and ϕ_{vs} are two neural network functions. The first term in the equivariant message function (2.40) consists of an invariant filter, $\mathcal{W}_{vv}$, that is convolved with scaled equivariant features, $\mathbf{v}_j \circ \phi_{vv}(\mathbf{s}_j)$. Note that scaling the equivariant features by the invariant features maintains equivariance. Equivalently, the convolution of an equivariant quantity and an invariant filter will also maintain equivariance. The second term in Equation (2.40) is a convolution of invariant features with an equivariant filter. Note that while $\mathcal{W}'_{vs}$ is an invariant quantity, $\frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}$ is an equivariant quantity, thus ensuring that the resulting filter is equivariant. Since $\mathcal{W}'_{vs}$ is another invariant quantity it can be modelled directly through a neural network [28].

The equivariant messages differ a bit in the ChiroPaiNN architecture, compared to the PaiNN architecture [7], [28]. In difference to the original PaiNN architecture [28], ChiroPaiNN is not equivariant with respect to parity (i.e. reflections). This is achieved through the addition of a cross product in the message function. Since the cross product of two regular polar vectors returns a pseudovector, taking the cross product between two polar vectors in the message function will break the equivariance with respect to reflections, ensuring that the architecture is SE(3)-equivariant [7].

2.6.2.3 Update Rules

Once the messages have been constructed, the node features can be updated. The updates are applied in an atom-wise manner. The update function for the invariant features is given as

$$\Delta \mathbf{s}_i^u = \mathbf{a}_{ss}\left(\mathbf{s}_i,\, \|\mathbf{V}\mathbf{v}_i\|\right) + \mathbf{a}_{sv}\left(\mathbf{s}_i,\, \|\mathbf{V}\mathbf{v}_i\|\right) \langle \mathbf{U}\mathbf{v}_i,\, \mathbf{V}\mathbf{v}_i \rangle, \tag{2.41}$$

where $\mathbf{a}_{ss}$ and $\mathbf{a}_{sv}$ are two neural networks. Similarly the update function for the equivariant features is given as

$$\Delta \mathbf{v}_i^u = \mathbf{a}_{vv}(\mathbf{s}_i,\, \|\mathbf{V}\mathbf{v}_i\|)\mathbf{U}\mathbf{v}_i. \tag{2.42}$$

3

Methods

3.1 Model Systems

Four different datasets were used for evaluating the models. To more easily be able to develop and evaluate new methods, the models were first tested on two low-dimensional toy systems. Still, one of the goals of this thesis is to investigate how such models scale to larger systems. Thus, the more successful methods were also evaluated on higher-dimensional molecular systems. In this section, we describe each model system in more detail.

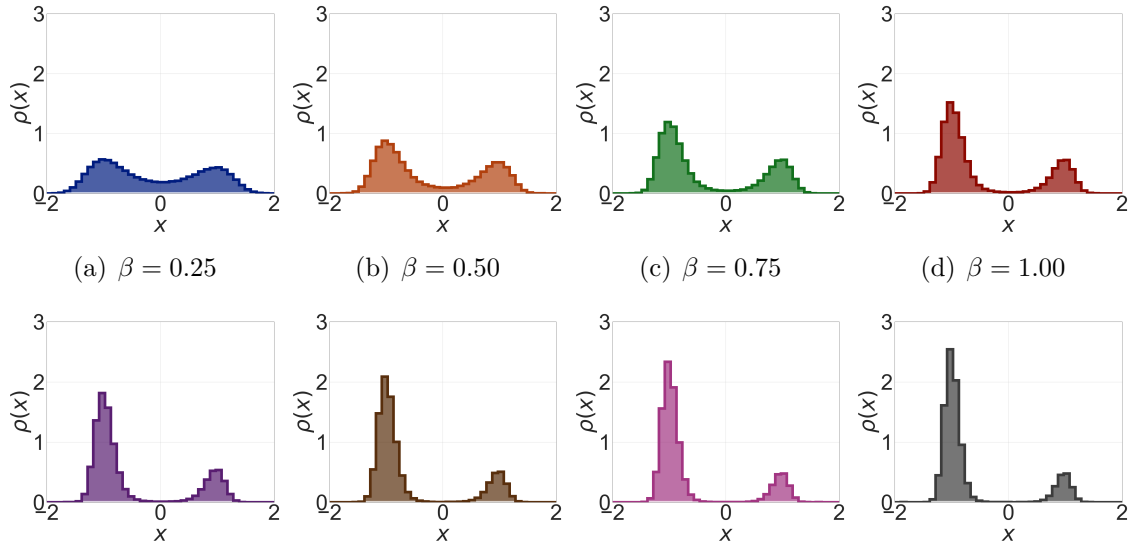
3.1.1 Asymmetric Double Well (ADW)

The first model system was very simple and consisted of a one-dimensional particle confined in an asymmetric double-well potential, defined through

$$E(x) = a(x^2 - 1)^2 + bx, \tag{3.1}$$

with $a = 4$ and $b = 0.5$ for $x \in \mathbb{R}$.

A dataset was generated by sampling the Boltzmann distribution associated with this potential. The sampling was done through numerical integration of the Brownian dynamics (2.16) across the range of inverse temperature values seen in Figure 3.1. More details on the dataset generation, such as hyperparameters for the simulation, can be found in Appendix A.2.1.



(a) $\beta = 0.25$ (b) $\beta = 0.50$ (c) $\beta = 0.75$ (d) $\beta = 1.00$

(e) $\beta = 1.25$ (f) $\beta = 1.50$ (g) $\beta = 1.75$ (h) $\beta = 2.00$

Figure 3.1: Histograms of sampled particle coordinates for different values of the inverse temperature β. The dataset was generated through numerical integration of the Brownian dynamics (2.16).

3.1.2 Müller Brown (MB)

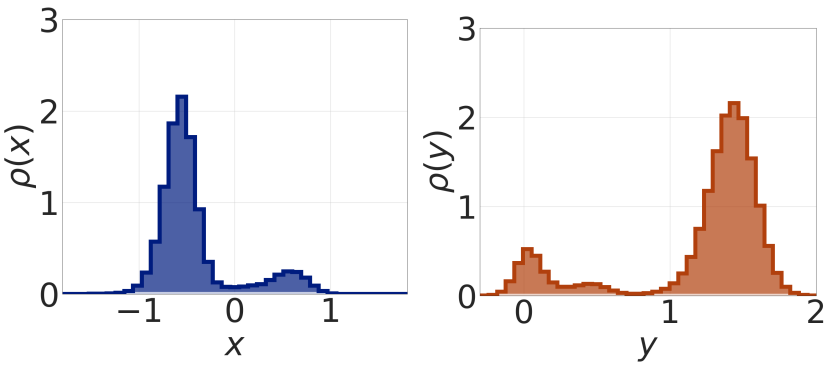
Another relatively simple system is a two-dimensional particle under the Müller Brown potential, defined through Equation (3.2)

$$E(x,y) = \sum_{i=1}^{4} A_i \exp\left[a_i\,(x-\bar{x}_i)^2 + b_i\,(x-\bar{x}_i)\,(y-\bar{y}_i) + c_i\,(y-\bar{y}_i)^2\right]. \tag{3.2}$$

The sampling was done by simulating Brownian dynamics SDE, through an Euler-Maruyama time discretization, where:

$$\begin{aligned} A &= (-200,\ -100,\ -170,\ 15)\\ a &= (-1,\ -1,\ -6.5,\ 0.7)\\ b &= (0,\ 0,\ 11,\ 0.6)\\ c &= (-10,\ -10,\ -6.5,\ 0.7)\\ \bar{x} &= (1,\ 0,\ -0.5,\ -1)\\ \bar{y} &= (0,\ 0.5,\ 1.5,\ 1). \end{aligned}$$

Except from model training, this dataset was also used for benchmarking purposes to compare with the DDPM ITO model, so the same hyperparameters are aligned. The data distributions are shown in Figure 3.2. More details on the dataset generation, such as hyperparameters for the simulation, can be found in Appendix A.2.1.



(a) Histogram of x-coordinates. (b) Histogram of y-coordinates.

Figure 3.2: Histograms of sampled particle coordinates under MB potential.

3.1.3 Alanine Dipeptide (ALA2)

Alanine dipeptide is a small peptide with 22 atoms, commonly used for benchmarking machine learning models for molecular systems. Figure 3.3 shows a visualization of an alanine dipeptide molecule.

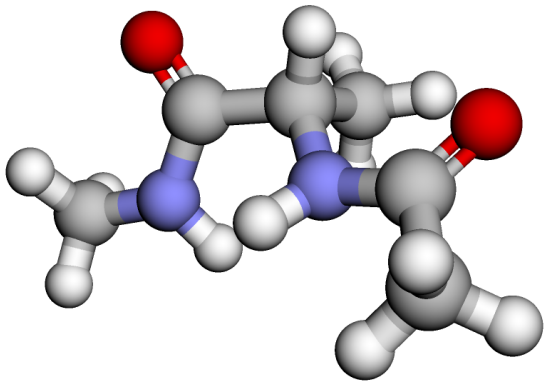


Figure 3.3: Visualization of an alanine dipeptide molecule.

The dataset is openly accessible online and consists of a long MD-trajectory of 750 000 samples [30], [31]. Since one of the goals of this thesis is to compare the Stochastic Interpolants ITO model to the original diffusion model architecture, we benchmarked our model on the alanine dipeptide dataset, i.e. the same system used in the original ITO paper [7].

3.1.4 Replica Exchange QM9 (MDQM9-nc)

While the ADW and MB datasets are simple, the generalization error (2.36) grows exponentially with the dimension. Thus, it is important to note that good results obtained on the one-dimensional particle simulated in the ADW dataset does not directly translate into good results on a molecular system. Therefore, we also tested our methods on molecular systems from the MDQM9-nc dataset [32].

The MDQM9-nc dataset consists of approximately Boltzmann distributed non-cyclic molecular samples, simulated through molecular dynamics. About 10% of the molecules have been simulated through the parallel tempering replica exchange method, described in Section 2.2.3 [32]. It is this subset of the MDQM9-nc dataset we used when testing on molecular systems. Since this data was generated through the replica exchange method, we are provided with samples of high quality at multiple temperatures. Specifically, each system consists of between 9-25 atoms and has been simulated for every temperature $T \in$ [300K, 400K, 500K, 600K, 700K, 800K, 900K, 1000K]. Figure 3.4 shows a few example molecules from the dataset.

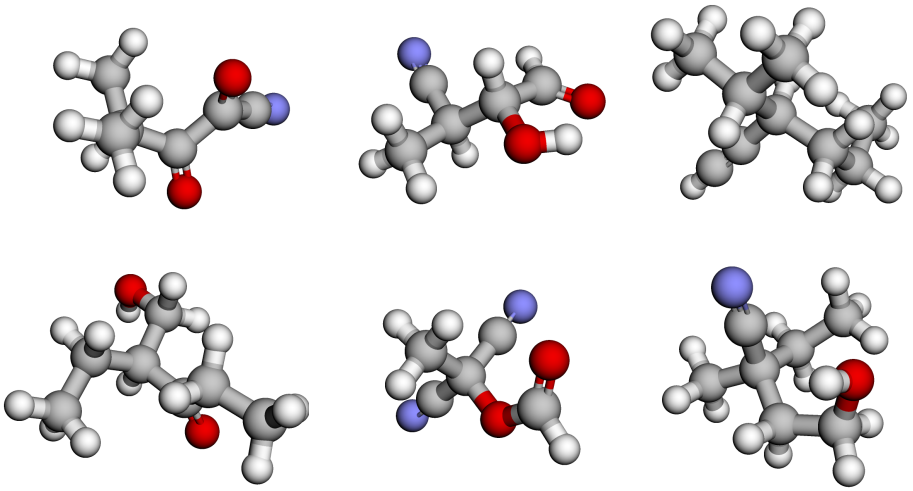


Figure 3.4: Example molecules from the replica exchange MDQM9-nc dataset.

3.2 Stochastic Interpolants ITO Model

In order to accelerate molecular simulations across multiple time and space resolutions, we introduce the *Stochastic Interpolants ITO*.

The implicit Transfer Operator (ITO) is an effective way to learn multiple time-step surrogate models of the stochastic generating distribution of MD. While the original paper [7] uses DDPM, we implement it with the Stochastic Interpolants framework without bias, as described in section 2.3.4.

3.2.1 Interpolation Methods

Instead of directly mapping between two arbitrary distributions e.g. two Boltzmann distributions, ITO model requires conditional inputs such as time lag and initial positions. The overall aim remains to learn a surrogate model to approximate $\rho\left(\mathbf{x}_{N\tau} \mid \mathbf{x}_0\right)$.

For dataset creation, rather than sampling from two equilibrium distributions, we integrated a time lag into the dataset construction process. The original data format can be as straightforward as trajectories from MD simulations. During data preprocessing, the option for either stochastic or fixed lag should be enabled. This

was accomplished by iterating through the MD trajectories, where each data point has initial position $\mathbf{x}_{0,N}$ with time lag N and its corresponding position $\mathbf{x}_{1,N}$.

For the ITO model, we used a one-sided interpolant for training as described in 2.3.3.6. Unlike the diffusion model where the path is diffusion process through learning a score field, we employed one-sided interpolant to specify a flow path that maps also from Gaussian distribution to target distribution by learning a velocity field. The learning objective is outlined as Equation 3.3:

$$\mathcal{L}_{\mathbf{b}}[\mathbf{b}_\theta] = \int_0^1 \mathbb{E}\left[\frac{1}{2}|\mathbf{b}_\theta(t,\, \mathbf{x}_t, N, \mathbf{x}_{0,N})|^2 - \partial_t I(t, \mathbf{z}, \mathbf{x}_{1,N}) \cdot \mathbf{b}_\theta(t,\, \mathbf{x}_t, N, \mathbf{x}_{0,N})\right] \mathrm{d}t, \quad (3.3)$$

During inference, unlike the typical method of using the evaluation part of a split dataset, a conditional dataset should be created by fixing the $\mathbf{x}_{0,N}$ and N. There are two ways of sampling: direct sampling and ancestral sampling. Direct sampling involves passing the desired time lag directly, while ancestral sampling involves splitting the time lag into steps and running inference several times to obtain the output.

3.2.2 Neural Network Model Architectures

For low-dimensional systems such as ADW and MB, a Multi-layer Perceptron (MLP) is sufficient to learn the vector field. However, for larger systems like molecules, geometric priors should be considered, and a message passing neural network architecture can be helpful.

3.2.2.1 Low-Dimensional System (MB)

For this two-dimensional system, an MLP is sufficient to learn the vector field, and handling the conditional inputs $\mathbf{x}_0$ and N can be done in two ways: either by concatenating them with noise samples from a Gaussian distribution or by passing them through a positional encoder first and then concatenating. The network can either learn the vector field directly or learn the residual between the vector field and the initial position $\mathbf{x}_0$. There is no clear evidence showing that any particular method provides a significant improvement in performance for low-dimensional systems, so no specific recommendation is made. A typical model architecture for Müller Brown system is shown in Figure 3.5.

3.2.2.2 Molecular Systems (ALA2)

For molecular systems, we use the same architecture as the original ITO paper. PaiNN is modified to remove parity and is called ChiroPaiNN as described in 2.6.2. The atom type and initial positions are used to construct nodes and edges, typical of graph neural networks. The atomic coordinates are standardized before model training, each atom has a unique nominal embedding as atom type. For the conditional inputs, such as time lag N and interpolation time t, they are converted into positional embeddings and added to the invariant features. There is a separate embedding procedure for $\mathbf{x}_0$ where the features of initial noise samples are replaced

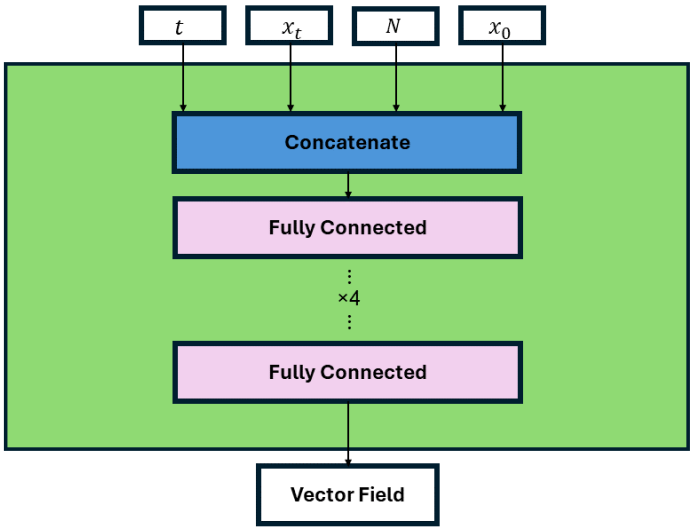


Figure 3.5: An overview of the vector field used for stochastic ITO model

by conditional features. The network input thus consists of noise samples with conditional features, and the output is the residual between the vector field and initial positions. The model architecture for Stochastic Interpolant ITO is shown in Figure 3.6.

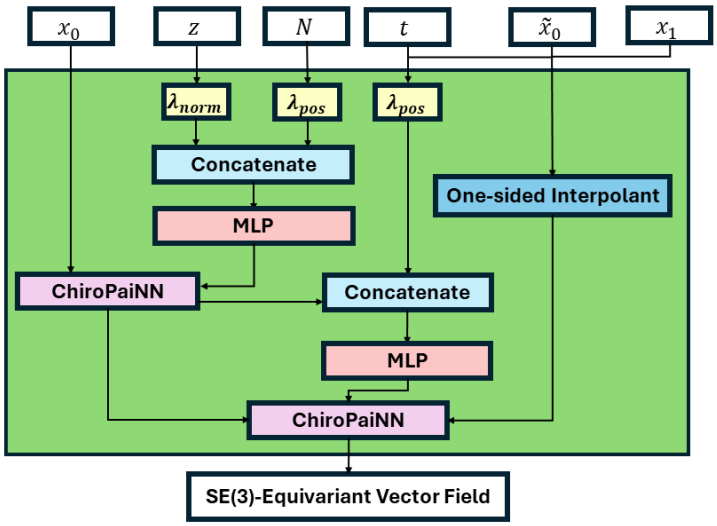


Figure 3.6: An overview of the SE(3)-equivariant vector field used for Stochastic Interpolants ITO model.

3.2.3 Model Evaluation

To evaluate the ITO model, the core task is to compare distributions $\rho(\mathbf{x}_{N\tau} \mid \mathbf{x}_0)$ between the ITO model and the ground truth. For simple systems like the Müller Brown potential, we are able to obtain the ground truth through Langevin simulation, while for Alanine Dipeptide, we use MSM on MD trajectories to estimate the ground truth. Additionally, the VAMP-2 score described below is also used as a benchmark.

3.2.3.1 VAMP Scores

Variational Approach for Markov Processes (VAMP) [18], is a recent result which allows us to find optimal feature mappings and approximate Markovian models based on the dynamics from time series data. VAMP models project data into low dimensional representations to analyse and compare the dynamical behaviour. One main contribution of this model family is the VAMP-scores. These scores are developed based on the understanding that the most effective linear model, characterized by minimal prediction error, can be described using the principal singular components of the Koopman operator $\mathcal{K}_\tau$ as defined in Equation (2.14). Calculated as the sum of the singular values of $\mathcal{K}_\tau$ multiplied by the overlap coefficients between a set of ortho-normalized feature-maps f and g. These scores can thus be used to compare the dynamics between models. VAMP $-$ r score is computed via

$$\text{VAMP} - r = \sum_{i=0}^{k} \sigma_i^r \tag{3.4}$$

where $r \in \mathbb{N}_+$.

The VAMP-r scores describe the meta-stability of a Koopman matrix. For comparison purposes, we define the VAMP-gap ΔV between two Koopman matrices, K and K', as the difference between their VAMP-2 scores:

$$\Delta V = \text{VAMP} - 2(K) - \text{VAMP} - 2\,(K') \tag{3.5}$$

where K' is the reference and K is the query matrix, respectively. In our case, we compare the ITO model with Langevin simulation data. A perfect match is indicated by 0. Negative values correspond to underestimation, and positive values correspond to overestimation of meta-stability.

3.2.4 Experiment Setups

Below, we outline the detailed model setup for each system. For comparison purposes, the setup follows DDPM ITO settings.

3.2.4.1 Low-Dimensional System (MB)

As a fully observed case, it is reasonable to start from the MB dataset and test its performance and determine the architecture for larger systems.

To learn and sample from the conditional distribution of the future system state given its current state, it's natural to specify a time lag both during training and inference. We consider two types of lags: stochastic lag and fixed lag.

When using a stochastic lag, the time lag is sampled stochastically within a specified range during the training process. Consequently, data pairs ($\mathbf{x}_0$, $\mathbf{x}_t$) are determined according to different time lags. During inference, various time lags within the training range can be utilized.

In fixed lag, only one value is employed for the time lag during training. Thus, during inference, predictions can only be made for that specific time lag.

We conduct experiments on both cases and compare the probability distributions as well as the VAMP-2 Score. For the stochastic case, a single model is trained with stochastic lag $N \sim \text{DisExp}(1000)$, while for the fixed lag case, a set of models are trained with fixed constant lags $N = \{10, 100, 1000\}$. For detailed setup parameters, refer to Table A.4 in appendix.

3.2.4.2 Molecular Systems (ALA2)

Since we only utilize the 3D positions of 22 atoms of the solute for training, neglecting the consideration of velocities, the dataset represents a partially observed case.

Similar to the MB case, we conduct experiments with two types of time lags. To align with the setup of DDPM ITO, we sample trajectories with 4, 64, or 512 steps for 15,000 trajectories. MD simulations for training data are executed with 2fs integration time-steps, and data is stored at 1ps intervals. Consequently, we aim to compare the transition densities $\rho\left(\mathbf{x}_{\Delta t=4,\,64,\,512\,\text{ps}} \mid \mathbf{x}_0\right)$. It's important to note that the ground truth for comparing transition probability densities needs to be estimated from MSMs, as we have only a limited number of trajectories available from MD. To compare the conditional probability density, we compute the dihedral angles ϕ, ψ and plot the marginal distribution, comparing it with the MSM results. For detailed setup parameters, refer to Table A.5 in appendix.

3.3 Thermodynamic Interpolants

Many proposed methods have tackled the issue of estimating free energy differences between thermodynamic states located at different temperatures. Targeted free energy perturbation, described in Section 2.1.4.1, allow us to calculate free energy differences between two thermodynamic states, given an invertible mapping between them and samples drawn from the canonical distribution of the initial state [15]. However, there is no general approach to obtaining such a map. An appealing strategy is to learn a continuous normalizing flow to map samples between the distributions. In that case, one could also obtain the change in probability that is required for TFEP to work.

Another proposed solution to the same problem is through generative models. Learned Replica Exchange (LREX) [33] and Temperature Steerable Flows combine replica

exchange with normalizing flows [34]. The main limitations of these methods are due to the limitations of normalizing flows, such as lower training efficiency, making it difficult to generalize to larger systems. The Thermodynamic Map employs a diffusion model to learn the joint distribution $\rho(\mathbf{x}, \beta)$ of microscopic states $\mathbf{x}$ and inverse temperatures β [35]. However, the principle of this method differs from TFEP in that the map will not be invertible, and no probabilities will be obtained. This is not ideal because it complicates the estimation of free energies.

To make transformations across temperature, we employed the use of *Thermodynamic Interpolants*. While thermodynamic interpolation can be performed in multiple ways, all our methods share the feature that they transport samples, drawn from a Boltzmann distribution of a given energy, between different temperatures in a manner that is based upon the Stochastic Interpolant framework described in Section 2.3.3. Still it is worth to note that none of the following approaches have any guarantees in terms of generalization power outside of the model's training data.

3.3.1 Interpolation Methods

To develop the methods presented in this thesis, we also experimented with many other ways of performing transformations. Below, we describe the two most successful thermodynamic interpolation methods.

3.3.1.1 Augmented Two-Sided Linear Interpolant

The first thermodynamic interpolant we used was the augmented two-sided linear interpolant. As described in Section 2.3.3.1, a linear interpolant can be obtained by setting

$$\alpha(t) = 1 - t, \quad \beta(t) = t, \quad \gamma(t) = \sqrt{t(1-t)},$$

so that for $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$, $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$ an interpolant (2.23) is defined through

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1 + \sqrt{t(1-t)}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathrm{Id}). \tag{3.6}$$

This choice of α, β and γ results in a simple transformation between any two densities. While this is a solid and well-performing approach when training on data from both densities, there are no indications that this type of model will be able to interpolate or extrapolate to previously unseen temperatures, rendering this method rather data-inefficient.

To develop a more data-efficient model, that can interpolate or extrapolate to temperatures it has not seen during training, we employed the use of a two-sided linear interpolant. By defining a joint distribution, $\rho_t(\mathbf{x}, \beta_0, \beta_1)$, so that we also included temperature arguments in the model, we hoped to facilitate sample transformations from any β_0 to any β_1 we would like. The question then arises of how to learn such a model.

A quite straightforward strategy is to train on data from a few different temperatures, hoping that the model picks up on the temperature-dependency of the data. This

is the idea behind the augmented two-sided linear interpolant. We trained on data corresponding to the two sets $\mathcal{B}_0 = \{\hat{\beta}_0, \ldots, \hat{\beta}_N\}$ and $\mathcal{B}_1 = \{\tilde{\beta}_0, \ldots, \tilde{\beta}_M\}$, where M, $N \in \mathbb{N}$, optimizing against the original Stochastic Interpolant objective (2.26). In this way, the model would encounter all combinations of $\beta_0 \in \mathcal{B}_0$ and $\beta_1 \in \mathcal{B}_1$ during training. Similarly to the original version, the augmented two sided linear interpolant is also defined through Equation (3.6).

3.3.1.2 Scaled Mirror Interpolant

Another strategy for building a model that extrapolates to previously unseen temperatures is to exploit the mirror interpolant defined in Equation (2.30). Here, we could use that the velocity field is proportional to the score field defined in Equation (2.25) for this choice of interpolant.

The interpolated density $\rho_t(\mathbf{x})$ should ideally follow a Boltzmann distribution at all times $t \in [0, 1]$, e.g. it should be on the form

$$\rho_t(\mathbf{x}) = \frac{1}{Z_t} e^{-\beta_t E(\mathbf{x})}, \quad t \in [0, 1].$$

In that case, the score $\mathbf{s}$ can be written as

$$\mathbf{s}(t, \mathbf{x}) = \nabla_{\mathbf{x}} \rho_t(\mathbf{x}) = -\beta_t \nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_t, \tag{3.7}$$

The first thermodynamic interpolant used is the augmented two-sided linear inter-

for the setup above. Thus, if a score field model $\mathbf{s}_0$ is trained on data $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$, generated at a temperature β_0, it can be approximately transferred to another close-by temperature, β_1, through a simple scaling

$$\mathbf{s}_1(t, \mathbf{x}) \approx \left((1 - t) + \frac{\beta_1}{\beta_0} t \right) \mathbf{s}_0(t, \mathbf{x}), \quad t \in [0, 1]. \tag{3.8}$$

Noting that Equation (2.29) allows us to write the velocity field in terms of the score field, samples $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$ can be approximately generated by integrating Equation (2.20) for a scaled score field, given by Equation (3.8), from initial conditions $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$.

Now, it is important to note a significant limitation to this method. While the setup should work well in theory, when ρ_t is a Boltzmann distribution, that is not at all a guarantee. Neither is it a guarantee that a linear interpolation in β corresponds to how the neural network has learned the temperature-dependency. Thus, this method runs the risk of introducing unnecessary errors.

3.3.2 Neural Network Model Architectures

While the thermodynamic interpolation methods served as part of the model architectures, the neural network design would also impact the results. For low-dimensional systems, the learning task is quite simple and there is no need for

complicated architectures. However, as outlined in Section 2.6, for larger molecular systems it is important to incorporate geometrical knowledge about the data into the neural network design. With this in mind, we decided to use different architectures depending on the dimension of the samples in the training data.

3.3.2.1 Low-Dimensional System (ADW)

For the one-dimensional asymmetric double well toy potential (3.1) we used a simple neural network consisting of two multi-layer perceptrons. As input, the network took the one-dimensional interpolated coordinates x_t, the interpolation time t, the initial inverse temperature β_0 and the target temperature β_1. The vector field was then taken directly as the model output. Figure 3.7 shows an overview of the architecture.

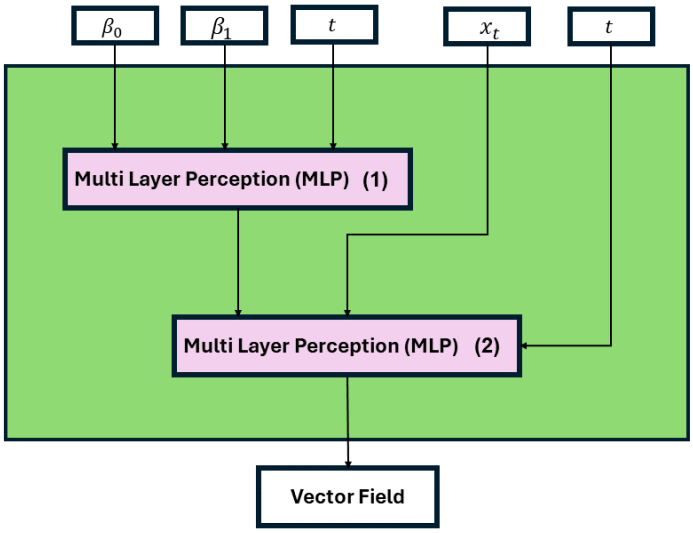


Figure 3.7: An overview of the vector field used for modelling the stochastic interpolant score and velocity field when running on low-dimensional systems.

3.3.2.2 Molecular Systems (MDQM9-nc)

When working with higher-dimensional molecular systems, we used the ChiroPaiNN model described in Section 2.6.2 to build an SE(3)-equivariant vector field. The input to ChiroPaiNN was given by the interpolated atom coordinates $\mathbf{x}_t$, the interpolation time t, the order of the atoms z, the initial temperature T_0 and the target temperature T_1. The time and the temperatures were embedded through positional embeddings, denoted λ_{pos}, similar to those used in the original ChiroPaiNN architecture [7]. The atom ordering was embedded in a standard learnable embedding, denoted by λ_{nom}.

To construct an SE(3)-equivariant output vector field, a final readout layer was added to the original ChiroPaiNN structure. For the readout, we kept in mind that the readout layer needs to combine the equivariant and invariant features in a way

that preserves the equivariance in the output. An operation fulfilling this criteria is a simple scaling of the equivariant features. Since the invariant features are invariant scalar values, they can be used to scale the equivariant features. Using the notation from Section 2.6.2, the output from the readout layer can be written as

$$\mathbf{u}_\theta(t, \mathbf{x}) = \langle \mathbf{v}_i(t, \mathbf{x}), \mathbf{s}_i(t, \mathbf{x}) \rangle,$$

where $\mathbf{u}_\theta$ denotes a general vector field of equal dimension to the velocity field $\mathbf{b}_\theta(t, \mathbf{x})$. Figure 3.8 shows the SE(3)-equivariant vector field used to model the score and velocity fields.

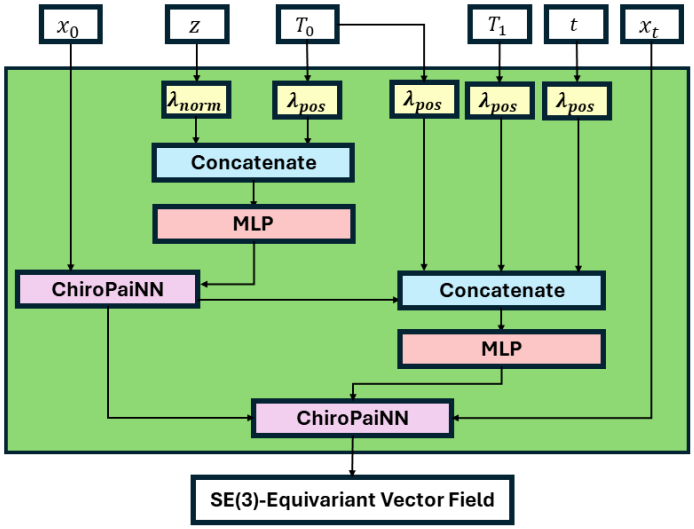


Figure 3.8: An overview of the SE(3)-equivariant vector field used for modelling the Stochastic Interpolant score and velocity field when running on molecular systems.

3.3.3 Model Evaluation

Depending on the dimension of the model system, we evaluate the surrogate models differently. In this part, we describe how the Thermodynamic Interpolant models were evaluated and motivate the reason for those choices of evaluation criteria.

3.3.3.1 Effective Sample Size

To assess the efficiency of the generative models, we calculated the ESS for a transformation between the two states S_0 and S_1 by measuring the design effect [36]. For n samples generated by a neural network model, the ESS can then be calculated as

$$\text{ESS} = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2},$$

where for a one-dimensional system the weights w_i can be taken to be the importance weights from Equation (2.18). Note that for a Boltzmann distribution, the unnormalized distribution $\hat{\rho}$ is given as $\hat{\rho}(\mathbf{x}) = \exp(-\beta E(\mathbf{x}))$ so that the importance weights w_i take the form

$$w(\mathbf{x}) = \frac{\exp\left(-\beta E(\mathbf{x})\right)}{\rho_\theta(\mathbf{x})}. \tag{3.9}$$

In lower dimensions, the intractability of the partition function Z does not have as big of an impact, since we can either compute it explicitly or using a numerical integration scheme. For higher-dimensional systems though, the partition function is intractable, and since the estimated sample probabilities ρ_θ in this case depend on the partition function they can not be computed. Here, we instead use the same definition of the weights as used in the Boltzmann generator [4] where

$$\hat{w}(\mathbf{x}_0) \propto w(\mathbf{x}_0) = \exp\left(E_0(\mathbf{x}_0) - E_1(\mathcal{M}(\mathbf{x}_0) + \log\det|J(\mathbf{x}_0)|\right), \tag{3.10}$$

for an initial sample $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ and where $\mathcal{M}(\mathbf{x}_0) = \mathbf{x}_1 \sim \rho_1(\mathbf{x})$ with J being the Jacobian of $\mathcal{M}$. There is a similarity in these weights to the function ϕ in Equation (2.5) from the targeted free energy perturbation method. Through comparison of the two functions, we note that

$$w(\mathbf{x}_0) = \exp(-\phi(\mathbf{x}_0)). \tag{3.11}$$

3.3.3.2 Internal Coordinates and Z-matrix

The MDQM9 models were also evaluated with respect to their *internal coordinate representations*. The internal coordinates consist of bond lengths, bond angles and torsion (dihedral) angles, uniquely defining the set of Cartesian coordinates of 3-dimensional positions of the molecule's atoms. The internal coordinates can be represented in matrix form using a *Z-matrix*. Below is an algorithm for computing the Z-matrix given a molecule representation in Cartesian coordinates.

1. Compute the bond length as the distance d_{12} from the first to the second atom using
$$d_{ij} = \|\mathbf{r}_{ij}\|, \tag{3.12}$$
with $i = 1$ and $j = 2$.

2. Then calculate the distance d_{23}, again using Equation (3.12) but with $i = 2$ and $j = 3$. Then compute the bond angle θ_{ijk} that is described by the three atoms through
$$\theta_{ijk} = \arccos\left(\frac{\mathbf{r}_{ji} \cdot \mathbf{r}_{jk}}{\|\mathbf{r}_{ji}\|\|\mathbf{r}_{jk}\|}\right), \tag{3.13}$$
with $i = 1$, $j = 2$ and $k = 3$.

3. Finally, use previously added atoms as reference for calculating one bond length, one bond angle and one torsion angle for each of the remaining atoms.

Here, the torsions ϕ_{ijkl} are defined through

$$\begin{aligned} \cos(\phi_{ijkl}) &= \frac{\mathbf{r}_{ij} \times \mathbf{r}_{jk}}{\|\mathbf{r}_{ij}\|\|\mathbf{r}_{jk}\|} \times \frac{\mathbf{r}_{jk} \times \mathbf{r}_{kl}}{\|\mathbf{r}_{jk}\|\|\mathbf{r}_{kl}\|}, \\ \sin(\phi_{ijkl}) &= \frac{\mathbf{r}_{ij} \times \mathbf{r}_{jk}}{\|\mathbf{r}_{ij}\|\|\mathbf{r}_{jk}\|} \times \frac{\mathbf{r}_{jk} \times \mathbf{r}_{kl}}{\|\mathbf{r}_{jk}\|\|\mathbf{r}_{kl}\|} \cdot \frac{\mathbf{r}_{jk}}{\|\mathbf{r}_{jk}\|}, \end{aligned} \tag{3.14}$$

where the indexes i, j, k and l correspond to the current atom and three reference atoms.

Following the algorithm above will result in a matrix similar to

$$Z = \begin{bmatrix} d_{12} & 0 & 0 \\ d_{23} & \theta_{123} & 0 \\ d_{34} & \theta_{234} & \phi_{1234} \\ \vdots & \vdots & \vdots \end{bmatrix}, \tag{3.15}$$

where we note that the reference atoms can be picked in any manner. This implies that there are many Z-matrices for each set of Cartesian coordinates. Therefore, the Z-matrix (3.15) is not the only possible choice since different reference atoms can be used when computing the torsion angles.

3.3.4 Experiment Setups

Below, we describe the experiments that were performed for each model system in more detail.

3.3.4.1 Low-Dimensional System (ADW)

To begin with, we used a one-dimensional asymmetric double well as model system since it is simple enough to allow for analytical treatment and extensive sampling yet still exhibiting slow mixing, the property we seek to overcome. We trained the two-sided linear interpolant model on data corresponding to a set of inverse temperatures β_{train}. Then, we used it to transform from a given β_0 to another β_1. We wanted to test both interpolation and extrapolation, and so we performed two types of experiments, one for each of the two cases.

To investigate the model's interpolation capabilities, we trained on data taken from across the full range of β-values. Then we made transformations from the lowest (or highest) training temperature into an intermediate temperature, i.e. a temperature that the model has not seen during training but that lie within the range of training temperatures that the model has seen. To test differences in transforming from low to high β or high to low β, we also made sure to perform transformations in both of these "directions". With this in mind, two specific experiments were performed to test interpolation capacity.

1. Firstly, the augmented two-sided linear interpolant was trained on $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$. Then, we transformed data from $\beta_0 = 0.5$ into each $\beta_1 \in \{0.75, 1.00, 1.50, 1.75\}$. For each transformation the sampling efficiency was evaluated by measuring the ESS.

2. Secondly, the augmented two-sided linear interpolant was trained on $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$. Then data was transformed from $\beta_0 = 2.0$ into each $\beta_1 \in \{0.75, 1.00, 1.50, 1.75\}$. Again, the sampling efficiency of each transformation was evaluated by measuring the ESS.

For every transformation above, an equivalent transformation was made with the scaled mirror interpolant for comparison. We also compared both the mirror interpolant and the two-sided linear interpolant to direct re-weighting.

Next, we tested the model's extrapolation capabilities. Here, the model was trained on a smaller range of β-values. Then, a β_0 was selected from the training set and transformed into a β_1 that was outside of the training set. Again, two sets of experiments were performed.

1. Firstly, the augmented two-sided linear interpolant model was trained on $\beta_{\text{train}} \in \{1.00, 1.25, 2.00\}$. Then transformations were made from $\beta_0 = 1.00$ into each $\beta_1 \in \{0.50, 0.75, 1.75, 2.00\}$. This choice of β_0 corresponds to a transformation from the lower end of the set of β_{train}.

2. Secondly, we trained the augmented two-sided linear interpolant on $\beta_{\text{train}} \in \{1.00, 1.25, 2.00\}$, and then transformed data from $\beta_0 = 1.50$ into each $\beta_1 \in \{0.50, 0.75, 1.75, 2.00\}$. This choice of β_0 corresponds to a transformation from the upper end of the set of β_{train}.

For each transformation above, an equivalent transformation was made with the scaled mirror interpolant for comparison. Just as previously, both these models were compared to direct re-weighting.

3.3.4.2 Molecular Systems (MDQM9-nc)

When extending the model to molecular data we performed a number of experiments, similar to those performed on the asymmetric double well potential, but not as extensive. While we for the asymmetric double well system controlled the temperature dependency with the inverse temperature β, it is simpler to work with the temperature T for molecular systems. Because of this, T is used equivalently here to how β was previously used in Subsection 3.3.4.1.

Again, we tested both extrapolation and interpolation capabilities. Each model was trained on all temperatures but one, leaving out a different temperature each time. For example, the first model was trained on data corresponding to temperatures $T_{\text{train}} \in \{400\text{K}, 500\text{K}, 600\text{K}, 700\text{K}, 800\text{K}, 900\text{K}, 1000\text{K}\}$, leaving out data at 300K, while the second model was trained on data corresponding to temperatures $T_{\text{train}} \in \{300\text{K}, 500\text{K}, 600\text{K}, 700\text{K}, 800\text{K}, 900\text{K}, 1000\text{K}\}$ where data at 400K was left out. This process was repeated for all temperatures, so that eight different models were trained in total. In this optimized setting, we hoped to make the most of the available data.

As with the ADW system, we tested both interpolation and extrapolation. To test interpolation capabilities, we transformed from higher $T_0 = 800\text{K}$ into lower $T_1 \in \{700\text{K}, 600\text{K}, 500\text{K}, 400\text{K}\}$, where the models were trained on all temperatures

but the current value of T_1. To test extrapolation capabilities, we transformed from higher $T_0 \in \{700\text{K}, 600\text{K}, 500\text{K}, 400\text{K}\}$ into a lower $T_1 = 300$, where the model was trained on all available temperatures but T_1. For each transformation we computed the effective sample size to measure the sampling efficiency of the thermodynamic interpolant.

3.3.4.3 A Note on Outliers

Outliers in, for example, a dataset are values that lie very far from the other values. The issue with an outlier is that it can impact sample-estimated quantities significantly. If one outlier sample contributes much more to the average than the remaining samples, the effect of the non-outlier samples can get "drowned out". When evaluating the weights in Equations (3.9) and (3.10), using the obtained samples and log probabilities, we noted that there seem to exist outliers in the results. A more thorough discussion of this matter can be found in Appendix A.3.

To select a proper approach for dealing with the issue, we investigated the impact of filtering of the weights on the molecular system. Weights with values very far from the others were filtered out. Then we compared the weighted marginal histograms of torsions, bond angles and bond lengths for the case with filtered weights to the corresponding non-weighted marginals. We decided that if an improvement could be seen when using filtered weights, then our weights would still carry some form of relevance and it would be acceptable to use the filtered weights to calculate for example effective sample sizes. Additionally, in Appendix A.3 we present a comparison between marginals weighted with unfiltered and filtered weights for further verification of the approach.

4

Results

In the following chapter, we present the results obtained with the Stochastic Interpolant ITO models and with the Thermodynamic Interpolants. For each of the two methods, we trained on one lower-dimensional system and on one higher dimensional system. To clearly distinguish between the results, we have split this chapter into a separate section for each of the two methods.

4.1 Stochastic Interpolants ITO Model

The following section is dedicated to the Stochastic Interpolant ITO model. In the two upcoming subsections, we report results for this architecture, when trained on the low-dimensional MB system and the higher-dimensional Alanine Dipeptide molecular system.

4.1.1 Müller Brown

First, we present results from the MB system. We show plots of transition probability densities between the ITO model and the ground truth from MD simulations, $\rho(\mathbf{x}_{N\tau} \mid \mathbf{x}_0)$, in Figure 4.1. One can see generally the results from ITO model align well with simulation results across various time lags.

We also compute the VAMP-2 score gap to compare dynamical properties. The results are shown in Table 4.1. We obtained a relatively lower absolute value for the VAMP-2 score gap compared to the original DDPM model, indicating better dynamical properties.

One notable point is that with the Stochastic Interpolant model, we get positive values for all the VAMP-2 score gap, which suggests an overestimation of metastability. This contrasts with the results from the DDPM model, where the conclusion is underestimation of metastability. Additionally, for our model, the fixed lag shows a low absolute VAMP-2 score gap, while the DDPM model performs better with a stochastic lag.

Since the Stochastic Interpolant is an unbiased generative model, whereas the diffusion model has bias due to truncated integration time, we anticipate that our model would produce better results. However, it is difficult to conclude which type of lag is

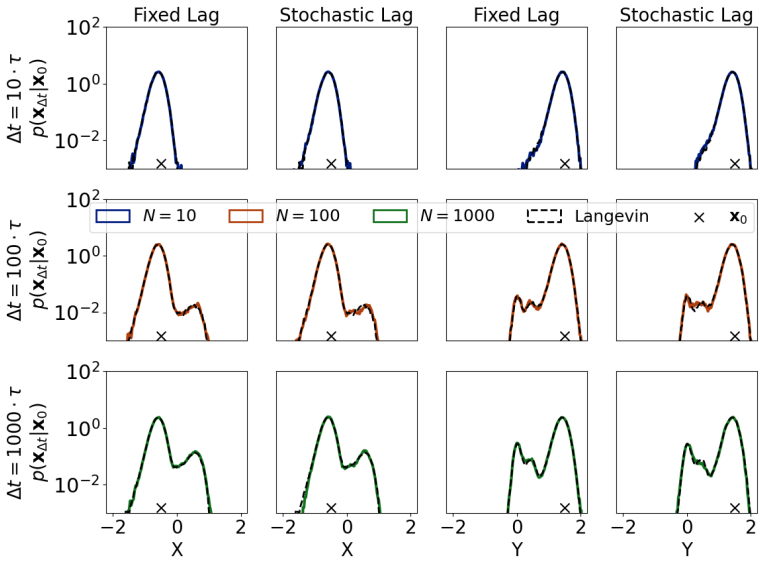


Figure 4.1: Histogram Plots illustrating log-scale transition probability densities across various time lags N, comparing ITO models with Langevin simulations. $\mathbf{x}_0$ is indicated by a cross, representing the initial position. Both the x and y dimensions of the MB system are shown. ITO models trained with stochastic and fixed lags are included.

superior. As shown in the ITO paper, there is no clear preference for the lag option in molecular systems according to the VAMP-2 score.

Table 4.1: Vamp-2 score gap values.
Time lags are chosen to be $N = 10, 100$ to align with the DDPM ITO paper. DDPM ITO results are also shown here for comparison purposes. Both fixed and stochatics lag results are presented.

ITO Model	Stoch. Interp.	DDPM	Stoch. Interp.	DDPM
Lag Value	10	10	100	100
VAMP (fixed)	0.0283	−0.0351	0.0699	−0.1189
VAMP (stochastic)	0.0297	−0.0312	0.0767	−0.0970

4.1.2 Alanine Dipeptide

Transition probability densities of alanine dipeptide dynamics with SE3-ITO model are shown in Figure 4.2. Corresponding transition densities computed from molecular dynamics simulations are presented as ground truth for comparison.

In the plot, we compare the marginal distribution from computed dihedral angles. As we can see, it aligns well with the estimated MSM results. We chose a different initial position than the one used in the DDPM ITO results, yet we still obtained good results compared to the simulation data. For the ALA2 system, as a partially observed case, we demonstrate our model's robustness to larger systems.

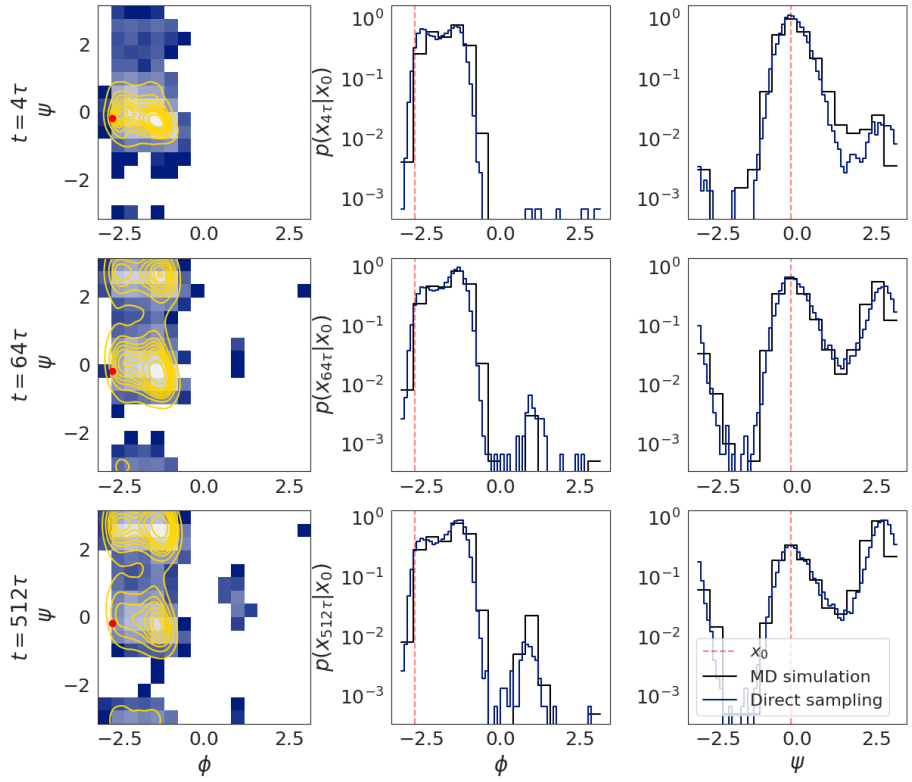


Figure 4.2: Histogram Plots illustrating transition probability densities across various time lags N, comparing ITO model with estimated MSM results. Initial torsion angles are indicated by red dot. Both the torsion angles ϕ, ψ of the ALA2 system are shown. ITO model is trained with stochastic lag in this case.

4.2 Thermodynamic Interpolants

This next section is dedicated to the Thermodynamic Interpolants. As outlined in Subsection 3.3.4, we investigated the performance of the Thermodynamic Interpolants when trained on both the low-dimensional ADW system and the higher-dimensional MDQM9-nc molecular systems. In the following two subsections, we report the results that we obtained from those experiments.

4.2.1 Asymmetric Double Well

First, we present results on the asymmetric double well potential. Table 4.2 shows the exact effective sample size percentages for the first experiment, e.g. where $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$, $\beta_0 = 0.50$ and $\beta_1 \in \{0.75, 1.00, 1.50, 1.75\}$. Here, we tested the interpolation capabilities of the model when transforming from a low β_0 to a higher β_1.

Figure 4.3 shows the same ESS values as in Table 4.3 in a plot against β_1.

Table 4.2: Percentages of effective number of samples for an augmented two-sided linear interpolant trained on $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$ compared to a scaled mirror interpolant baseline and direct re-weighting. Transformations are from a lower β to a higher β and all reported transformations are interpolations. In order to avoid impact of outliers, these have been filtered using the interquartile range method (IQR). More details can be found in Appendix A.3.

β_0	β_1	ESS Aug. lin. (%)	ESS Mirror (%)	ESS RW (%)
0.50	0.75	97.3087	92.3910	92.4146
0.50	1.00	96.0209	88.5511	80.0153
0.50	1.50	94.5692	73.2183	59.7757
0.50	1.75	95.1663	67.4071	52.6152

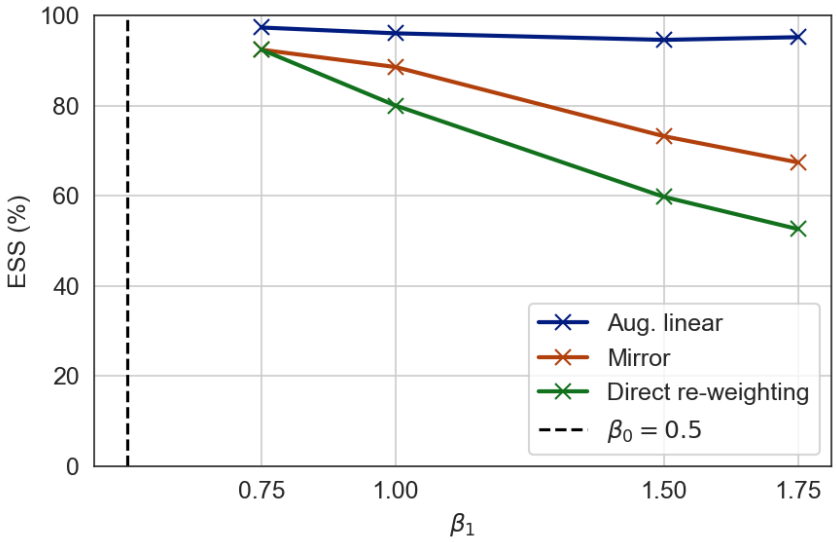


Figure 4.3: A plot of the values from Table 4.2. Here, the augmented two-sided linear interpolant was trained on data corresponding to $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$. The x-axis shows the value of β_1 and the y-axis shows the value of the effective samples size (ESS) measured in percentage. In the figure, the value of β_0 is marked with a dashed black line. To avoid the impact of outliers, these have been filtered using the interquartile range (IQR) method. More details can be found in Appendix A.3.

As can be seen in both Table 4.2 and Figure 4.3, the mirror interpolant performs better than direct re-weighting in all scenarios. Still, the augmented two-sided linear interpolant outperforms both the mirror interpolant and direct re-weighting across all temperature transformations. We note that while the sampling efficiencies of the other two methods decay with the length of the temperature transformation, the augmented two-sided linear interpolant is not as fatally affected by the transformation length.

Next, we present results for the model's interpolation capabilities when transforming from a high to a low β_0. Table 4.3 shows the exact effective sample size percentages

for the second experiment, e.g. where $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$, $\beta_0 = 2.0$ and $\beta_1 \in \{0.75, 1.00, 1.50, 1.75\}$.

Table 4.3: Percentages of effective number of samples for an augmented two-sided linear interpolant trained on $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$ compared to a scaled mirror interpolant baseline. Transformations are from a higher β to a lower β and all reported transformations are interpolations. To avoid the impact of outliers, these have been filtered using the interquartile range (IQR) method. More details can be found in Appendix A.3.

β_0	β_1	ESS Aug. lin. (%)	ESS Mirror (%)	ESS RW (%)
2.00	0.75	84.6233	49.6543	61.3789
2.00	1.00	87.6705	55.7321	71.0267
2.00	1.50	96.9007	66.7864	90.9686
2.00	1.75	98.1648	69.7357	97.6236

Figure 4.4 shows the same ESS values as in Table 4.4 in a plot against β_1.

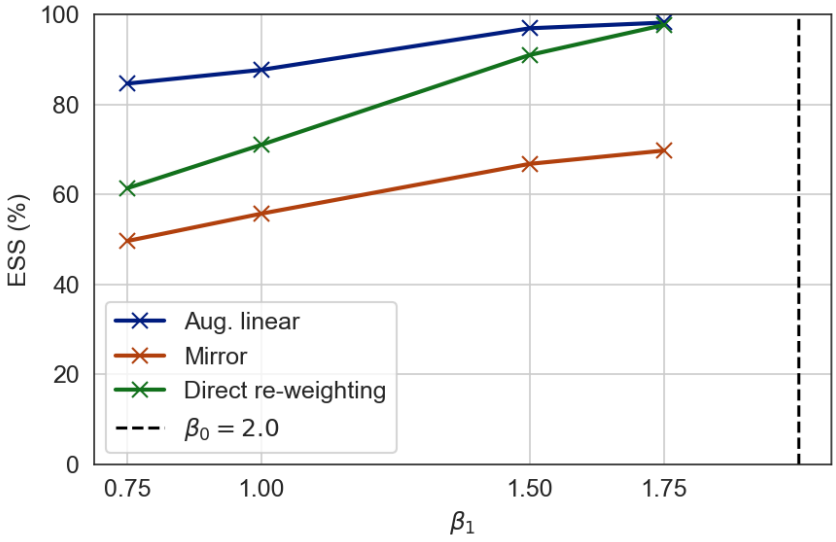


Figure 4.4: A plot of the values from Table 4.3. Here, the augmented two-sided linear interpolant was trained on data corresponding to $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$. The x-axis shows the value of β_1 and the y-axis shows the value of the effective samples size (ESS) measured in percentage. In the figure, the value of β_0 has been marked out with a dashed black line. In order to avoid impact of outliers, these have been filtered using the interquartile range method (IQR). More details can be found in Appendix A.3.

As can be seen in both Table 4.3 and Figure 4.4, the mirror interpolant performs significantly worse than direct re-weighting in all scenarios. The augmented two-sided linear interpolant outperforms both the mirror interpolant and direct re-weighting across all temperature transformations. Still, we note that it seems to struggle more when transforming from a high β_0 into a low β_1 compared to the reverse.

After this we present the results for the two experiments on the model's extrapolative capabilities. We start by showing results for the third experiment, where we test transformation from a β_0 that lies at the lower end of the range of β_{train}. Table 4.4 shows the exact effective sample size percentages for this experiment, i.e. where $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$, $\beta_0 = 1.00$ and $\beta_1 \in \{0.50, 0.75, 1.75, 2.00\}$.

Table 4.4: Percentages of effective number of samples for an augmented two-sided linear interpolant trained on $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$ compared to a scaled mirror interpolant baseline. All reported transformations are extrapolations. In order to avoid impact of outliers, these have been filtered using the interquartile range method (IQR). More details can be found in Appendix A.3.

β_0	β_1	ESS Aug. lin. (%)	ESS Mirror (%)	ESS RW (%)
1.00	0.50	82.7412	62.2818	78.5553
1.00	0.75	94.2216	67.2811	92.9339
1.00	1.75	96.3144	78.2762	81.6740
1.00	2.00	90.1935	76.5954	74.8553

Figure 4.5 shows the same ESS values as in Table 4.5 in a plot against β_1.

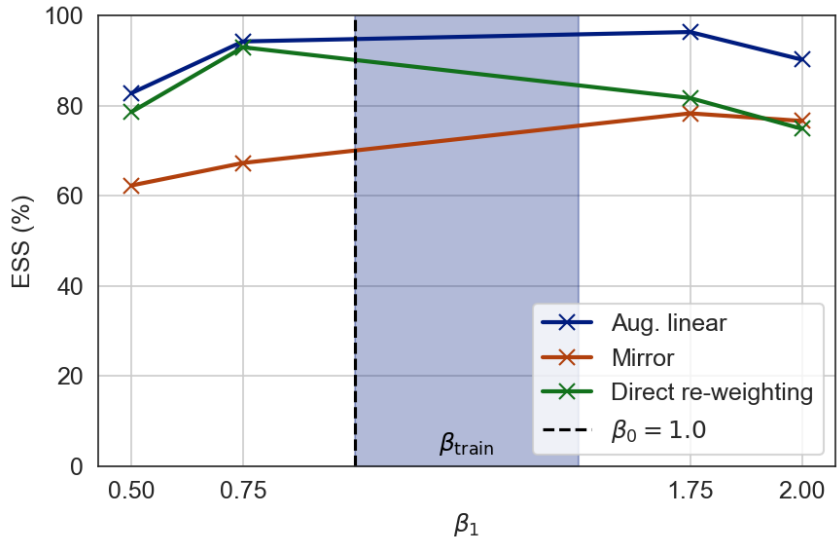


Figure 4.5: A plot of the values from Table 4.4. Here, the augmented two-sided linear interpolant was trained on data corresponding to $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$. The x-axis shows the value of β_1 and the y-axis shows the value of the effective samples size (ESS) measured in percentage. In the figure, the value of β_0 has been marked out with a dashed black line and the range of β_{train} is marked out in a light blue color. To avoid the impact of outliers, these have been filtered using the interquartile range (IQR) method. More details can be found in Appendix A.3.

From both Table 4.4 and Figure 4.5 it can be seen that the mirror interpolant performs significantly worse than direct re-weighting in a majority of the cases. In the one case that it outperforms direct re-weighting, it is only a very slight improvement.

The augmented two-sided linear interpolant again outperforms both the mirror interpolant and direct re-weighting across all temperature transformations. It also seems to be quite efficient in extrapolation, especially when β_1 lies close to the training range of β-values.

Finally, we show results for the fourth experiment where we test transformation from a β_0 that lies at the upper end of the range of β_{train}. Table 4.5 shows the exact effective sample size percentages for the this experiment, where $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$, $\beta_0 = 1.50$ and $\beta_1 \in \{0.50, 0.75, 1.75, 2.00\}$.

Table 4.5: Percentages of effective number of samples for an augmented two-sided linear interpolant trained on $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$ compared to a scaled mirror interpolant baseline. All reported transformations are extrapolations. In order to avoid impact of outliers, these have been filtered using the interquartile range method (IQR). More details can be found in Appendix A.3.

β_0	β_1	ESS Aug. lin. (%)	ESS Mirror (%)	ESS RW (%)
1.50	0.50	81.4382	54.5221	61.8995
1.50	0.75	92.1830	59.5279	73.8632
1.50	1.75	97.5086	70.7608	97.7652
1.50	2.00	92.5584	71.3206	93.2048

Figure 4.6 shows the same ESS values as in Table 4.6 in a plot against β_1.

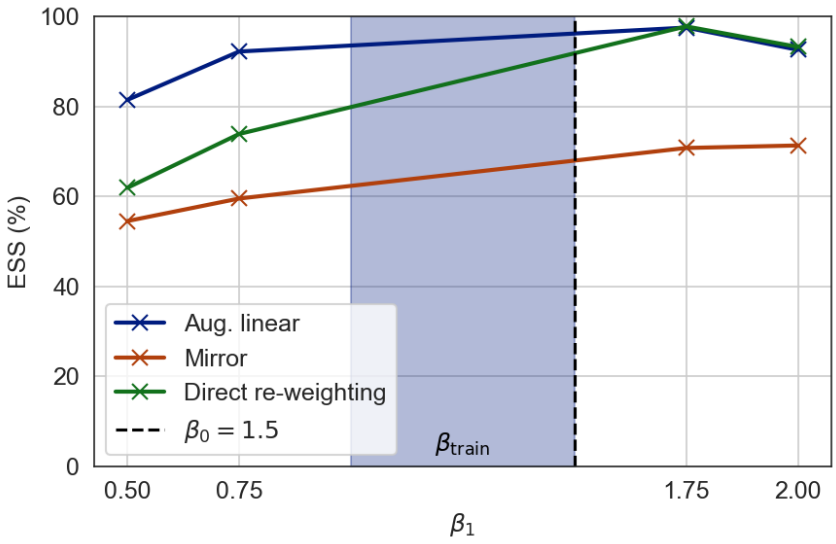


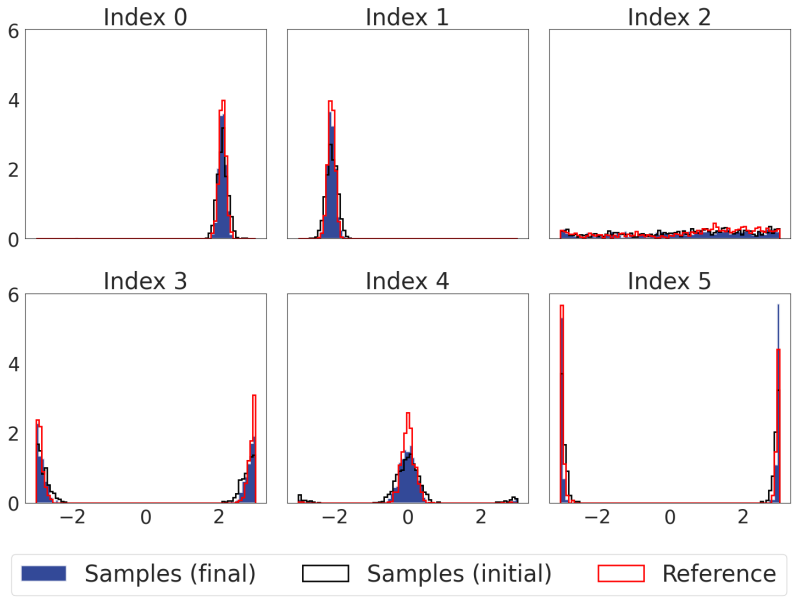
Figure 4.6: A plot of the values from Table 4.5. Here, the augmented two-sided linear interpolant was trained on data corresponding to $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$. The x-axis shows the value of β_1 and the y-axis shows the value of the effective samples size (ESS) measured in percentage. In the figure, the value of β_0 has been marked out with a dashed black line and the range of β_{train} is marked out in a light blue color. To avoid the impact of outliers, these have been filtered using the interquartile range (IQR) method. More details can be found in Appendix A.3.

Table 4.5 and Figure 4.6 again shows the mirror interpolant performing significantly worse than direct re-weighting in all transformations. The augmented two-sided linear interpolant again outperforms both the mirror interpolant across all temperature transformations. Compared to direct re-weighting, it performs better in many cases and where it doesn't it at least has a similar performance. Again, it can be seen how the mirror interpolant and direct re-weighting methods are significantly affected by the length of the transformation. While this also holds true for the augmented two-sided linear interpolant, the effect does not seem to be as large as for the other two methods.

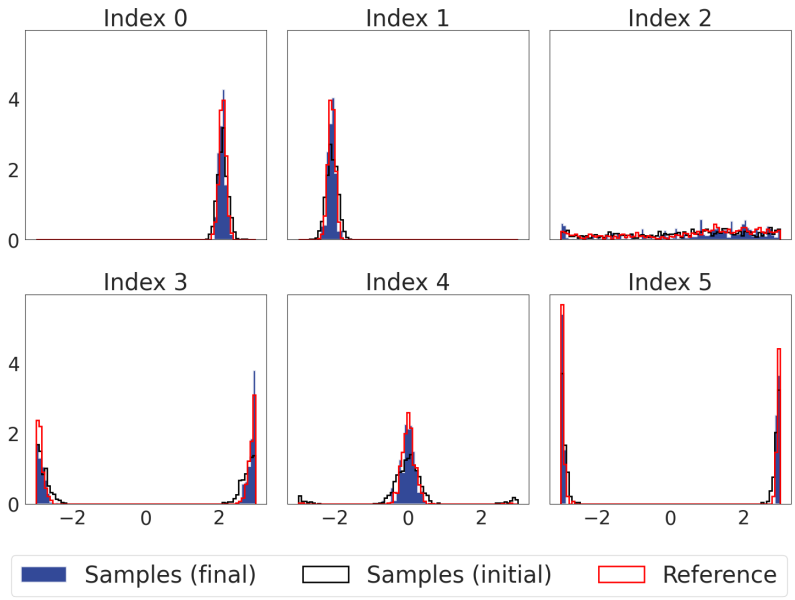
4.2.2 Replica Exchange QM9

Next, we report the results on the MDQM9-nc dataset. To investigate the effect of IQR outlier filtering on the results, we first plot the marginal histograms of the different degrees of freedom in the molecule. To reduce redundancy in the results we only report the best set of marginal histograms, where the selection was based upon the unweighted samples. These correspond to a transformation from 700K to 300K, where the model was trained on $T_{\text{train}} \in \{400\text{K}, 500\text{K}, 600\text{K}, 700\text{K}, 800\text{K}, 900\text{K}, 1000\text{K}\}$.

We show the weighted histograms, using IQR-filtered weights given by Equation 3.10, along with the non-weighted histograms for comparison. Figure 4.7 depicts a side-to-side comparison of the non-weighted and the weighted torsion angles. As can be seen in the figure, weighting of the torsions seem to be slightly helpful in correcting model errors and the weighted torsion angles are more aligned with the reference histogram than the non-weighted torsion angles.



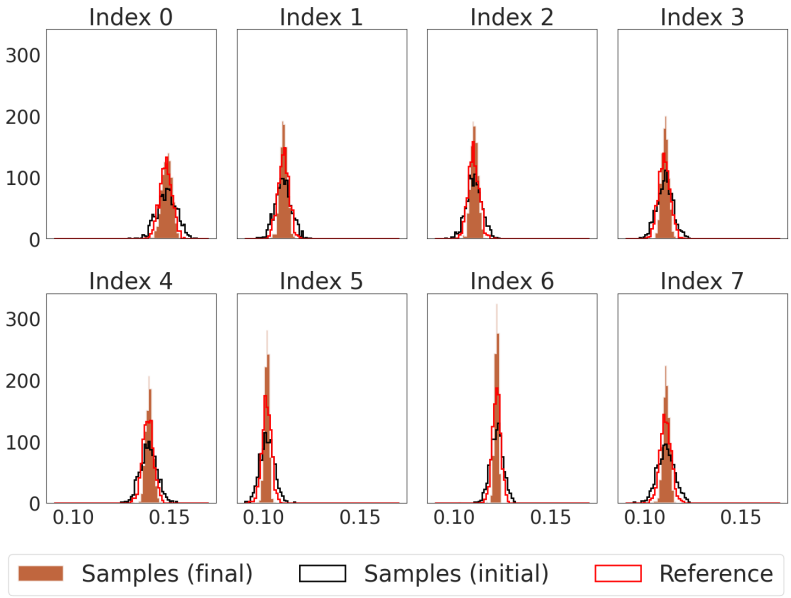
(a) Non-weighted torsion angles.



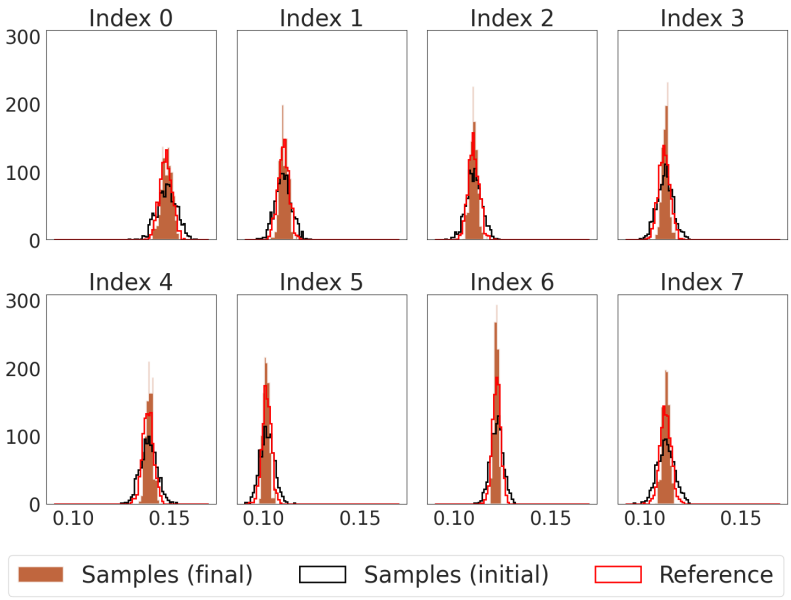
(b) Weighted torsion angles.

Figure 4.7: The weighted and non-weighted torsion angles. Subfigure 4.7(a) shows a histogram of the non-weighted torsion angles, while Subfigure 4.7(b) shows the weighted values. The filtered weights were obtained through IQR-filtering, more details can be found in Appendix A.3.

Figure 4.8 shows a side-to-side comparison of the non-weighted and the weighted bond lengths. While the impact of weighting of the bond lengths does not seem to be as significant as it was for the torsion angles, it is also hard to determine if it instead worsens the marginal histograms.



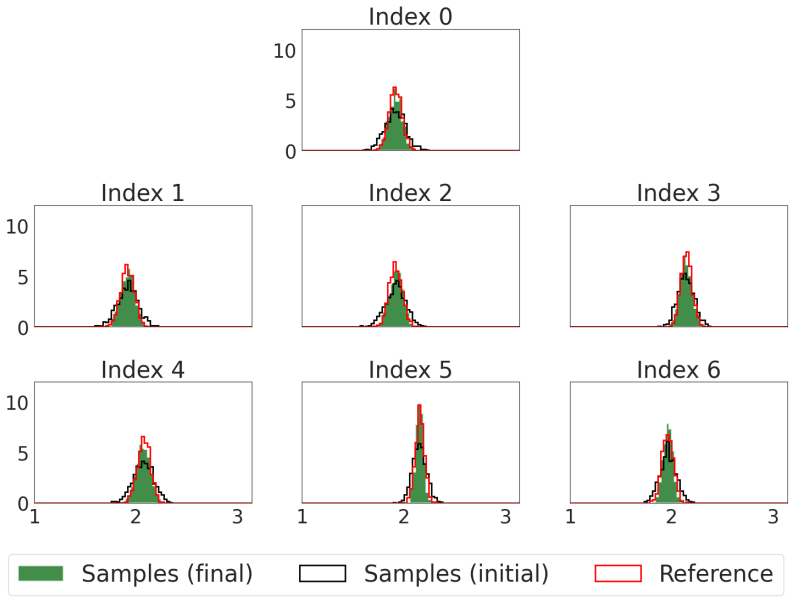
(a) Non-weighted bond lengths.



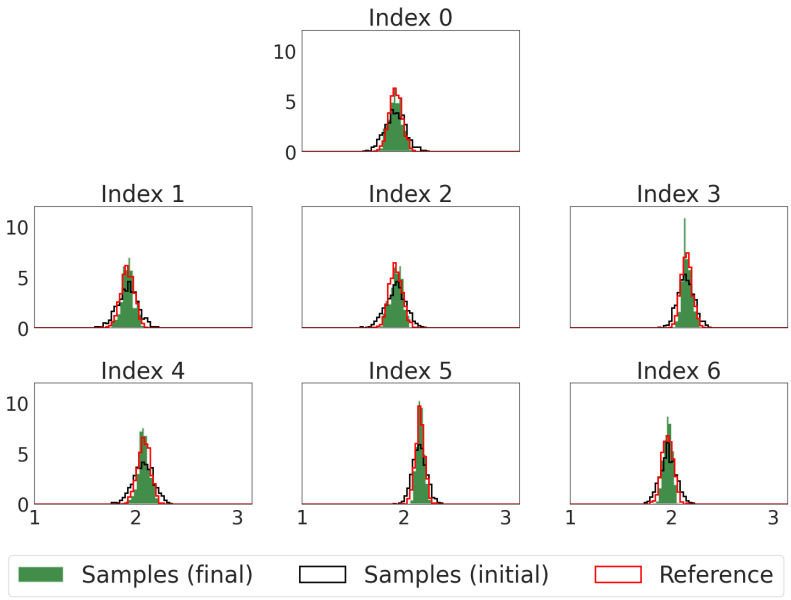
(b) Weighted bond lengths.

Figure 4.8: The weighted and non-weighted bond lengths. Subfigure 4.8(a) shows a histogram of the non-weighted bond lengths, while Subfigure 4.8(b) shows the weighted values. The filtered weights were obtained through IQR-filtering, more details can be found in Appendix A.3.

Figure 4.9 shows a side-to-side comparison of the non-weighted and the weighted bond angles. Just as with the bond lengths, the impact of weighting of the bond angles does not seem to be as significant as it was for the torsion angles. Again, it is hard to determine the impact on the bond angle marginal histograms.



(a) Non-weighted bond angles.



(b) Weighted bond angles.

Figure 4.9: The weighted and non-weighted bond angles. Subfigure 4.9(a) shows a histogram of the non-weighted bond angles, while Subfigure 4.9(b) shows the weighted values. The filtered weights were obtained through IQR-filtering, more details can be found in Appendix A.3.

Since the process of filtering outliers in the weights using IQR does not seem to significantly worsen the resulting marginal histograms of the bond angles and lengths, while at the same time it seemingly improves the torsions, we employ filtering in order to calculate the effective sample sizes. For a more thorough discussion on the matter, we refer the reader to Appendix A.3. The results of the interpolation experiment is shown in Figure 4.10, where the obtained effective sample sizes are

plotted against the target temperature T_1.

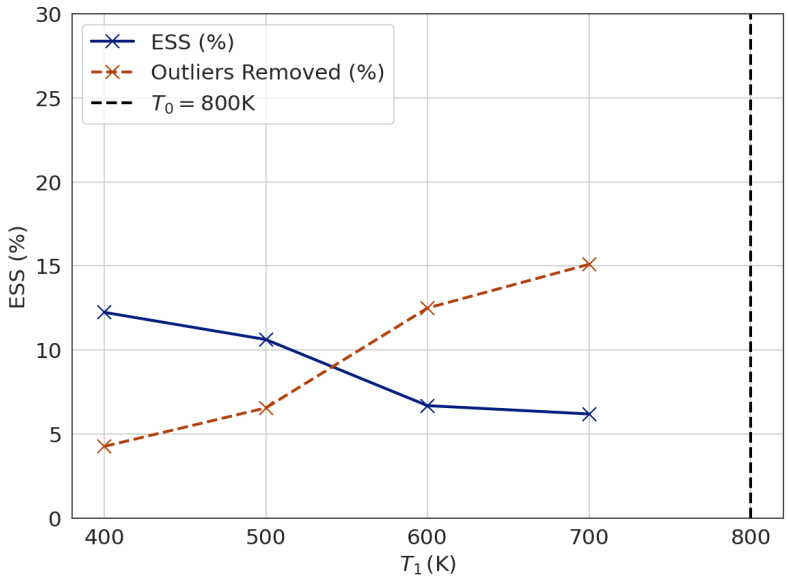


Figure 4.10: Effective sample sizes for the interpolation experiment, where we transform into from a T_0 that the model has seen during training into a T_1 that the model has not seen before, but which lies within the range of temperatures that the model has seen during training. IQR was used to filter out outliers in the weights so that they do not significantly impact the final outcome. For transparency, we therefore also include the percentage of outliers that were filtered out in the figure.

As can be seen in Figure 4.10, the models can sample at efficiencies between $5-10\%$. Still it is worth to note the visible correlation between the percentage of outliers removed and the obtained effective sample size.

Next, we present the results for the extrapolation experiment. Figure 4.11 shows the effective sample sizes plotted against the initial temperature T_0.

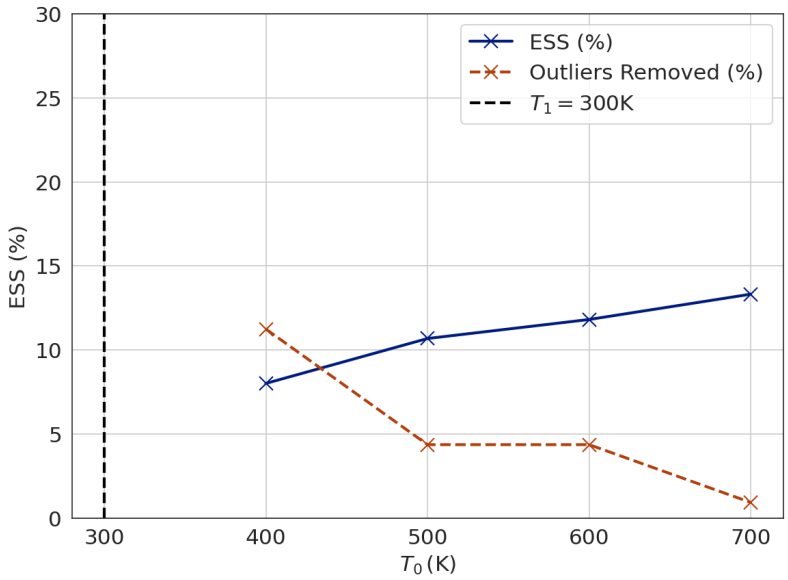


Figure 4.11: Effective sample sizes for the extrapolation experiment, where we transform into from a T_0 that the model has seen during training into a T_1 that the model has not seen before and lies outside the range of temperatures that the model has seen during training. IQR was used to filter out outliers in the weights so that they do not significantly impact the final outcome. For transparency, we therefore also include the percentage of outliers that were filtered out in the figure.

As can be seen in Figure 4.10, the models can also in this case sample at efficiencies between $7-13\%$, but again it is worth to point out the notable correlation between the percentage of outliers removed and the obtained effective sample size.

5

Conclusion

5.1 Discussion

5.1.1 Stochastic Interpolant ITO

Together with the above results, we demonstrate that the Stochastic Interpolant framework, as an unbiased generative model, is suitable for learning transition probability densities using the ITO learning method. This validates its capability to capture slow dynamics in molecular systems and learn from partially observed data.

With Stochastic Interpolants, we observed slightly better results in terms of VAMP-2 scores compared to the DDPM ITO. For molecular systems, we noted that longer training improves the model's accuracy slightly. For interpolant methods, the conditional linear one-sided interpolant shows its capability to accept conditional inputs and map Gaussian samples to the target distribution, similar to cDDPM. Unlike cDDPM, our model allows for the estimation of changes in probability density and remains unbiased compared to DDPM.

Theoretically, this Stochastic Interpolant ITO can be further applied to other probabilistic forecasting tasks. Instead of predicting deterministic values in typical time series tasks, this framework allows for the stochastic prediction of a probability density, which can be used for sampling or computing statistical variables. Further research is needed to address the data efficiency of the training process.

5.1.2 Thermodynamic Interpolants

The results in Section 4.2.1 makes it clear that the mirror interpolant does not perform better than direct re-weighting in most cases. As previously mentioned, this is likely due to accumulated numerical errors that are introduced through the scaling scheme. It is based upon quite a heavy approximation and it seems like this might be hindering its performance.

The augmented two-sided linear interpolant on the other hand, seems to not suffer from the same performance issues as the mirror interpolant. While effective sample sizes produced by both re-weighting and the mirror interpolant models seem to decay with the transformation distance, the augmented two-sided linear interpolant does not struggle with it as much. For the low-dimensional asymmetric double well system, the models can both interpolate and extrapolate with quite high sampling

efficiency. This opens up for the possibility of extending the augmented two-sided linear interpolant model to molecular systems.

Scaling the augmented two-sided linear model to higher-dimensional molecular systems proved to be a difficult task. When comparing marginal histograms of the different degrees of freedom, it seems as if the model has learned the distribution quite well. The effective sample sizes reported in Subsection 4.2.2 seem to agree with this result, returning values of between $5 - 13\%$ sampling efficiency. Although, here it is highly relevant to point out the impact of the IQR filtering. With no weight-filtering, the sampling efficiency is close to zero. While motivation of the approach is done in Subsection 4.2.2 through comparisons between filtered-weighted and non-weighted marginals, it is still debatable how solid this approach is. In previous work it has been shown that TFEP free energy estimates can benefit from filtering of outliers in the function ϕ. Since ϕ is related to the weights w through Equation (3.11), it could be argued that this type of filtering is relevant. Although not encompassed by this work, investigating the impact of filtering on free energy differences would be a further step to validate the method.

Regarding computational advantages of using thermodynamic interpolants to transform samples between temperatures, there is still much work to be done. The model architecture could unquestionably benefit from some optimization to allow for faster sampling. As of now, we have trained models on the smallest molecular system in the dataset, consisting of merely 27 dimensions (9 atoms). While we do find interesting properties in the thermodynamic interpolant models, such as allowing for interpolation and extrapolation, there is still some optimization work left until this method is efficient enough to compete with methods like Replica-Exchange Molecular Dynamics.

5.2 Summary

Connecting back to the initial goals and research questions of this project, we can conclude that the majority of them have been answered through the work presented in this thesis.

We utilized the advantages of the Stochastic Interpolants framework, which allows direct transformation between arbitrary densities, and successfully applied it to thermodynamic ensembles for both low-dimensional cases and molecular systems. To address the limitation of requiring training data from both ensembles, we explored various methods and finally proposed the augmented two-sided linear interpolant. This approach allows for both interpolation and extrapolation to predict unseen distributions. By computing the ESS, we demonstrated a computational advantage compared to direct MD simulation in the lower-dimensional case. However, optimization work of the model architecture could most likely be done to extend that advantage to molecular systems. Additionally, we successfully applied Stochastic Interpolants to ITO learning for both low-dimensional and molecular systems, out of consideration for paving the way for generalizing ITO across different thermodynamic states.

5.2.1 Future Work

While we have achieved the goals outlined in our thesis, there remains potential for model optimization and other future extensions.

First of all, our current model implementations employ an older version of the ChiroPaiNN architecture for, used in the message passing neural network architecture. Upgrading to more recent and advanced versions of ChiroPaiNN could possibly yield better performance and sample efficiency. To allow for faster sample generation, one could also experiment with different regularization techniques applied during training to reduce complexity of the learned vector fields.

Secondly, regarding the Thermodynamic Interpolants, the IQR-filtering that was applied to the weights should be investigated in more detail to understand the method's behavior and applicability. Possible approaches of doing so could include estimation of free energy differences or measurements of Kullback-Leibler divergences between sampled and reference marginals for the filtered-weighted, non-filtered-weighted and non-weighted cases.

Lastly, one should preferably try training on other model systems. As of now, the largest system we trained on was alanine dipeptide. Although we trained models on different molecules, each model has been trained on only one type of molecule. Future work could involve training on a diverse set of molecules to investigate and enhance the model's generalization capabilities.

Bibliography

[1] M. E. Tuckerman, *Statistical Mechanics: Theory and Molecular Simultion*, 2nd ed. Oxford: Oxford University Press, 2023.

[2] E. Marinari and G. Parisi, "Simulated tempering: A new monte carlo scheme," *Europhysics Letters (EPL)*, vol. 19, no. 6, pp. 451–458, Jul. 1992, ISSN: 1286-4854. DOI: 10.1209/0295-5075/19/6/002. [Online]. Available: http://dx.doi.org/10.1209/0295-5075/19/6/002.

[3] J. Hénin, T. Lelièvre, M. R. Shirts, O. Valsson, and L. Delemotte, "Enhanced sampling methods for molecular dynamics simulations [article v1.0]," *Living Journal of Computational Molecular Science*, vol. 4, no. 1, 2022, ISSN: 2575-6524. DOI: 10.33011/livecoms.4.1.1583. [Online]. Available: http://dx.doi.org/10.33011/livecoms.4.1.1583.

[4] F. Noé, S. Olsson, J. Köhler, and H. Wu, "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning," *Science*, vol. 365, no. 6457, eaaw1147, 2019. DOI: 10.1126/science.aaw1147. eprint: https://www.science.org/doi/pdf/10.1126/science.aaw1147. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaw1147.

[5] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, 2016. arXiv: 1505.05770 [stat.ML].

[6] E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Communications in Mathematical Sciences*, vol. 8, pp. 217–233, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:17933194.

[7] M. Schreiner, O. Winther, and S. Olsson, *Implicit transfer operator learning: Multiple time-resolution surrogates for molecular dynamics*, 2023. arXiv: 2305.18046 [physics.chem-ph].

[8] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. arXiv: 2006.11239 [cs.LG].

[9] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, *Ffjord: Free-form continuous dynamics for scalable reversible generative models*, 2018. arXiv: 1810.01367 [cs.LG].

[10] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *CoRR*, vol. abs/1806.07366, 2018. arXiv: 1806.07366. [Online]. Available: http://arxiv.org/abs/1806.07366.

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

[12] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, *Stochastic interpolants: A unifying framework for flows and diffusions*, 2023. arXiv: 2303.08797 [cs.LG].

[13] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[14] R. W. Zwanzig, "High-temperature equation of state by a perturbation method. i. nonpolar gases," *Journal of Chemical Physics*, vol. 22, pp. 1420–1426, 1954. [Online]. Available: https://api.semanticscholar.org/CorpusID:94688458.

[15] C. Jarzynski, "Targeted free energy perturbation," *Phys. Rev. E*, vol. 65, p. 046 122, 4 Apr. 2002. DOI: 10.1103/PhysRevE.65.046122. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.65.046122.

[16] W. E, T. Li, and E. Vanden-Eijnden, *Applied Stochastic Analysis*, 1st ed. United States of America: American Mathematical Society, 2019.

[17] P. Langevin, "Sur la théorie du mouvement brownien," *Compt. Rendus*, vol. 146, pp. 530–533, 1908.

[18] J.-H. Prinz, H. Wu, M. Sarich, *et al.*, "Markov models of molecular kinetics: Generation and validation," *The Journal of chemical physics*, vol. 134, no. 17, 2011.

[19] S. Klus, F. Nüske, P. Koltai, *et al.*, "Data-driven model reduction and transfer operator approximation," *Journal of Nonlinear Science*, vol. 28, pp. 985–1010, 2018.

[20] D. L. Ermak and J. A. McCammon, "Brownian dynamics with hydrodynamic interactions," *The Journal of Chemical Physics*, vol. 69, no. 4, pp. 1352–1360, Aug. 1978. DOI: 10.1063/1.436761. [Online]. Available: https://doi.org/10.1063/1.436761.

[21] Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics method for protein folding," *Chemical Physics Letters*, vol. 314, no. 1, pp. 141–151, 1999, ISSN: 0009-2614. DOI: https://doi.org/10.1016/S0009-2614(99)01123-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0009261499011239.

[22] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2017. arXiv: 1605.08803 [cs.LG].

[23] L. Klein, A. Krämer, and F. Noé, *Equivariant flow matching*, 2023. arXiv: 2306.15030 [stat.ML].

[24] A. Tong, K. Fatras, N. Malkin, *et al.*, *Improving and generalizing flow-based generative models with minibatch optimal transport*, 2024. arXiv: 2302.00482 [cs.LG].

[25] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velikovi, *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*, 2021. arXiv: 2104.13478 [cs.LG].

[26] A. Bietti, L. Venturi, and J. Bruna, *On the sample complexity of learning under invariance and geometric stability*, 2021. arXiv: 2106.07148 [stat.ML].

[27] J. Köhler, L. Klein, and F. Noé, *Equivariant flows: Exact likelihood generative learning for symmetric densities*, 2020. arXiv: 2006.02425 [stat.ML].

[28] K. T. Schütt, O. T. Unke, and M. Gastegger, "Equivariant message passing for the prediction of tensorial properties and molecular spectra," *CoRR*,

vol. abs/2102.03150, 2021. arXiv: 2102.03150. [Online]. Available: https://arxiv.org/abs/2102.03150.

[29] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, *Schnet: A continuous-filter convolutional neural network for modeling quantum interactions*, 2017. arXiv: 1706.08566 [stat.ML].

[30] F. Nüske, H. Wu, J.-H. Prinz, C. Wehmeyer, C. Clementi, and F. Noé, "Markov state models from short non-equilibrium simulationsanalysis and correction of estimation bias," *The Journal of Chemical Physics*, vol. 146, no. 9, Mar. 2017, ISSN: 1089-7690. DOI: 10.1063/1.4976518. [Online]. Available: http://dx.doi.org/10.1063/1.4976518.

[31] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241 703, Mar. 2018, ISSN: 0021-9606. DOI: 10.1063/1.5011399. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.5011399/16653374/241703_1_online.pdf. [Online]. Available: https://doi.org/10.1063/1.5011399.

[32] J. Viguera Diez, S. Romeo Atance, O. Engkvist, and S. Olsson, "Generation of conformational ensembles of small molecules via surrogate model-assisted molecular dynamics," *Machine Learning: Science and Technology*, 2024. [Online]. Available: http://iopscience.iop.org/article/10.1088/2632-2153/ad3b64.

[33] M. Invernizzi, A. Krämer, C. Clementi, and F. Noé, "Skipping the replica exchange ladder with normalizing flows," *The Journal of Physical Chemistry Letters*, vol. 13, no. 50, pp. 11 643–11 649, 2022.

[34] M. Dibak, L. Klein, A. Krämer, and F. Noé, "Temperature steerable flows and boltzmann generators," *Physical Review Research*, vol. 4, no. 4, p. L042005, 2022.

[35] L. Herron, K. Mondal, J. S. Schneekloth, and P. Tiwary, *Inferring phase transitions and critical exponents from limited observations with thermodynamic maps*, 2023. arXiv: 2308.14885 [cond-mat.stat-mech].

[36] L. Kish, *Survey Sampling*. New York: John Wiley & Sons, Inc, 1965.

[37] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq.

A

Appendix 1

A.1 Formal Definitions

Below we include a few formal definitions that we considered overly complicated or lengthy to put in the main report.

A.1.1 Stochastic Interpolant

While we introduced the stochastic interpolant in Section 2.3.3, we did not include the full definition there. A proper definition can instead be found below.

A stochastic interpolant is defined as an interpolant between individual samples $\mathbf{x}_0 \sim \rho_0(\mathbf{x})$ and $\mathbf{x}_1 \sim \rho_1(\mathbf{x})$, where the initial distribution ρ_0 and the target distribution ρ_1 are such that $\rho_0,\, \rho_1 : \mathbb{R}^d \to \mathbb{R}^+$. Specifically, it is defined as a stochastic process $\mathbf{x}_t$ such that

$$\mathbf{x}_t = I(t,\, \mathbf{x}_0,\, \mathbf{x}_t) + \gamma(t)\mathbf{z}, \quad t \in [0,\, 1], \tag{A.1}$$

where

1. $I \in C^2\left([0,\, 1],\, (C^2(\mathbb{R}^d \times \mathbb{R}^d)^d)\right)$ fulfills the boundary conditions $I(0,\, \mathbf{x}_0,\, \mathbf{x}_1) = \mathbf{x}_0$ and $I(1,\, \mathbf{x}_0,\, \mathbf{x}_1) = \mathbf{x}_1$. Another condition can be put upon $I(t,\, \mathbf{x}_0,\, \mathbf{x}_1)$, namely

   $$\exists C_1 < \infty : \quad |\partial_t I(t,\, \mathbf{x}_0,\, \mathbf{x}_1)| \leq C_1 |\mathbf{x}_0 - \mathbf{x}_1|, \tag{A.2}$$

   so that the individual samples never move too fast along their respective trajectories.

2. $\gamma : [0,\, 1] \to \mathbb{R}$ fulfills $\gamma(0) = \gamma(1) = 0$, $\gamma(t) > 0$ for all $t \in (0,\, 1)$ and $\gamma^2 \in C^2([0,\, 1])$.

3. The pair $(\mathbf{x}_0,\, \mathbf{x}_1)$ is drawn from a probability measure ν that marginalizes on ρ_0 and ρ_1.

4. The standard normally distributed random variable $\mathbf{z}$ is independent of the pair $(\mathbf{x}_0,\, \mathbf{x}_1)$ [12].

A.2 Training and Sampling Details

A lot of information on exact training and sampling schemes has been left out of Chapter 3. To improve reproducibility of our results, we leave a more in-depth description here.

A.2.1 Hyperparameter Choices

In this subsection, we report choices of the hyperparameters used during training and when sampling with the different models.

A.2.1.1 Asymmetric Double Well (ADW)

ADW training data is generated through Langevin dynamics in a high-friction regime, employing several hyperparameters shown in Table A.1 to control the simulation. Table A.2 shows the hyperparameter choices used for the models trained on

Table A.1: Hyperparameters for ADW Data Generation

Parameter	Value
Time Step (`dt`)	0.001
β-range	[0.25, 2]
Friction Coefficient (`mGamma`)	1.0
Initial Position Range (`x0`)	$[-1.5, 1.5]$
Number of Trajectories (`N_trajs`)	10
Number of Simulation Steps (`N_samples`)	100 000
Burn-in Steps	1000
Save Frequency	10

the asymmetric double well potential.

Table A.2: Hyperparameters used for the asymmetric double well model dataset.

Parameter	Value
Training Size	240 000
Validation Size	30 000
Test Size	30 000
Hidden Size	256
Number of Layers	4
Learning Rate	10^{-4}
Batch Size	256
Number of Epochs	100
Solver Type	Dopri5
Relative Solver Tolerance	10^{-4}
Absolute Solver Tolerance	10^{-4}

A.2.1.2 Müller Brown (MB)

For MB ITO dataset, trajectories are generated with hyperparamters shown in Table A.3. Table A.4 shows the hyperparameter choices used for the models trained on the Müller Brown potential.

Table A.3: Hyperparameters for MB Dataset Generation

Parameter	Value
Time Step (`dt`)	0.1
Thermal Energy (`kT`)	15.0
Friction Coefficient (`mGamma`)	1000.0
Initial Position Range (`x0`)	$(\mathbf{x}, \mathbf{y}) = ([-1.5, 1.2], [-0.2, 2.0])$
Number of Trajectories (`N_traj`)	32
Number of Simulation Steps (`N_step`)	100 000
Burn-in Steps (`burnin`)	1000
Save Frequency	10

Table A.4: Hyperparameters used for the Müller Brown model dataset.

Parameter	Value
Training Size	250 880
Validation Size	31 360
Number of Samples	250 000
Hidden Size	128
Number of Layers	4
Learning Rate	10^{-3}
Batch Size	128
Number of Epochs	300
Solver Type	Dopri5
Relative Solver Tolerance	10^{-5}
Absolute Solver Tolerance	10^{-5}

A.2.1.3 Alanine Dipeptide (ALA2)

Table A.5 shows the hyperparameter choices used for the models trained on the Alanine Dipeptide dataset.

Table A.5: Hyperparameters used for the alanine dipeptide model dataset.

Parameter	Value
Training Size	630 000
Validation Size	35 000
Number of Samples	15 000
Hidden Size	64
Number of Layers	5
Learning Rate	10^{-4}
Batch Size	128
Number of Epochs	120
Solver Type	Euler
Integration Steps	100

A.2.1.4 Replica Exchange MDQM9 (MDQM9-nc)

When training the thermodynamic interpolant model on the MDQM9 dataset, we highlight the importance of reducing the network complexity. Adaptive-step solvers such as Dopri5 provide an easy way of specifying the tolerated error in the solution. However, they can struggle to produce a solution if the ODE becomes too stiff [37]. In a stiff ODE the adaptive solvers sometimes crashes with underflow errors, since the solver fails to take a step that produces an error less than allowed by the specified tolerances. To counteract this behaviour, we implemented SiLU activation functions throughout the network and used weight-decay during training.

The MDQM9 dataset is quite limited in regard to the number of samples, with there only being 10 000 samples per temperature. This means that it is important to use the available data in an effective manner. Since selecting a model that performs well on the validation split does not directly translate to it performing well in an interpolation or extrapolation setting, we chose to disregard the use of a validation split. Instead, we pick the model that performs the best on the full training set, while using early stopping to reduce the risk of overfitting.

With the above in mind, we selected the hyperparameters seen in Table A.6 for the thermodynamic interpolant models trained on the Replica Exchange MDQM9 dataset.

Table A.6: Hyperparameters used for the MDQM9-nc dataset

Parameter	Value
Training Size	72 000
Test Size	1000
Hidden Size	256
Number of Layers (MLP)	2
Number of Layers (embedding)	2
Number of Layers (ChiroPaiNN)	5
Learning Rate	10^{-4}
Weight Decay	10^{-4}
Batch Size	256
Number of Epochs	75
Solver Type	Dopri5
Relative Solver Tolerance	10^{-5}
Absolute Solver Tolerance	10^{-5}

A.2.2 Solver Information

We employ the use of two different solvers, namely the Dopri5 and the Euler solvers. While the inner mechanisms of the solvers are not so relevant to us, a brief overview of important differences is given below.

A.2.2.1 Euler Solver

The Euler solver is a fixed step solver. A step size is given upon initialization of the solver, then the ODE will be solved only using this step size. In this way, the step size will not vary throughout the integration process [37].

A.2.2.2 Dopri5 Solver

In difference to the Euler solver, the Dopri5 solver is an adaptive step solver, implying that it does not use a fixed step size. When initializing the solver, one provides an absolute and a relative tolerance. Then, the solver adapts its step size according to the specified tolerances. The step is taken in such a way that it is large to begin with, then the errors in the provided solution are estimated. If the error is larger than what the specified tolerances allow, the step size is recalculated. This process is repeated until the error falls below the threshold specified through the tolerances, then the step is actually taken. In this way, it is possible to solve the ODE to the accuracy of the given tolerance [37].

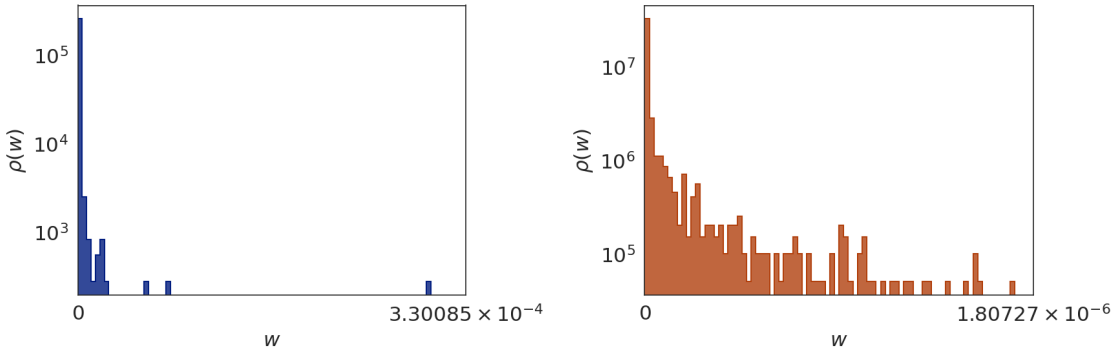
A.3 Filtering of Outliers

The estimated weights can include outliers, i.e. values that empirically lie extremely far away from the remaining values. Such outliers can significantly impact the estimated quantities, possibly rendering the generated samples and their probability

estimates seemingly useless. To address this issue, a number of the outlier values were filtered out, hopefully leading to more reliable and numerically stable estimates.

The filtering was done by calculating the interquartile range (IQR), which in our case is defined as the difference between the lower and upper quartiles of the weight distribution. An outlier was identified as a data point lying outside the range $[Q_1 - k \cdot \text{IQR}, Q_3 + k \cdot \text{IQR}]$, where Q_1 and Q_3 represent the lower and upper quartiles respectively, and k is a constant. The constant k was set to $k = 5$ in the lower-dimensional case and to $k = 20$ in the higher-dimensional case. Note that removing too many samples will reduce sampling efficiency, since samples are filtered out, resulting in a lower total amount of samples in the end. It is generally also not a good approach to remove higher valued statistical weights, as they theoretically correspond to more relevant samples. Setting $k = 1.5$ corresponds to 3σ if the data is normally distributed, so we hope that our choice only removes the most extreme outliers.

Figure A.1 gives an intuitive idea of the issue. As can be seen in Subfigure A.1(a), the distribution of non-filtered weights have an extremely long tail where the weights furthest to the right seem to lie very far from the values to the left. Subfigure A.1(b) on the other hand, shows the distribution of filtered weights, where the values furthest to the right have now been removed.



(a) Non-filtered weights.

(b) Filtered weights.

Figure A.1: A comparison of example histograms of non-filtered and filtered weights. The weights were obtained on the smallest molecule of the replica exchange MDQM9-nc dataset when transforming from $T_0 = 800\text{K}$ to $T_1 = 400\text{K}$ using a model trained on temperatures $T_{\text{train}} \in \{300\text{K}, 500\text{K}, 600\text{K}, 700\text{K}, 800\text{K}, 900\text{K}, 1000\text{K}\}$ and a cutoff of $k = 20$.

As a small investigation into why this issue appears, we also show the obtained end-point energies, E_1, plotted against the change in log probability $\Delta \log \rho$ in Figure A.2. In the figure, values corresponding to the weights from Figure A.1 that were considered outliers by the IQR algorithm are marked in a different color than values that were not. Since the weights, defined through Equation (3.10), are computed through an exponentiation, a small value of E_1 could result in a large weight. As can be seen in Figure A.2, the weights that were filtered out possibly seem to correspond

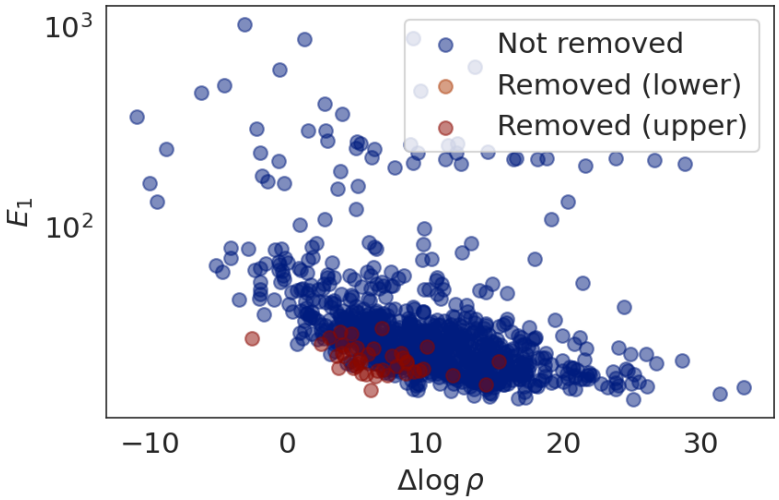
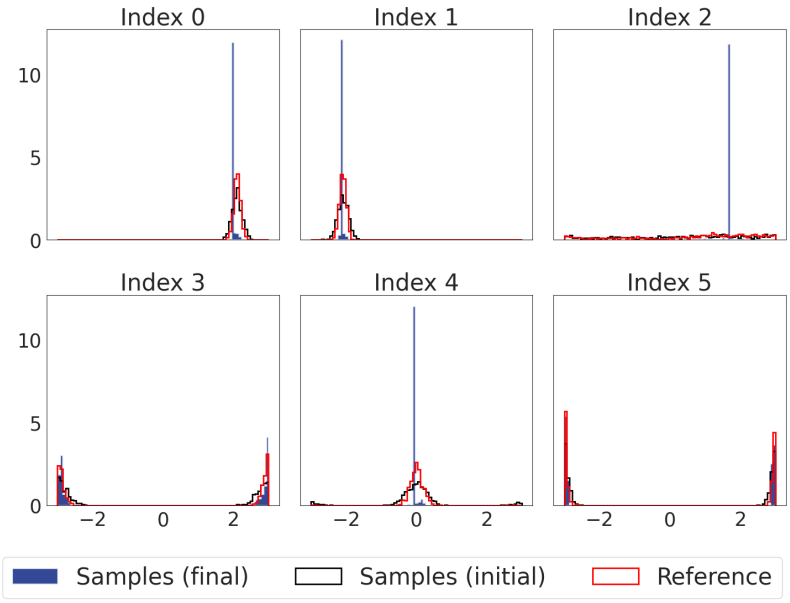


Figure A.2: The change in log probability, $\Delta \log \rho$, is plotted against the end-point energies, E_1. In the figure, all values are shown, but those corresponding to the removed "outlier" weights from Figure A.1 are marked in a different color than the ones the IQR algorithm did not consider outliers. For clarity, values corresponding to outliers removed by the lower versus the upper bound are plotted in different colors.

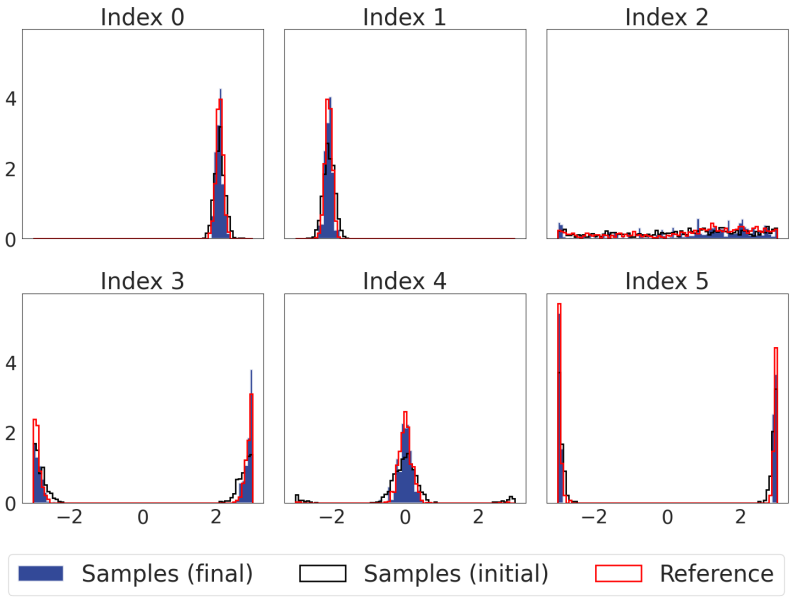
to that scenario.

A.3.1 Comparison of Filtered and Non-Filtered Weights

In Section 4.2.2, we made a comparison between marginal histograms of bond angles, bond lengths and torsion angles. There, we showed how weighting with the filtered weights impacted the resulting marginals in comparison to the original non-weighted marginal histograms. To reduce redundancy in Chapter 4, we did not make a direct comparison between the non-filtered weighted histograms and the filtered weighted histograms there. Instead this comparison is done below. Figure A.3 depicts a side-to-side comparison of the non-filtered weighted and the filtered weighted torsion angles. As can be seen in the figure, filtering seems to improve the resulting marginals significantly compared to when using non-filtered weights. By not including very large weights we obtain marginal distributions that adhere more to the reference marginals, and as was shown in Section 4.2.2 using such weights seemed to slightly improve the results compared to the non-weighted case as well.



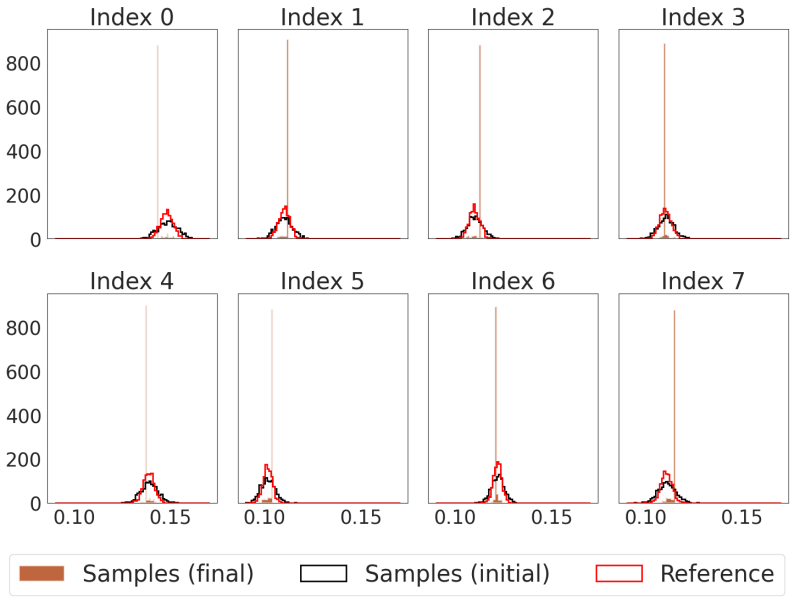
(a) Non-filtered weighted torsion angles.



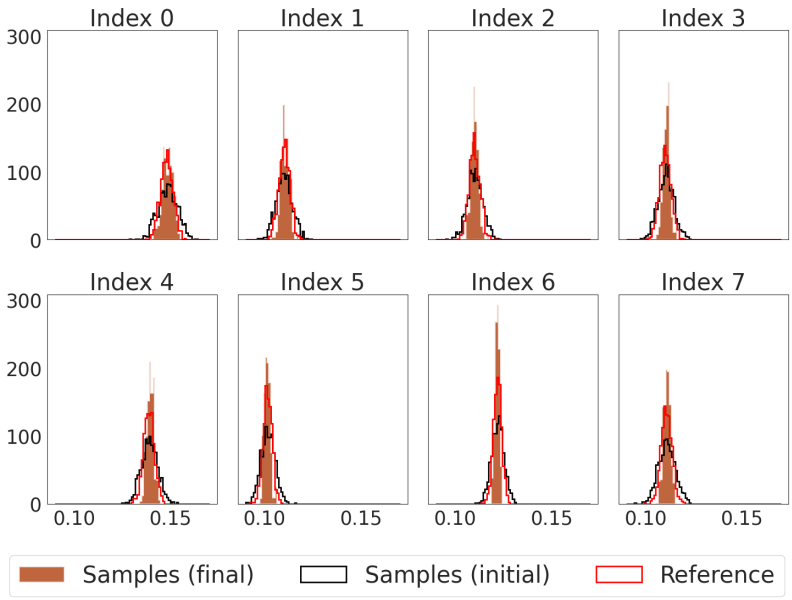
(b) Filtered weighted torsion angles.

Figure A.3: The non-filtered weighted and filtered weighted torsion angles. Subfigure A.3(a) shows a histogram of the non-filtered weighted torsion angles, while Subfigure A.3(b) shows the filtered weighted values.

Figure A.4 depicts a side-to-side comparison of the non-filtered weighted and the filtered weighted bond lengths. As can be seen in the figure, filtering seems to again improve the resulting marginals significantly compared to when using non-filtered weights, and as was shown in Section 4.2.2 using such weights seemed to at least not worsen the results compared to the non-weighted case.



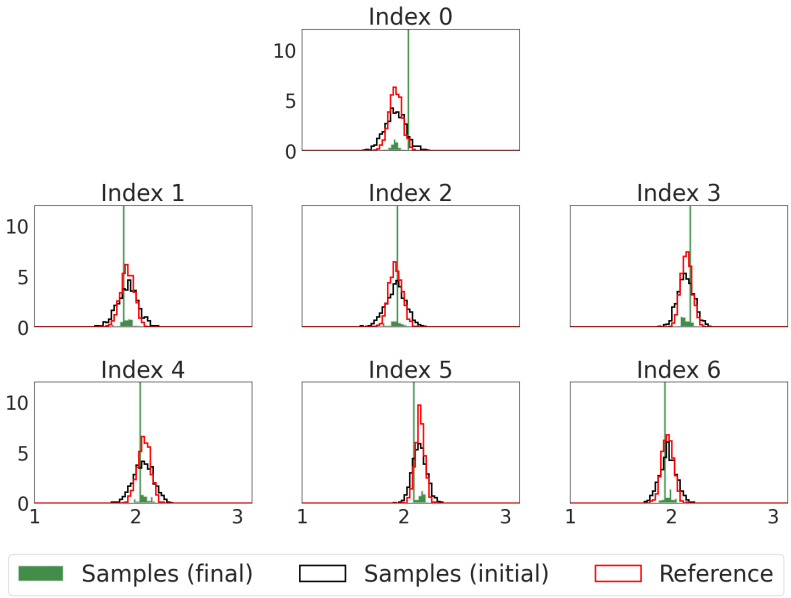
(a) Non-filtered weighted bond lengths.



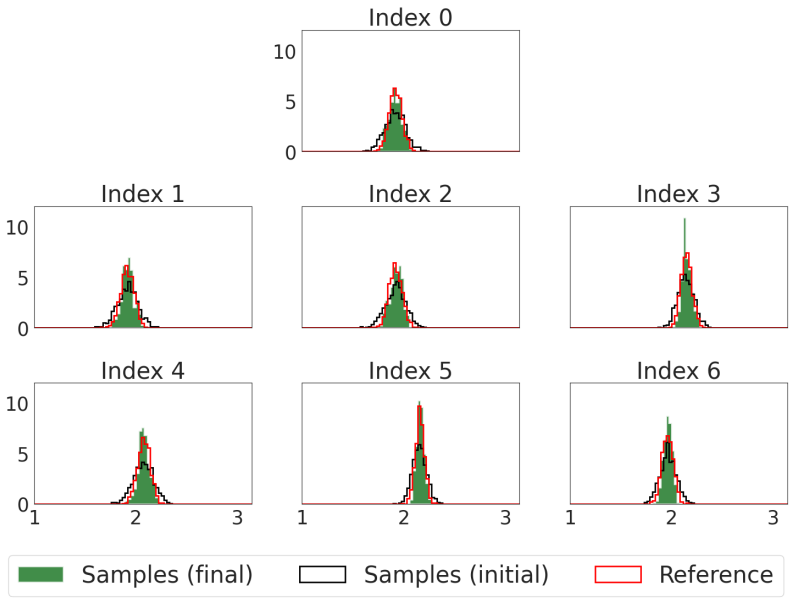
(b) Filtered weighted bond lengths.

Figure A.4: The non-filtered weighted and filtered weighted bond lengths. Subfigure A.4(a) shows a histogram of the non-filtered weighted bond lengths, while Subfigure A.4(b) shows the filtered weighted values.

Lastly, Figure A.5 depicts the side-to-side comparison of the non-filtered weighted and the filtered weighted bond angles. As again can be seen in the figure, filtering seems to again improve the resulting marginals significantly compared to when using non-filtered weights, and as was shown in Section 4.2.2 using such weights seemed to at least not worsen the results compared to the non-weighted case.



(a) Non-filtered weighted bond angles.



(b) Filtered weighted bond angles.

Figure A.5: The non-filtered weighted and filtered weighted bond angles. Subfigure A.5(a) shows a histogram of the non-filtered weighted bond angles, while Subfigure A.5(b) shows the filtered weighted values.

A.4 Additional Results

In this section we fit a few results that were considered unnecessary for the main report, but still interesting enough to include.

A.4.1 Training and Validation Losses

In this subsection, we show plots of losses to show convergence of our models.

A.4.1.1 Müller Brown

Figure A.6 shows the training loss for the four Stochastic Interpolant ITO models trained on the Müller Brown dataset.

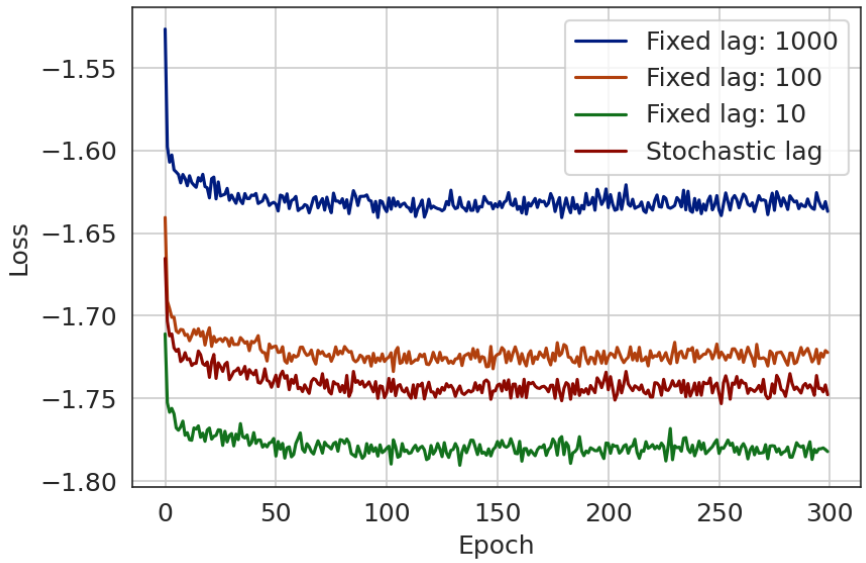


Figure A.6: Training losses for stochastic interpolants ITO model on Müller Brown dataset. Trained with fixed lags $N = 10, 100, 1000$ and stochastic lag.

A.4.1.2 Alanine Dipeptide

Figure A.7 shows the training loss of the Stochastic Interpolant ITO models trained on the alanine dipeptide dataset.

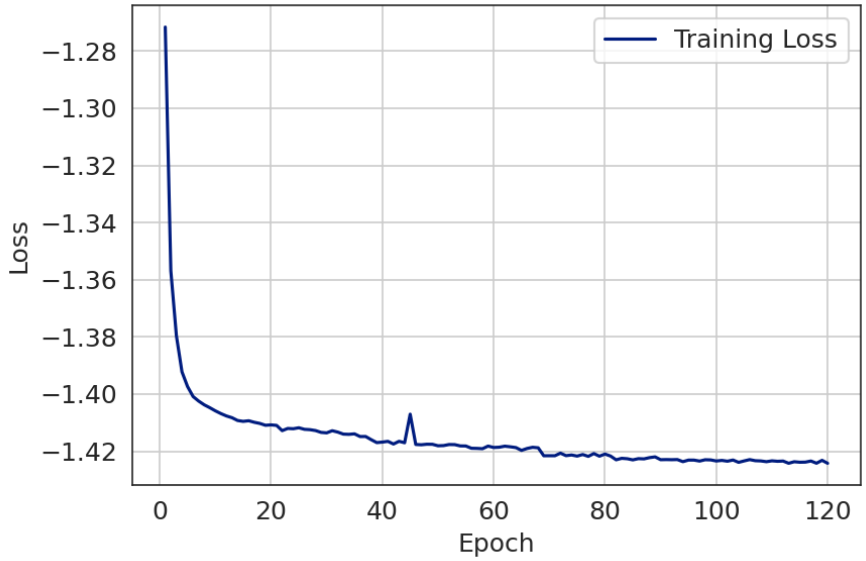


Figure A.7: Training loss for the Stochastic Interpolants ITO model with the alanine dipeptide dataset. Initially, the model is trained for 120 epochs.

A.4.1.3 Asymmetric Double Well

Figure A.8 shows the training and validation loss of the augmented two-sided linear interpolant model trained data at inverse temperatures $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$ from the asymmetric double well dataset.

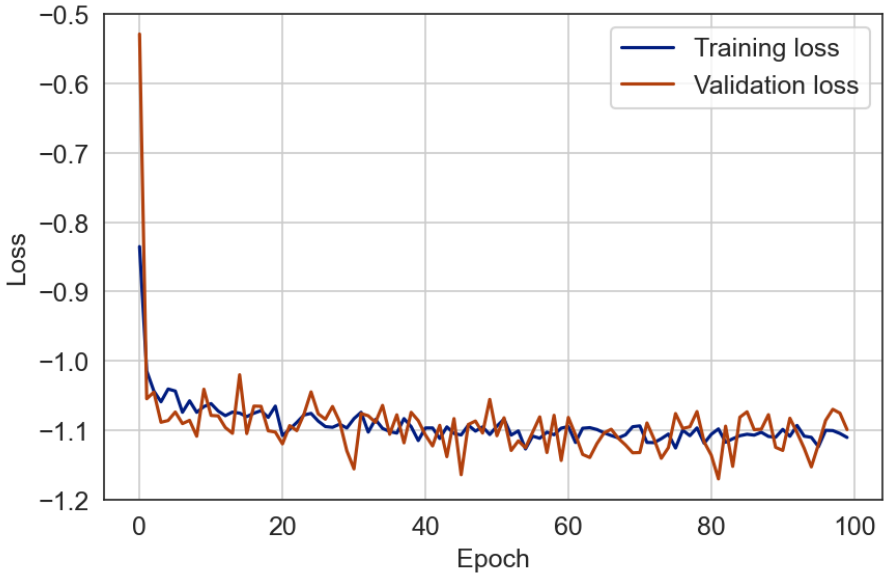


Figure A.8: Training and validation losses for the augmented two-sided linear thermodynamic interpolant model trained on $\beta_{\text{train}} \in \{1.00, 1.25, 1.50\}$.

Figure A.9 shows the training and validation loss of the augmented two-sided linear interpolant model trained data at inverse temperatures $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$ from the asymmetric double well dataset.

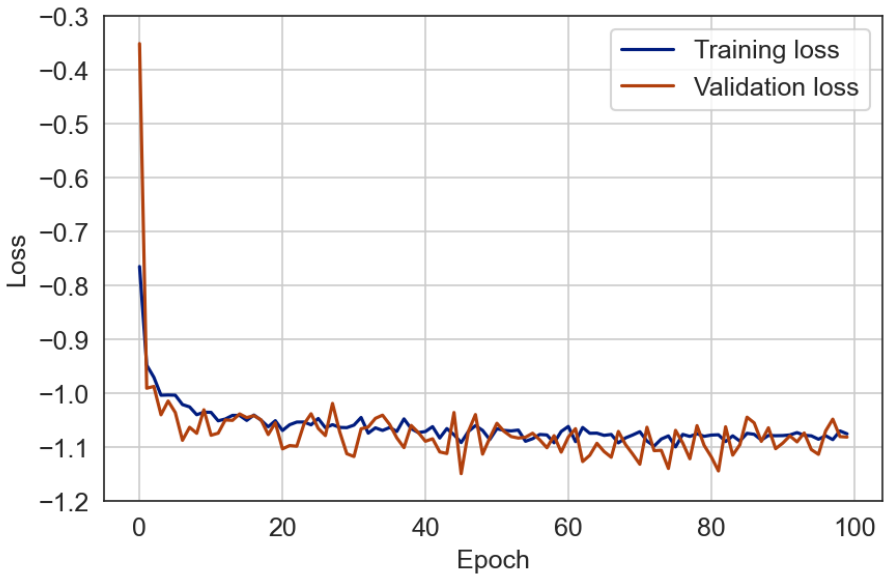
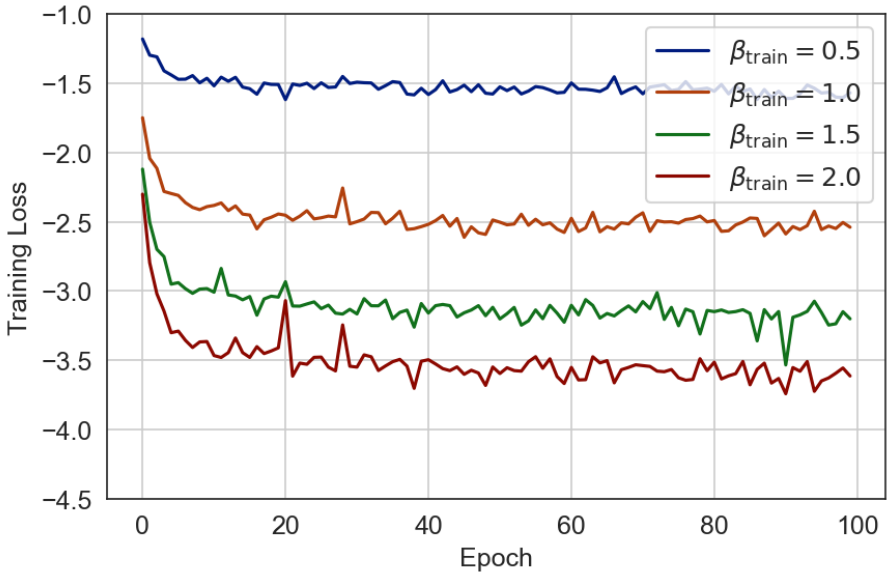
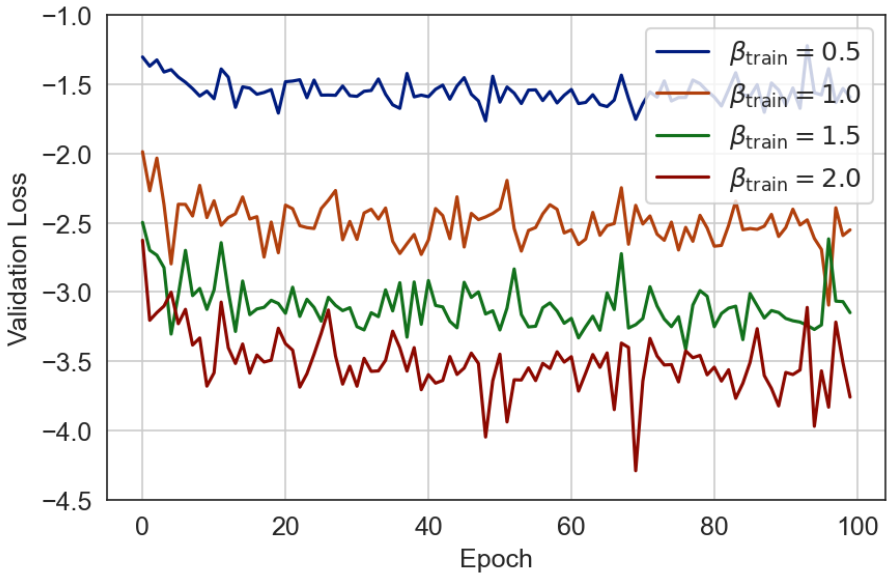


Figure A.9: Training and validation losses for the augmented two-sided linear thermodynamic interpolant model trained on $\beta_{\text{train}} \in \{0.50, 1.25, 2.00\}$.

Figure A.10 shows the training and validation losses of the four mirror interpolant models trained on data at inverse temperatures $\beta_{\text{train}} = 0.50, 1.00, 1.50, 2.00$, from the asymmetric double well dataset.



(a) Training loss.



(b) Validation loss.

Figure A.10: Training and validation losses for four different mirror interpolant models trained on $\beta_{\text{train}} \in \{0.50, 1.00, 1.50, 2.00\}$.

A.4.1.4 Replica Exchange MDQM9

Figure A.11 shows the training losses of the augmented two-sided linear interpolant models trained on data from the Replica Exchange MDQM9 dataset.

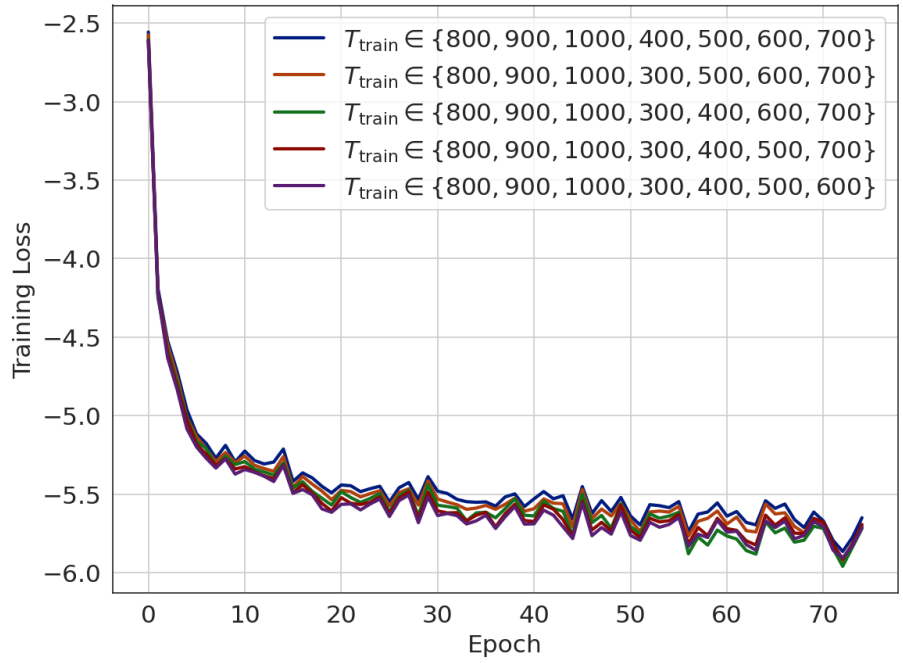


Figure A.11: Training and validation losses for the augmented two-sided linear thermodynamic interpolant models trained on the MDQM9 dataset.