universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Correlation-free and parallelizable Transition Path Sampling using Boltzmann Generators and committor learning

verfasst von | submitted by

Maximilian Negedly BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 876
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Physics
Betreut von | Supervisor:	Univ.-Prof. Mag. Dr. Christoph Dellago

Acknowledgements

I would like to start by thanking all the people who helped me on this rather extensive journey. First, Sebastian Falkner, who went out of his way to support me, be it by answering my questions, showing me tricks, finding bugs in my code that would have taken me days to locate or simply by taking my mind off work through discussions of unrelated topics. Next, my supervisor Christoph Dellago for always providing an open ear (and door) for any problem or question I had, suggesting new ways forward and helping me understand fundamental ideas. Alessandro Coretti, for our helpful discussions even outside of meetings and his positive and supportive attitude all the way through. My friends and office mates Lydia Morscher, Nora Giesinger and Salvatore Romano, whom I could always rely on to give me feedback and provide much needed mental support, even if that meant spending hours talking and drinking coffee in the kitchen instead of focussing on our work. Lastly, my parents for all the loving support I received and continue to receive. All of you made this possible, and you have my deepest gratitude for that.

Abstract

Molecular and atomistic systems often exhibit a complex energy landscape with multiple stable and metastable states in which they preferentially dwell. This poses a challenge if one wishes to perform computer simulations of such systems, as the majority of computer time will be spent in those stable states, while interesting transitions between them are rare. Transition Path Sampling (TPS) is a well established method that focusses computational resources on the transition region, but its efficiency is hindered by its sequentiality, its ignorance towards potentially useful information provided by the underlying reaction mechanism and by oftentimes long correlation times between the samples it produces. Previous works have used a generative machine learning model called Boltzmann Generators to eliminate correlations between paths and make TPS parallelizable, but at the cost of requiring knowledge of a good reaction coordinate parametrizing the transition, which is highly non-trivial or even impossible to obtain in complex systems. This work explores the use of the committor of a system as an ideal reaction coordinate for accelerating path sampling. Since it is intractable analytically and costly to obtain through simulation, machine learning is used again to learn the committor. However, this approach requires uncorrelated training data obtained via TPS, which in turn can be produced using Boltzmann Generators. This thesis proposes an algorithm that resolves this circular dependence by combining committor learning with the generative capabilities of Boltzmann Generators, training both machine learning models simultaneously in a self-consistent way. The result is a new TPS scheme that is fully parallelizable, produces uncorrelated samples and optimises sampling efficiency without prior knowledge of a reaction coordinate. The algorithm is tested on two example systems: a two-dimensional model and a simple polymer model, acting as a proof of concept for efficient generative path sampling.

Deutschsprachiger Abstract

Molekulare und atomistische Systeme halten sich bevorzugterweise in Regionen minimaler potentieller oder freier Energie, sogenannten stabilen oder metastabilen Zuständen, auf. Will man solche Systeme per Computer simulieren, stellt diese Eigenschaft eine große Herausforderung dar, da ein beträchtlicher Teil der begrenzten Simulationszeit innerhalb dieser Zustände verbracht wird. Ist man daran interessiert, das Übergangsverhalten eines solchen Systems zu verstehen, so ist dies besonders problematisch, da Übergänge selten und daher kostspielig zu gewinnen sind. Transition Path Sampling (TPS) ist eine bewährte Methode, Übergänge zwischen stabilen Zuständen zu erzwingen. Der ihr zugrundeliegende Algorithmus wird allerdings durch drei Faktoren in seiner Effizienz reduziert. Einerseits sind die Daten, die er produziert, oftmals mit langen Korrelationszeiten behaftet. Andererseits ist er durchwegs sequenziell, weshalb er nicht von heutzutage üblichen parallelen Rechnerstrukturen wie Supercomputern profitieren kann. Außerdem ignoriert er die dem Übergang zugrundeliegenden Mechanismen, obwohl eine geeignete Nutzung dieser oftmals seine Effizienz erhöhen kann. Bestehende Publikationen haben gezeigt, dass es möglich ist, ein generatives Machine Learning Modell namens Boltzmann Generator zu nutzen, um TPS parallelisierbar zu machen und Korrelationen zu eliminieren, jedoch unter der Voraussetzung, dass eine geeignete Reaktionskoordinate bekannt ist, welche den Übergang parametrisiert. Solch eine Reaktionskoordinate ist jedoch meist schwer oder gar unmöglich zu finden. Diese Arbeit verwendet den Committor eines Systems als ideale Reaktionskoordinate. Da dieser weder analytisch berechnet noch effizient im Rahmen von Simulationen approximiert werden kann, wird abermals Machine Learning eingesetzt, um den Committor zu erlernen. Dafür werden allerdings unkorrelierte Trainingsdaten aus einer TPS Simulation benötigt, welche, wie bereits diskutiert, mithilfe von Boltzmann Generatoren erzeugt werden können. In dieser Arbeit wird ein Algorithmus entwickelt, welcher das Erlernen des Committors mit dem Erzeugen unkorrelierter Konfigurationen mittels Boltzmann Generatoren kombiniert und das simultane Training beider Machine Learning Modelle auf selbstkonsistente Art und Weise ermöglicht. Das Resultat ist eine neue TPS Methode, welche unkorrelierte Daten produziert, vollständig parallelisiert werden kann und sich die dem Übergang zugrundeliegenden Mechanismen effizient zunutze macht - all dies ohne vorherige Kenntnis einer Reaktionskoordinate. Dieser Algorithmus wird mithilfe zweier Testsysteme auf seine Funktionsfähigkeit überprüft: einem einfachen zweidimensionalen Modell und einem komplexeren Modell eines simplen Polymers. Dies unterstreicht das Potential von effizientem generativem Path Sampling.

Contents

List of Acronyms

AIMMD Artificial Intelligence for Molecular Mechanism Discovery.

ANNs Artificial Neural Networks.

BGs Boltzmann Generators.

CVs Collective Variables.

GANs Generative Adversarial Networks.

GenAIMMD Generative AIMMD.

LJ Lennard-Jones.

MC Monte Carlo.

MCMC Markov Chain Monte Carlo.

MD Molecular Dynamics.

MLP Multilayer Perceptron.

MLPs Multilayer Perceptrons.

MSE Mean Squared Error.

NF Normalizing Flow.

NFs Normalizing Flows.

RC Reaction Coordinate.

ReLU Rectified Linear Unit.

RESS Relative Effective Sample Size.

SP Shooting Point.

TP Transition Path.

TPS Transition Path Sampling.

US Umbrella Sampling.

VAEs Variational Auto-Encoders.

WHAM Weighted Histogram Analysis Method.

1. Introduction

Understanding the behaviour of complex molecular systems has been one of the primary goals of modern statistical physics for the past several decades and is essential to advance research in other fields of science. Although the physical and mathematical models describing such systems have mostly been known since long before the first general purpose computer was ever built, their applicability in solving problems analytically is restricted to a limited class of idealized systems, often far away from anything one might encounter in the real world, necessitating the use of numerical simulations.
A computer simulation of a molecular system typically revolves around propagating its microstate (e.g. the positions and momenta of the atoms) by evaluating interactions between its constituents, while extracting the macro- and/or microscopic information of interest [1]. In contrast to a real experiment, one has full control over every aspect of the simulation such as the initial condition and desired evolution equation. The two main methods for propagating a system are Molecular Dynamics (MD) and Markov Chain Monte Carlo (MCMC), often abbreviated as MC. MD propagates a system by integrating some equation of motion, while MCMC uses a purely stochastic approach that does not aim to reproduce any physically realistic dynamics. Both methods can be used to produce configurations (samples) that follow the stationary probability distribution of the system, which, in turn, allow for the estimation of observables through statistical averages. MD additionally provides meaningful information about the time evolution of a system, which for example enables one to study the time dependence of observables and to calculate transport coefficients. A trait that most sampling algorithms share is that the samples they produce are correlated. The correlation times depend on the specific algorithm and, most importantly, the system under consideration. Complex systems can be exceedingly difficult to sample because of long correlation times.
Although estimated averages from samples obtained via either method are guaranteed to converge to the correct statistical averages in the limit of infinite sampling, limited computational resources gives rise to a plethora of new challenges. One of these is the occurrence of so-called *rare events*, i.e., events that happen rarely but exceedingly quickly if they do. They are usually a consequence of large energy barriers between high-probability regions in phase space that the system must cross. Examples include the nucleation of a raindrop from oversaturated water vapour [2, 3] and the transition of supercooled water from the liquid phase to the solid phase [4]. While it is often computationally inexpensive to simulate a system within a high probability region, for example, in stable or metastable states, it is usually unfeasible to wait until that system spontaneously transitions from one state to another. This is particularly expensive if one is interested in obtaining good statistics within the transition region. Therefore, it can become necessary to employ special *enhanced sampling* methods that drive the system to explore larger portions of

phase space in shorter simulation times. Such methods include Umbrella Sampling [5], Replica Exchange methods [6–9], Metadynamics [10] and Transition Path Sampling [11]. In this work, the main method of interest is Transition Path Sampling (TPS), so it shall briefly be introduced here. The goal of TPS is to sample the so-called *transition path ensemble*, that is, the ensemble of all paths that connect two regions, usually (meta)stable states, in phase space. This is achieved by a random walk in path space, meaning an existing Transition Path (TP) is perturbed in order to produce the next candidate for a TP. By accepting or rejecting candidates with a suitable, algorithm specific criterion, one will sample the correct distribution of transition paths. The most common way of producing a candidate TP is the *shooting move* [11], which involves choosing a random point along the existing path and, depending on the underlying dynamics, applying a small perturbation to it. From this point, called the Shooting Point (SP), two new trajectories are started, propagated until they reach one of the target regions and subsequently stitched together to form the candidate path. While TPS enables efficient sampling of transitions related to rare events and is conceptually straightforward, it is not without shortcomings. Due to the nature of the shooting move, the algorithm is inherently sequential and, as a consequence, subsequently generated paths exhibit strong correlations, necessitating long simulation times depending on the system's complexity.

In the past few years, the field of machine learning has received unprecedented attention, not only from the scientific community but also from the general public, the most notable applications being text [12, 13] and image [14, 15] generation. Researchers have successfully applied machine learning in fields like chemistry [16, 17], discovery of new materials [18], biosciences [19] and various fields in physics, like high-energy physics [20, 21] and material and soft matter physics [22, 23]. Considering the constant increase in availability and performance of computational resources, the research interest in machine learning is likely to continue growing.

As could be expected, machine learning would eventually find recognition from the enhanced sampling community as a potentially invaluable tool to combat the sampling problem [24]. Starting in the mid-2000s with works by Ma and Dinner [25] as well as Peters and Trout [26], effort has been put into developing dimensionality reduction techniques, which can then be used in combination with existing enhanced sampling schemes. These include methods of finding and constructing suitable Collective Variables (CVs) from simulation data [25–28], which parametrize the relevant aspects of a system and can thus be used to bias simulations towards regions of interest, transfer operator-based approaches [29, 30], where the goal is to determine which eigenfunctions of a system's transfer operator are the slowest [24], and many more. Another use of machine learning in enhanced sampling is to try to optimally initialize [31–33] or place bias within [34–36] a simulation [24].

As it is possible to create text and images using generative machine learning models, the idea of learning a potentially complex probability distribution like the distribution of microscopic states in a molecular system (Boltzmann distribution) and drawing samples from it does not seem unreasonable. Over the years, various types of generative models like Generative Adversarial Networks (GANs) [37] or Variational Auto-Encoders

(VAEs) [38] have been proposed, although most have proven to be unsuitable for practical application [39].
First introduced by Tabak and Vanden-Eijnden [40] and Tabak and Turner [41], Normalizing Flows (NFs) are a class of generative models that can be used for a broad variety of tasks [39, 42–44]. They constitute an invertible transformation from an easy-to-sample prior distribution (e.g. a Gaussian) to an arbitrarily complex posterior distribution while allowing for efficient and exact density evaluation of samples by reversing the transformation [39]. Noé et al. [45] introduced Boltzmann Generators (BGs) as a type of Normalizing Flow specifically tailored to generating samples of condensed matter systems and protein structures. The promising feature of those samples is that they are independent from one another, eliminating the aforementioned problem of correlations.
Recently, Falkner et al. [46] demonstrated that it is possible to introduce a condition to the transformation performed by BGs, effectively steering the generation of samples towards regions of interest along some arbitrary (but known) coordinate. In their work, the authors utilized this to generate configurations on the top of an energy barrier separating two stable states, with the aim being to use those as shooting points associated with a high probability of producing reactive paths. Those paths can then be generated in parallel and are uncorrelated, resulting in an efficient sampling of the transition path ensemble and removing the restrictions imposed by TPS. However, for complex molecular systems, finding a suitable reaction coordinate is far from trivial as it can be a non-linear combination of many observables. This complicates the generation of shooting points and decreases the efficiency of the proposed algorithm.
Around the same time, Jung et al. [47] introduced a path sampling algorithm named Artificial Intelligence for Molecular Mechanism Discovery (AIMMD) that, while performing a standard TPS simulation, learns the committor $p_{\mathrm{B}}(x)$ of the system. The committor is then used to bias the shooting point selection towards the $p_{\mathrm{B}}(x) \simeq 0.5$ region. This results in more reactive path candidates and therefore increases the sampling efficiency. The committor $p_{\mathrm{B}}(x)$ is the probability of a trajectory initiated at some configuration x to enter a state B before entering any other state. This makes the committor the ideal reaction coordinate as it parametrizes the progress of a transition from one state to another [47].
This thesis aims to combine the committor learning scheme devised by Jung et al. [47] with the generative modelling based approach to Transition Path Sampling described by Falkner et al. [46]. As discussed above, the latter requires knowledge of a good reaction coordinate, which can be provided through AIMMD in the form of the committor. Training AIMMD, however, requires decorrelated transition paths, which can be efficiently obtained using conditioned Boltzmann Generators. This work sets out to resolve this circular dependence by developing an algorithm called Generative AIMMD (GenAIMMD) that self-consistently learns to generate shooting points in the high-efficiency region defined by the learned committor of a system. This would allow for parallelizable sampling of uncorrelated transition paths without prior knowledge of a suitable reaction coordinate.

2. Theory

2.1. Statistical treatment of physical systems

Statistical mechanics describes the emergence of macroscopic observables like temperature or pressure in terms of the statistical treatment of a system's microscopic quantities like particle positions and momenta. When studying a particular system, constraints are placed upon a subset of those macroscopic parameters while letting others fluctuate freely. This introduces the notion of statistical ensembles as a large number of copies of a system assuming all possible microstates compatible with the particular choice of fixed parameters [48]. For example, by fixing the total energy E, the number of particles N and the volume V of a system, one obtains the microcanonical or NVE ensemble. While being a physically valid choice, constraining E is not realizable in experiments [49]. By instead fixing the temperature T, one obtains the canonical or NVT ensemble, which is also the ensemble used throughout this work because of its popularity.
Each ensemble is associated with its own probability density function $\rho_{\text{ens}}(\mathbf{\Gamma})$, where $\mathbf{\Gamma}$ is the microstate of a system [1]. Therefore, the probability of observing a specific microstate is dictated by the choice of ensemble. For the overdamped dynamics in the canonical ensemble considered in this work, it is sufficient to describe a system of N particles through their positions r_i with $i = 1, \ldots, N$, integrating out momenta. For a state $r^N = \{r_1, r_2, \ldots, r_N\}$, the probability density function is given by

$$\rho_{NVT}(r^N) = \frac{1}{Z_{NVT}} e^{-\beta U(r^N)}, \tag{2.1}$$

where $\beta = 1/k_{\text{B}}T$ is the Boltzmann factor with Boltzmann constant k_{B}, U is the potential energy of the configuration and Z_{NVT} is the partition function, acting as a normalization factor [48]. This can be used to compute the expectation value of an arbitrary observable $A(r^N)$ as an ensemble average [1]

$$\langle A \rangle_{\text{ens}} = \int \mathrm{d}r^N \rho_{\text{ens}}(r^N) A(r^N), \tag{2.2}$$

where $\int \mathrm{d}r^N := \int \mathrm{d}r_1 \int \mathrm{d}r_2 \cdots \int \mathrm{d}r_N$.

2.2. Sampling from statistical ensembles

Equation 2.2 implicitly depends on the partition function Z, which includes an integral (or a sum, in case of discrete coordinates) over all possible microstates r^N. As a result, evaluating Z analytically is impossible but for a few simple toy systems. To circumvent this

problem using computer simulation, there exist two main methods: Molecular Dynamics (MD) and Monte Carlo (MC).

2.2.1. Molecular Dynamics

The core of Molecular Dynamics is to prepare a system in some initial configuration and to then propagate it in time by numerically solving known equations of motion in a step-by-step manner using discrete time steps [1]. Macroscopic observables A are then obtained using a time average of instantaneous measurements within the trajectory $\{r^N(t)\,|\,t = 1, \ldots, \tau\}$, i.e., [1]

$$\langle A \rangle_{\text{time}} = \frac{1}{\tau} \sum_{t=1}^{\tau} A(r^N(t)). \tag{2.3}$$

When doing MD, it is essential to choose suitable equations of motion to integrate in order to achieve the desired result. For example, when Hamilton's equations of motion are integrated without modification, the resulting samples are distributed according to the microcanonical or NVE ensemble [49]. In this work, the canonical ensemble is used, so the simulation is performed at a constant temperature. There are several ways to accomplish this, for example, by stochastic re-sampling of velocities [50] or, more popularly, by the introduction of extended phase space dynamics [51–53] [49].
Another approach is to use the Langevin equation of motion, which is stochastic by design. It combines deterministic Newtonian dynamics with the stochastic interactions between the system and a virtual heat bath in which it is submerged [1]. The Langevin equation for a system of N particles with time-dependent positions $r^N(t) = \{r_1(t), \ldots, r_N(t)\}$ is given by [49]

$$m\frac{d^2 r_i(t)}{dt^2} = F_i(r^N(t)) - \gamma m \frac{dr_i(t)}{dt} + \sqrt{\frac{2\gamma m}{\beta}}\eta(t), \tag{2.4}$$

where $F_i(r^N)$ is the Newtonian force acting on particle i, m is the particle's mass and γ the friction coefficient determining the interaction strength between the system and the bath. The last term of the equation is the random force, with $\eta(t)$ being a stochastic process satisfying $\langle \eta(0)\eta(t) \rangle = \delta(t)$.
A special case of Langevin dynamics, referred to as *Brownian dynamics* or *overdamped Langevin dynamics*, happens in the limit $\gamma \gg 0$. Here, the particle momenta are neglected, the physical interpretation being that any systematic velocities quickly decay due to friction. It has to be mentioned that, in doing this, one loses physical realism in the short-term dynamics [1]. A solution to the corresponding Langevin equation in discrete time steps Δt is then given by the particularly simple Euler-Maruyama method [1, 54]

$$r_{n+1} = r_n + \beta D \Delta t\, F(r_n^N) + \sqrt{2D\Delta t}\, R_n^G, \tag{2.5}$$

where $D = 1/\beta m\gamma$ is the diffusion coefficient and R_n^G is a sample from a normal distribution centred on zero with uniform variance and the index i of the particle was dropped. This method will be used throughout the entirety of this thesis whenever MD is performed.

2.2.2. Monte Carlo sampling

The term Monte Carlo (MC) in the context of computer simulation refers to algorithms that utilize random numbers in order to obtain numerical solutions to oftentimes complex problems or systems [55]. The Monte Carlo method has applications in a broad spectrum of disciplines. The MC algorithms used in the field of statistical mechanics, however, mostly fall into the sub-class of Markov Chain Monte Carlo (MCMC) algorithms. Here, one can sample an arbitrary distribution by constructing a stochastic process that converges to the desired distribution. An important property of that process is its lack of memory, meaning that, in a sequence of events $\{r_1^N, \ldots, r_{k-1}^N, r_k^N\}$, the probability of going from state r_k^N to r_{k+1}^N is independent of the other states $r_1^N, \ldots, r_{k-1}^N$ in the sequence. Such a process is called a *Markov chain*. [1]
In contrast to MD, the dynamics generated by an MCMC algorithm have no physical meaning. Nevertheless, such algorithms can be of great use, for example to do statistics in the canonical ensemble without having to compute the value of the partition function Z_{NVT}. By choosing the canonical distribution ρ_{NVT} as the target distribution, one can draw samples $\{r_i^N \,|\, i = 1, \ldots, k\}$ from it and use them, for example, to compute expectation values of observables $A(r^N)$ using [1]

$$\langle A(r^N)\rangle_{\text{ens}} := \langle A(r^N)\rangle_{r^N \sim \rho_{NVT}(r^N)} = \frac{1}{k}\sum_{i=1}^{k} A(r_i^N). \tag{2.6}$$

One such MCMC algorithm is the Metropolis algorithm [56], which involves generating a new state y from the current state x and accepting or rejecting the move with a certain acceptance probability $p_{\text{acc}}(x \to y)$. If the move is accepted, y becomes the next state in the Markov chain, otherwise x is counted again [48]. To ensure convergence to the stationary target distribution, the acceptance probability must fulfil a condition called *detailed balance*, which, assuming a symmetric probability to generate a move, reads [48]

$$\rho(x) \cdot p_{\text{acc}}(x \to y) = \rho(y) \cdot p_{\text{acc}}(y \to x), \tag{2.7}$$

which can be rearranged to obtain

$$\frac{p_{\text{acc}}(x \to y)}{p_{\text{acc}}(y \to x)} = \frac{\rho(y)}{\rho(x)}. \tag{2.8}$$

The above condition can be satisfied using the Metropolis criterion [56]

$$p_{\text{acc}}(x \to y) = \min\left[1, \frac{\rho(y)}{\rho(x)}\right]. \tag{2.9}$$

Assuming that the target distribution is the canonical distribution ρ_{NVT} (2.1), equation 2.9 can be written as

$$\begin{aligned} p_{\text{acc}}(x \to y) &= \min\left[1, \frac{\rho_{NVT}(y)}{\rho_{NVT}(x)}\right] = \min\left[1, \frac{\frac{1}{Z}e^{-\beta U(y)}}{\frac{1}{Z}e^{-\beta U(x)}}\right] = \min\left[1, e^{-\beta \Delta U}\right] \\ &= \begin{cases} e^{-\beta \Delta U} & \text{if } \Delta U > 0 \\ 1 & \text{otherwise} \end{cases} \end{aligned} \tag{2.10}$$

where $\Delta U = U(y) - U(x)$ is the difference in potential energy caused by the move. Therefore, each move that lowers the potential energy is accepted immediately. It should be noted that in equation 2.10, the partition function Z cancels, which is important since we cannot compute its value in most cases as discussed above.

2.3. Enhanced sampling

Both sampling methods, MD and MCMC, move through a system's energy landscape by iteratively updating the positions of a walker. Complex systems contain numerous (meta)stable states that are characterized by minima in which the system preferentially dwells during an equilibrium simulation. Transitions from one minimum to another require the crossing of energetic or entropic barriers, events that become exceedingly rare with increasing barrier heights as they purely rely on spontaneous thermal fluctuations. As a result, the waiting times between the spontaneous occurrence of such *rare events* can be drastic, which is problematic if one is interested in studying the behaviour of a system out of equilibrium. Therefore, the need arises for so-called *enhanced sampling* techniques that steer the system to visit unlikely regions of configuration space whilst still retaining consistency with the equilibrium distribution one aims to sample. Two such techniques, namely Umbrella Sampling (US) and Transition Path Sampling (TPS) shall now be introduced.

2.3.1. Umbrella Sampling

The idea of Umbrella Sampling [5] is to shift the focus of the sampling from the stable states to a number of pre-defined windows which typically lie at regular intervals between the states. This is done by modifying the potential energy function $U(r^N)$ of the system by introducing a bias potential $U_B^{\hat{\xi}_i}(r^N)$ of variable strength, which can be tuned to be positioned at regions of interest in phase space [57].
The windows are defined using a so-called Reaction Coordinate (RC) ξ, which parametrizes the progress of a transition between two states. It can be an arbitrary function of (parts of) the system's coordinates r^N and is oftentimes related to the system's geometry [57]. The bias centre corresponding to the window i is then denoted by $\hat{\xi}_i$. This results in a modified expression for the system's potential energy of the form

$$U_i'(r^N) = U(r^N) + U_B^{\hat{\xi}_i}(r^N). \tag{2.11}$$

In this work, a harmonic bias potential

$$U_B^{\hat{\xi}_i}(r^N) = \frac{k_{\text{bias}}}{2}\left[\xi(r^N) - \hat{\xi}_i\right]^2 \tag{2.12}$$

is used, where the parameter k_{bias} determines the strength of the bias.
A typical application of Umbrella Sampling is free energy estimation along a reaction coordinate. In the canonical ensemble, the Helmoltz free energy can be calculated

using [48]

$$F = -\frac{1}{\beta} \ln Z_{NVT}. \tag{2.13}$$

As discussed above, direct calculation of Z_{NVT} is impossible. Therefore, the free energy $F(\xi)$ along the RC is estimated using a histogram of the values of ξ observed during simulation. Using Umbrella Sampling, each simulation in each window produces one such histogram, which is biased by the potential U_B [57]. Since the bias potential solely depends on ξ and its specific form is known, one can correct for it and combine the histograms in order to retrieve the full and unbiased $F(\xi)$.
A popular method of achieving this is the Weighted Histogram Analysis Method (WHAM) [58]. Without going into detail, the central idea of WHAM is to minimize the statistical error of the combined and unbiased histograms [57] by solving the WHAM equations self-consistently. In order for this to work, the individual histograms need to partially overlap [57].

2.3.2. Transition Path Sampling

Umbrella Sampling is a powerful technique, since it enables low-probability regions of configuration space to be sampled with high efficiency. These samples can then be used to estimate stationary quantities, for example the height of a free energy barrier. However, since US relies on performing simulations using a modified (biased) potential, the true and unbiased dynamics of the system are lost and cannot be recovered. Transition Path Sampling (TPS) [59] is another enhanced sampling technique and it enables one to study transitions between stable states that happen rarely [60]. Not only does it preserve the dynamics of the unbiased system, which is important for example if one wishes to estimate reaction rate constants, but it also requires no prior knowledge of a reaction coordinate [60].
In the above description of MCMC methods, it was briefly mentioned that they can be used to sample from arbitrary distributions. The canonical distribution was then presented as a prominent example and discussed in more detail, which might give the impression that such methods are exclusively concerned with sampling distributions of microstates of molecular or atomistic systems. This, however, is not the case, and TPS is an example of such a MCMC method.
In TPS, the constituents of the Markov chain are not individual microstates but rather dynamical trajectories of the system [61]. The aim is to sample from the so-called transition path ensemble, i.e., the ensemble of all trajectories connecting two stable states A and B, a visualization of which can be seen in figure 2.1. These trajectories $X(\tau)$ can either have fixed or variable lengths τ, the latter case being considered in this work. They consist of points x_i with $i \in \{0, 1, \ldots, \tau/\Delta t\}$ where Δt is the timestep, such that $X(\tau) = \{x_0, x_1, \ldots, x_{\tau/\Delta t}\}$ [62]. Their probability density can be expressed as [62]

$$P_{\mathrm{AB}}[X(\tau)] = \frac{1}{Z_{\mathrm{AB}}} H_{\mathrm{AB}}(x_0, x_{\tau/\Delta t})\, \tilde{H}_{\mathrm{AB}}[X(\tau)]\, \rho(x_0) \prod_{i=0}^{\tau/\Delta t - 1} p_{\mathrm{trans}}(x_i \to x_{i+1}). \tag{2.14}$$

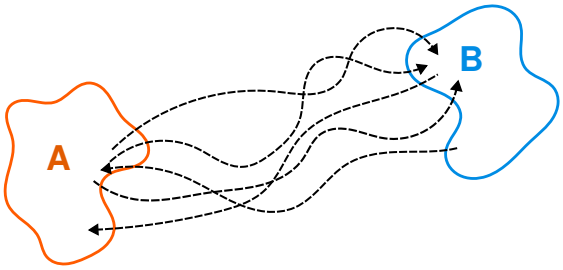


Figure 2.1.: Visualization of the Transition Path Ensemble, which is the ensemble of all paths connecting two stable states A and B.

Here, Z_{AB} is a normalization factor and $p_{\mathrm{trans}}(x \to y)$ is the short-time probability of transitioning from point x to point y. The function

$$H_{\mathrm{AB}}(x,y) = \begin{cases} 1 & \text{if } \; h_{\mathrm{A}}(x)h_{\mathrm{B}}(y) = 1 \; \vee \; h_{\mathrm{A}}(y)h_{\mathrm{B}}(x) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

ensures that only paths that start in A and end in B and vice-versa (reactive paths) are considered, where

$$h_{\mathrm{S}}(x) = \begin{cases} 1 & \text{if } \; x \; \text{ is in state S} \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

is the population function [62]. Similarly, the function

$$\tilde{H}_{\mathrm{AB}}[X(\tau)] = \prod_{i=1}^{\tau/\Delta t - 1} \tilde{h}(x_i) \tag{2.17}$$

with

$$\tilde{h}(x) = \begin{cases} 0 & \text{if } \; x \; \text{ is in state A or B} \\ 1 & \text{otherwise} \end{cases} \tag{2.18}$$

weights out any trajectories that have intermediate points inside A or B [62]. With these constraints in place, the only paths of interest are those which connect the two states and have exactly one point (the first/last point) inside each of them.
An important decision to make is the definition of the states A and B. This is usually done by introducing a so-called *order parameter* $q_{\mathrm{order}}(x)$, which is a low-dimensional function capable of characterizing whether a point x is inside state A, B or neither of them [60]. Unlike a reaction coordinate, the order parameter does not need to be able to parametrize the progress of the transition between A and B, but it has to fulfil other important criteria. For example, it must be chosen generous enough to allow for equilibrium fluctuations within A and B but strict enough for the basins of attraction of A and B not to overlap [60].
With these definitions, it is now possible to construct a random walk in path space that samples the distribution of reactive paths defined in equation 2.14. This is done following

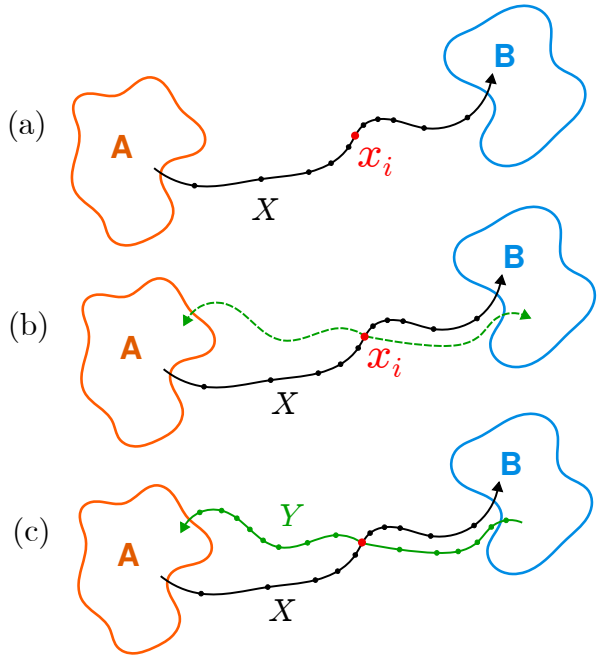


Figure 2.2.: Overview of the shooting algorithm to produce a reactive path candidate for Transition Path Sampling. In (a), a point x_i is chosen from an existing path X at random. In case of deterministic dynamics, a small perturbation is applied to it, and it is then called the Shooting Point. In (b), two new trajectories (green) are propagated from the shooting point, one forward and one backward in time. Lastly, in (c), if the two trajectories each reach a different state A or B, they are connected and thus form the new path candidate Y.

the typical MCMC procedure. First, a new trajectory is generated from the current one. Next, using a suitable acceptance probability, the candidate is either accepted or rejected and becomes the next link in the Markov chain in case of acceptance. Otherwise, the current path is counted again [61].
The most common way of generating a new path given a reactive path is the *shooting move* [11], which is summarized in figure 2.2. Along the current trajectory, a configuration x_i is chosen at random, which is further called the Shooting Point (SP). From this point, two new trajectories are started and propagated, one forward and one backward in time, until they reach either state A or B. The candidate is then constructed by stitching together the two trajectories. Importantly, if the system is propagated using deterministic dynamics (e.g. by integrating Newton's equations of motion), the shooting point needs to be perturbed prior to starting the new trajectories, otherwise one will produce the same path as before. Since the dynamics used in this work are purely stochastic, the perturbation of the shooting point is not necessary. Also, if momenta are considered in the dynamics, they need to be inverted in the trajectory that is propagated backwards in

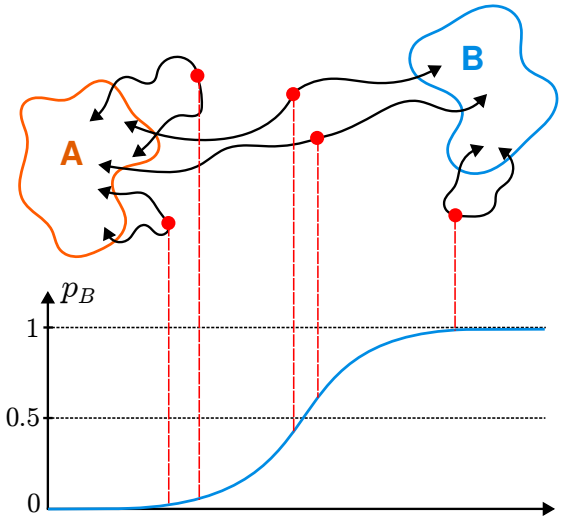


Figure 2.3.: Visualization of the definition of the committor $p_{\mathrm{B}}(x)$ of a system with two stable states A and B. The red dots indicate shooting points and the black lines fleeting trajectories started from those shooting points. Trajectories started close to a stable state have a high likelihood of first reaching that state, meaning that the starting configuration is associated with a committor close to zero or one.

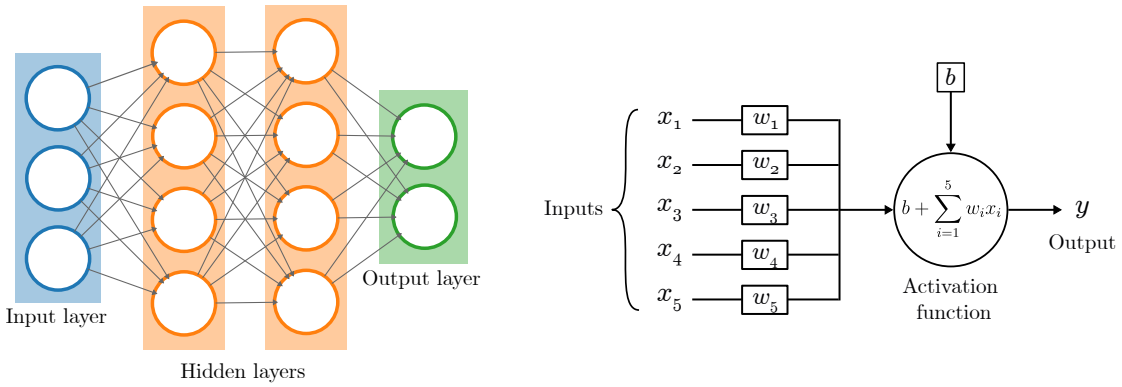
time before constructing the new path.
Once a candidate Y is obtained from the current path X, it has to be either accepted or rejected while satisfying detailed balance. This can again be done using the Metropolis rule, which results in the general acceptance probability

$$P_{\mathrm{acc}}(X \to Y) = \min\left[1, \frac{P_{\mathrm{AB}}(Y) P_{\mathrm{gen}}(Y \to X)}{P_{\mathrm{AB}}(X) P_{\mathrm{gen}}(X \to Y)}\right], \tag{2.19}$$

where $P_{\mathrm{gen}}(X \to Y)$ is the probability of generating the move $X \to Y$ (and analogously $P_{\mathrm{gen}}(Y \to X)$) [60]. Assuming that the probability of choosing a shooting point along the current path X is uniform and the forward and backward segments are generated with the same dynamics as the current path, the acceptance probability reduces to [62]

$$P_{\mathrm{acc}}(X \to Y) = H_{\mathrm{AB}}(Y) \min\left[1, \frac{L(X)}{L(Y)}\right], \tag{2.20}$$

where $L(X)$, $L(Y)$ are the path lengths of X and Y respectively and $H_{\mathrm{AB}}(Y)$ is a short-hand notation for the function H_{AB} evaluated at the first and last point in Y [63]. An important quantity in relation to TPS is the committor $p_{\mathrm{B}}(x)$, which is the probability of a trajectory started at some point x to enter state B before state A [60], as can be seen in figure 2.3. One can analogously define the committor $p_{\mathrm{A}}(x)$, however, for ergodic and overdamped systems considered in this work, $p_{\mathrm{A}}(x) + p_{\mathrm{B}}(x) = 1$, so knowledge of



(a) (b)

Figure 2.4.: The typical structure of a Multilayer Perceptron (a) and a schematic depiction of an artificial neuron (b). Each circle in (a) is such an artificial neuron.

$p_B(x)$ implies knowledge of $p_A(x)$. Intuitively, shooting points with a committor close or equal to 0.5 are most likely to result in reactive paths, since the two trajectories started from them have a high chance of ending up in different states. This can also be seen from the relation

$$p(\mathrm{TP}|x) = 2p_B(x)(1 - p_B(x)), \tag{2.21}$$

which holds for Markovian and time-reversible dynamics like Brownian dynamics and describes the probability $p(\mathrm{TP}|x)$ of a transition path passing through point x [47], which clearly has its maximum at $p_B(x) = 0.5$. Since, during a transition between states A and B, the committor gradually changes from 0 to 1 or from 1 to 0 depending on the direction, it constitutes an ideal reaction coordinate [60], which will be important later on.

2.4. Artificial Neural Networks

Artificial Neural Networks (ANNs) are mathematical models that aim to replicate the working principle of biological nervous systems in order to solve various data-related problems like classification, where input data is to be categorized based on pre-defined conditions, or regression, where the goal is the approximation of some arbitrary function [64]. There are various types of ANNs, including feed-forward networks, through which data is fed only in one direction, and recurrent networks, which allow the data to flow in both directions within the network, feeding outputs back to parts of the network as inputs [64].

2.4.1. Multilayer Perceptrons

An important type of feed-forward network is the Multilayer Perceptron (MLP), which was originally proposed by Frank Rosenblatt in 1958 [65]. It consists of so-called artificial

neurons that are organized in layers, the first layer being the *input layer* and the last layer being the *output layer*. In between, there is an arbitrary number of *hidden layers*, which, combined with the varying or constant number of neurons in each layer, defines the complexity of the network. The structure of a typical MLP can be seen on the left-hand side of figure 2.4. MLPs have the property of being fully connected, meaning that each neuron receives the outputs of all neurons of the previous layer as an input and, conversely, sends its output to each neuron of the following layer [66].
At their core, neurons are linear functions

$$x_i^l = b_i^l + \sum_{j=1}^{N_{l-1}} W_{ij}^l y_j^{l-1} \tag{2.22}$$

of the N_{l-1} outputs y_j^{l-1} of the previous layer $l-1$ to which one applies a non-linear function $F_{\text{act}}(x)$ called *activation function* to produce the outputs of the current layer l

$$y_i^l = F_{\text{act}}(x_i^l). \tag{2.23}$$

Equation 2.22 can be written in matrix form as

$$\vec{x}^l = \vec{b}^l + W^l \vec{y}^{l-1}, \tag{2.24}$$

where $\vec{y}^{l-1}$ is a vector of the outputs of all neurons of layer $l-1$, $W^l \in \mathcal{M}_{N_l \times N_{l-1}}(\mathbb{R})$ is a matrix containing the so-called weights of the connections between layer $l-1$ and l and $b^l \in \mathbb{R}^{N_l}$ is a vector of the so-called biases. During training, these weights and biases are optimized such that the network learns to approximate any [67] smooth and measurable function of arbitrary complexity [66]. A depiction of a neuron can be seen on the right-hand side of figure 2.4.

Activation functions

Activation functions play the important role of enabling non-linear regression with neural networks[66]. Without them, the entire network would amount to a linear function due to the fact that compositions of linear functions are themselves linear. This would make it impossible to approximate non-linear functions. The choice of activation function depends on the problem at hand, and each layer may have a different activation function. Popular examples that are also used in this thesis are the Rectified Linear Unit (ReLU)

$$\text{ReLU}(x) = \max[0, x] = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0, \end{cases} \tag{2.25}$$

the $\tanh(x)$ function and the standard logistic function

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.26}$$

the latter one typically being used as the activation function of the last layer if the network outputs are probabilities, requiring them to be in $[0, 1]$.

Training

Training of MLPs is done in a supervised way [66], meaning that the input data used for training needs to be equipped with the expected outputs, the so-called *labels*. During the training process, the network parameters (weights and biases) are then optimized such that a so-called *loss function* $L(\vec{y}, \widehat{\vec{y}})$ is minimized, where $\vec{y}$ is the output produced by the network and $\widehat{\vec{y}}$ the expected output, with $\vec{y}, \widehat{\vec{y}} \in \mathbb{R}^M$. An example for such a loss function is the Mean Squared Error (MSE) function

$$L_{\text{MSE}}(\vec{y}, \widehat{\vec{y}}) = \frac{1}{M} \sum_{i=1}^{M} \left| \vec{y} - \widehat{\vec{y}} \right|^2 . \tag{2.27}$$

Since $\vec{y}$ is obtained by passing an input vector through the network, it is actually a function of the input and all the network's weights and biases. Therefore, to efficiently minimize the loss function, one requires information about how each parameter contributes to the output. This information is provided by the gradient of the loss function $\partial L / \partial \theta_i$ with respect to each network parameter θ_i. In practice, these gradients are obtained using the backpropagation algorithm [68], which iterates backwards through the network layers computing gradients along the way using the chain rule [66].

Optimizers

Computing the gradients is only the first step towards minimizing the loss function. The crucial second step is to use these gradients to update the network parameters efficiently, meaning fast convergence at low computational cost. There are multiple algorithms called *optimizers* that aim to achieve this. The most widely used optimizers are based on the Gradient Descent algorithm, which updates the parameters along the negative gradient in small steps

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta L(\theta_{t-1}) \tag{2.28}$$

dictated by the learning rate η [69]. Examples of optimizers based on this idea include the momentum method [68], Adagrad [70], Adadelta [71] and Adam [72], the latter one being used throughout this work.
Adam, short for Adaptive Moment Estimation, improves upon the standard Gradient Descent method (equation 2.28) by implementing adaptive per-parameter learning rates, meaning that each parameter is updated with a different step size according to its importance [69]. It achieves this by including an exponentially decaying average of estimates of the gradients' first and second moments in the update rule for the network parameters, which helps with convergence [69, 72].

Batches

Another important consideration that needs to be made during training is the number of samples used in each parameter update step. By passing all available training samples through the network, computing the gradients and using them to update the parameters,

one follows the slope of the loss function very closely, achieving smooth convergence to the minimum. For large datasets, however, this approach is not feasible due to memory constraints and slow convergence [69]. The other extreme is called *Stochastic Gradient Descent*. Here, the parameters are updated for each sample in the training set, leading to large fluctuations of the training loss [69]. While the convergence is typically faster, this method is prone to overshooting of the minimum if the learning rate is kept constant [69]. The most common approach lies between the two extremes and is called *Mini-batch Gradient Descent*. The training set is partitioned into batches, the number of samples per batch being referred to as the *batch size*. Parameter updates are performed using the gradients produced by each batch, which smoothens the convergence while still being time- and memory-efficient [69]. In general, one also shuffles the training set each time the entire set has passed through the network, which is called an *epoch*, in order to avoid biasing the optimizer towards a specific order of the samples [69].

2.4.2. Artificial Intelligence for Molecular Mechanism Discovery

Neural networks are widely used in physics today, harnessing their expressiveness when applied to regression tasks. An example of such a task that is relevant to this thesis is learning and subsequently predicting the committor of a system, which is analytically intractable. The committor is of particular interest as it is a central quantity in optimizing path sampling.

Artificial Intelligence for Molecular Mechanism Discovery (AIMMD) is an algorithm proposed by Jung et al. [47] that trains an artificial neural network to predict the committor $p_{\mathrm{B}}(x)$ of a given point x in order to bias the shooting point selection during a TPS simulation towards the $p_{\mathrm{B}}(x) \simeq 0.5$ region, which, as discussed above, maximizes the likelihood of producing reactive path candidates.

They start by modelling the committor as a standard logistic function applied to the output $q(x|\theta)$ of an MLP, which takes a configuration x of the system as an input, i.e.,

$$p_{\mathrm{B}}(x) = \frac{1}{1 + e^{-q(x|\theta)}}. \tag{2.29}$$

This network is trained using the outcomes $s = \{s_i\}_{i=1\ldots k}$ of k shooting moves from shooting points $x = \{x_i\}_{i=1\ldots k}$, where $s_i = -1$ if the trajectory started at x_i first enters state B and $s_i = 1$ if it first enters state A. In the case of overdamped dynamics considered in this work, each shooting attempt constitutes a Bernoulli experiment with the probability of observing outcome $s_i \in \{-1, 1\}$ given by

$$p(s_i|x_i) = \begin{cases} p_{\mathrm{B}}(x_i) & \text{if } s_i = -1 \\ 1 - p_{\mathrm{B}}(x_i) & \text{if } s_i = 1. \end{cases} \tag{2.30}$$

Repeating this experiment for k shooting points and combining the probabilities, one obtains the likelihood

$$\mathcal{L} = \prod_{i=1}^{k} p(s_i|x_i). \tag{2.31}$$

After inserting equation 2.29 for $p(s_i|x_i)$, this likelihood depends on the output of the network. The objective of the training process is then to maximize $\mathcal{L}$, which is equivalent to minimizing the loss function

$$L_{\text{AIMMD}}(\theta|x,s) = -\log \mathcal{L} = \sum_{i=1}^{k} \log\left(1 + e^{s_i q(x_i|\theta)}\right). \tag{2.32}$$

In their paper, Jung et al. [47] train the network in parallel to a TPS simulation, picking shooting points according to a Lorentzian distribution centred at $q(x|\theta) = 0$ and using the results of the two-way shooting attempts to extend the training set for the network. In order to measure the performance of the committor model, the authors define an efficiency factor

$$\alpha_{\text{eff}} = \min\left[1, \left(1 - \frac{n_{\text{gen}}}{n_{\text{exp}}}\right)^2\right] \tag{2.33}$$

with n_{gen} being the number of reactive paths generated from a set of k shooting points $x = \{x_i\}_{i=1\ldots k}$ and

$$n_{\text{exp}} = \sum_{i=1}^{k} 2(1 - p_{\text{B}}(x_i))p_{\text{B}}(x_i) \tag{2.34}$$

being the number of expected reactive paths from the same set of shooting points according to the committors predicted by the network. During training, α_{eff} should converge to zero, indicating perfect agreement between predicted and expected transitions.
While the definition of the committor model and the loss function are unchanged throughout this thesis, the shooting points and outcomes are being obtained by other means.

2.5. Normalizing Flows

In recent years, the possibility of using machine learning to approximate an arbitrary probability distribution by training solely on samples drawn from it has been explored [39], notable examples being Variational Auto-Encoders (VAEs) [38] and Generative Adversarial Networks (GANs) [37]. Once sufficiently trained, such models can be used to produce new and before unseen samples of the learned distribution, hence their classification as *generative models* [39]. However, the aforementioned models provide no easy way of obtaining the probability density of the samples they produce.
Normalizing Flows [40, 41] enable both producing samples and obtaining probability densities in an exact and computationally efficient way and are yet another example of a family of generative models [39]. There are various types of Normalizing Flows, namely Elementwise, Linear, Planar, Radial, Coupling, Autoregressive and Residual Flows [39], but they share the same fundamental idea.
In general, a Normalizing Flow constitutes a diffeomorphism $F_{zx} : \mathbb{R}^D \to \mathbb{R}^D$ between the sample spaces of a tractable and easy-to-sample probability distribution $p_Z(z) : \mathbb{R}^D \to \mathbb{R}$ called *latent* or *prior distribution* and a potentially complex *target distribution* $p_X(x) : \mathbb{R}^D \to \mathbb{R}$, with random variables $z, x \in \mathbb{R}^D$ [39, 45]. The inverse mapping is denoted by

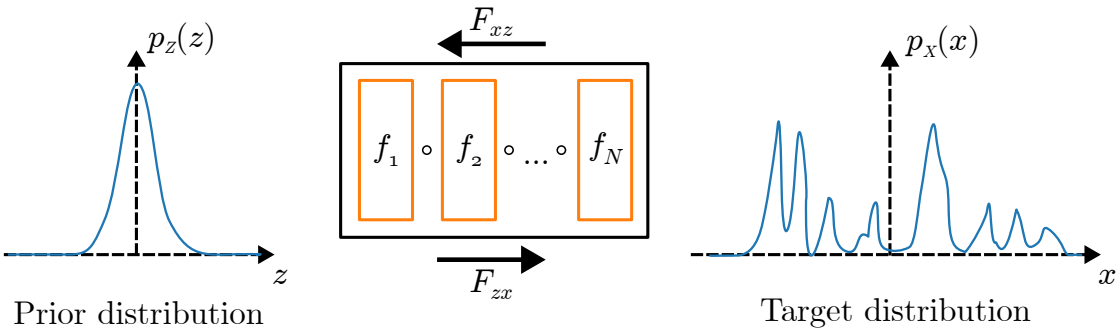


Figure 2.5.: Schematic depiction of a typical Normalizing Flow architecture. An invertible transformation between a prior and a target probability distribution is learned, which consists of a composition of multiple smaller transformations that are themselves invertible.

$F_{xz} := F_{zx}^{-1}$. A typical choice of latent distribution is the multivariate standard normal distribution [39] with density function

$$p_Z(z) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}||z||^2}, \tag{2.35}$$

where $||.||$ denotes the Euclidean norm on $\mathbb{R}^D$. The mappings F_{zx} and F_{xz} have Jacobian matrices [45]

$$J_{zx}(z) = \left[\frac{\partial F_{zx}(z)}{\partial z_1}, \ldots, \frac{\partial F_{zx}(z)}{\partial z_D}\right] \tag{2.36}$$

$$J_{xz}(x) = \left[\frac{\partial F_{xz}(x)}{\partial x_1}, \ldots, \frac{\partial F_{xz}(x)}{\partial x_D}\right], \tag{2.37}$$

which appear in the change of variable formula [45]

$$p_X(x) = p_Z(z) \left|\det J_{zx}(z)\right|^{-1} = p_Z(F_{xz}(x)) \left|\det J_{xz}(x)\right| \tag{2.38}$$

$$p_Z(z) = p_X(x) \left|\det J_{xz}(x)\right|^{-1} = p_X(F_{zx}(z)) \left|\det J_{zx}(z)\right| \tag{2.39}$$

to transform between the two distributions [39]. In principle, the functions F_{zx} and F_{xz} can be made as complex as necessary in order to successfully model the target distribution. This is typically done by considering compositions

$$F_{zx} = f_{zx}^N \circ f_{zx}^{N-1} \circ \cdots \circ f_{zx}^1 \tag{2.40}$$

$$F_{xz} = f_{xz}^1 \circ f_{xz}^2 \circ \cdots \circ f_{xz}^N \tag{2.41}$$

of N less complex, non-linear invertible functions, which are themselves invertible with

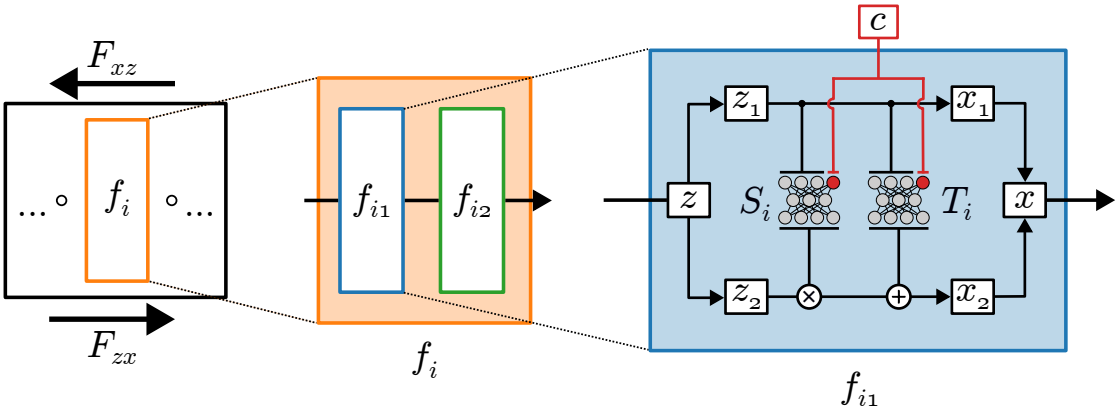


Figure 2.6.: Schematic depiction of the Real NVP architecture used in (conditioned) Boltzmann Generators. Left: Multiple Real NVP blocks $\{f_i\}_{i=1,\dots,N}$ making up the transformation F_{zx} and its inverse F_{xz}, as depicted in figure 2.5. Middle: A single Real NVP block f_i consisting of two affine coupling layers f_{i1} and f_{i2}, where the input channels are mirrored for f_{i2} compared to f_{i1}. Right: A single affine coupling layer f_{i1} in the zx direction with scaling and translation operations S_i and T_i modelled as multilayer perceptrons. The input c to S_i and T_i in red denotes an optional condition vector used for biasing the generator.

Jacobian determinants given by [39]

$$\det J_{zx}(z) = \prod_{i=1}^{N} \det J^i_{zx}(z) \tag{2.42}$$

$$\det J_{xz}(x) = \prod_{i=1}^{N} \det J^i_{xz}(x), \tag{2.43}$$

where $J^i_{zx}(z)$, $J^i_{xz}(x)$ are the Jacobian matrices of the i-th functions in the compositions. By carefully constructing the $\{f^i_{zx}\}, \{f^i_{xz}\}$ to have Jacobians that are cheap to compute, sample generation and density evaluation can be done very efficiently. This approach is schematically depicted in figure 2.5.

2.5.1. Real NVP architecture

In their 2017 paper, Dinh et al. [42] introduce a coupling flow [73] architecture capable of working with high-dimensional data while still being efficiently trainable. It uses real-valued non-volume preserving transformations, giving it the name *Real NVP*. An overview of this architecture can be seen in figure 2.6.
Extending their previous work [73], the $\{f^i_{xz}\}, \{f^i_{zx}\}$ now take the form of *affine coupling layers*, which are easily invertible and have Jacobians that are computationally inexpensive

to calculate. The input in either direction is each partitioned into two channels $x = (x_1, x_2)$, $z = (z_1, z_2)$ [42, 45]. The transformation is then given by [42, 45]

$$f^i_{xz}(x_1, x_2): \quad z_1 = x_1, \quad z_2 = x_2 \odot \exp\left[S_i(x_1, \theta_i)\right] + T_i(x_1, \theta_i) \tag{2.44}$$

$$f^i_{zx}(z_1, z_2): \quad x_1 = z_1, \quad x_2 = \left[z_2 - T_i(x_1, \theta_i)\right] \odot \exp\left[-S_i(z_1, \theta_i)\right], \tag{2.45}$$

where $\odot$ denotes the element-wise product and the S_i and T_i are modelled as artificial neural networks responsible for scaling and translating the input respectively and do not need to be invertible for the entire transformation to be invertible. In order to avoid the first channel of the input to remain unchanged by the transformation, the channels are swapped after each affine coupling layer [42, 45], thus defining a Real NVP block as two consecutive affine coupling layers in which all components are transformed [45].
The Jacobian matrices $J^i_{xz}(x_1, x_2)$, $J^i_{zx}(z_1, z_2)$ of these transformations are triangular [42], meaning that their determinants can be computed as [45]

$$\det J^i_{xz}(x) = \exp\left[\sum_j \left(S_i(x_1, \theta_i)\right)_j\right] \tag{2.46}$$

$$\det J^i_{zx}(z) = \exp\left[-\sum_j \left(S_i(z_1, \theta_i)\right)_j\right], \tag{2.47}$$

which is computationally inexpensive.

2.5.2. Boltzmann Generators

Boltzmann Generators, first introduced by Noé et al. [45], are generative models built upon the Real NVP architecture in figure 2.6 that specifically have a Boltzmann distribution $e^{-u(x)}$ with a general energy function $u(x)$ as their target distribution. Once sufficiently trained, they can be used to draw statistically independent samples with high weight in the target distribution. These samples can then be re-weighted in order to exactly follow the Boltzmann distribution of interest.
In the following, the exact distributions are denoted by $p_X(x)$ and $p_Z(z)$ respectively while the distributions resulting from the transformations $F_{xz}(x)$, $F_{zx}(z)$ are written as $q_X(x)$ and $q_Z(z)$, i.e., [45]

$$p_Z(z) \overset{F_{zx}}{\to} q_X(x) \tag{2.48}$$

$$p_X(x) \overset{F_{xz}}{\to} q_Z(z). \tag{2.49}$$

As in the original work by Noé et al [45], in this thesis, $p_Z(z)$ is chosen to be a multivariate normal distribution

$$p_Z(z) \propto \exp\left[-\frac{||z||^2}{2T^2}\right], \tag{2.50}$$

with standard deviation given by the temperature T relative to some reference temperature set to $1/k_\mathrm{B}$. The target distribution $p_X(x)$ is the Boltzmann distribution of the canonical ensemble

$$p_X(x) \propto \exp\left[-\beta U(x)\right], \tag{2.51}$$

where $U(x)$ is the potential energy of the configuration x.

Training

At the beginning of this section, Normalizing Flows were introduced as a means to learn an arbitrary distribution given samples from that distribution. This approach does not require the target distribution to be known beforehand, which is essential if one, for example, wishes to use Normalizing Flows to generate images or speech. In the case of Boltzmann Generators, this limitation vanishes, since the target distribution is known analytically and the probability density of samples can be evaluated exactly. However, samples from that distribution are costly to obtain, which is why one combines the classic *training by example* with a different strategy called *training by energy*.
During training by energy, samples are taken from the prior distribution $p_Z(z)$ and transformed using $F_{zx}(z)$ to the distribution $q_X(x)$. One would then like to compare this distribution to the target distribution $p_X(x)$ in order to update the model parameters such that the difference between the two distributions is minimized. This difference is quantitively given by the Kullback–Leibler (KL) divergence, which can be written as a loss function [45]

$$L_\mathrm{KL} = \mathbb{E}_{z\sim p_Z(z)}\left[\beta U(F_{zx}(z)) - \log(|\det J_{zx}(z)|)\right] \tag{2.52}$$

to be minimized during training.
Training by energy by itself is not mode-covering [45], meaning that the generator preferentially samples within the highest probability region of the target distribution while neglecting other important regions. To mitigate this, training by example is done in addition to training by energy. The idea is to maximize the likelihood of samples taken from the distribution $p_X(x)$ in the distribution $q_Z(z)$, similar to how other machine learning models are trained [45]. These samples can be obtained through standard (enhanced) sampling techniques like MD or MCMC. Also using the KL divergence, one arrives at a maximum likelihood (ML) loss function [45]

$$L_\mathrm{ML} = \mathbb{E}_{x\sim p_X(x)}\left[\frac{1}{T^2}||F_{xz}(x)||^2 - \log(|\det J_{xz}(x)|)\right], \tag{2.53}$$

where $p_X(x)$ is approximated as described above. The two loss functions are then combined into a single loss function [45]

$$L_\mathrm{BG} = \lambda_\mathrm{KL} L_\mathrm{KL} + \lambda_\mathrm{ML} L_\mathrm{ML} \tag{2.54}$$

with variable contributions from training by energy and training by example, their strength determined by λ_KL, λ_ML respectively.

Re-weighting

A trained Boltzmann generator produces configurations that, although having a high Boltzmann weight, are not exactly distributed according to the Boltzmann distribution. The reason for that is that the $q_X(x)$ merely approximates the $p_X(x)$. However, one can come up with a re-weighting factor [45]

$$w_X(x) = \frac{p_X(x)}{q_X(x)} \propto \exp\left[\beta U(F_{zx}(z)) - \log p_Z(z) + \log(|\det J_{zx}(z)|)\right] \tag{2.55}$$

that corrects for this.

Conditioning

In some cases, one might be interested in introducing a variable bias to the generation of samples in order to preferentially sample certain regions of interest. Falkner et al. [46] build on this idea and propose a path sampling scheme that does not select shooting points from existing paths but rather uses a Boltzmann Generator to generate them. This approach solves two of the main issues with the classic shooting algorithm, namely that it is inherently sequential and that the paths it produces are correlated. Generating shooting points using a Boltzmann Generator makes them and consequently the paths started from them statistically independent from one another. Also, since multiple shooting points can be obtained simultaneously, the process of generating trajectories can be completely parallelized.
One has to be careful though, since the paths resulting from this procedure are not necessarily distributed according to the transition path ensemble. To resolve this, a re-weighting procedure can be used, where each reactive path $X(\tau)$ receives a statistical weight [46]

$$\Omega[X(\tau)] \propto \left[\sum_{i=1}^{\tau/\Delta t} \frac{\rho_{\mathrm{SP}}(x_i)}{\rho(x_i)}\right]^{-1}. \tag{2.56}$$

Here, the $\rho_{\mathrm{SP}}(x)$ is the distribution of the shooting points.
As would be expected, training the Boltzmann Generator using the unmodified potential energy of a system results in the majority of the generated configurations to be within or close to the stable states. Such configurations, however, are unsuitable as shooting points, since they will most likely not result in reactive paths when propagated. To solve this, Falkner et al. [46] use an approach similar to Umbrella Sampling (section 2.3.1). They define a reaction coordinate $\xi(x)$ and use it to construct a biased potential like in equation 2.11. The generator architecture is then slightly modified to include the bias centre $\hat{\xi}_i$ around which configurations are to be sampled in the input vector c for each individual affine coupling layer, which can be seen on the right of figure 2.6. By training at different bias centres, the generator learns to produce samples under the additional condition, which can be used to guide shooting point generation towards the transition region of configuration space, where they are most likely to result in reactive paths.

Finding a good reaction coordinate can be an exceedingly complex task and oftentimes requires physical intuition about the system. Ideally, one would like to use the committor, since it perfectly defines the transition state ensemble, but it is mostly impossible to find analytically. Solving this problem is the aim of this thesis.

3. Results

The main result of this thesis is an algorithm called Generative AIMMD (GenAIMMD) that self-consistently learns the committor of a system and uses it as a condition during simultaneous training of a Boltzmann Generator. This allows for generating decorrelated shooting points in the high-efficiency region of configuration space without prior knowledge of a reaction coordinate.
In the first section of this chapter, the algorithm is introduced and explained. As a proof of concept, it is applied to two systems: a simple two-dimensional double-well-like system with two reaction channels referred to as *Wolfe-Quapp system* and a more complex model of a polymer in two dimensions. The results of GenAIMMD in these systems are presented and discussed in sections two and three of this chapter.

3.1. The GenAIMMD algorithm

GenAIMMD, an overview of which is depicted in figure 3.1, consists of multiple steps that can be grouped by their main area of focus into three categories

- **Dynamics**, which contains all steps where MD or MCMC simulations are performed,
- **AIMMD**, where the committor model described in section 2.4.2 is either trained or used to predict an estimate of the committor of configurations and
- **Generator**, which includes steps where the Boltzmann Generator training is performed or where the generator produces new configurations to be used during training.

These steps are repeated for a number of algorithm *cycles* until self-consistency is reached. The algorithm is started given a small number of configurations inside each stable state and in the transition region. These can be obtained using standard MD or MCMC simulation and TPS respectively. It should be noted that one does not need any reaction coordinate in order to produce these configurations, as the stable states can be sampled through equilibrium simulations and, as stated before, TPS operates without the need for a reaction coordinate. Once these samples are obtained and both the committor model as well as the Boltzmann Generator have been initialized, the algorithm loop begins, meaning that all of the following steps are repeated in each cycle. At the end of a cycle, the generator produces new configurations that, after some modification described below, join the training data set. This data set has a pre-defined maximum size and is structured as a queue, which means that the first set of training configurations that enters it will be the first to be replaced with new configurations.

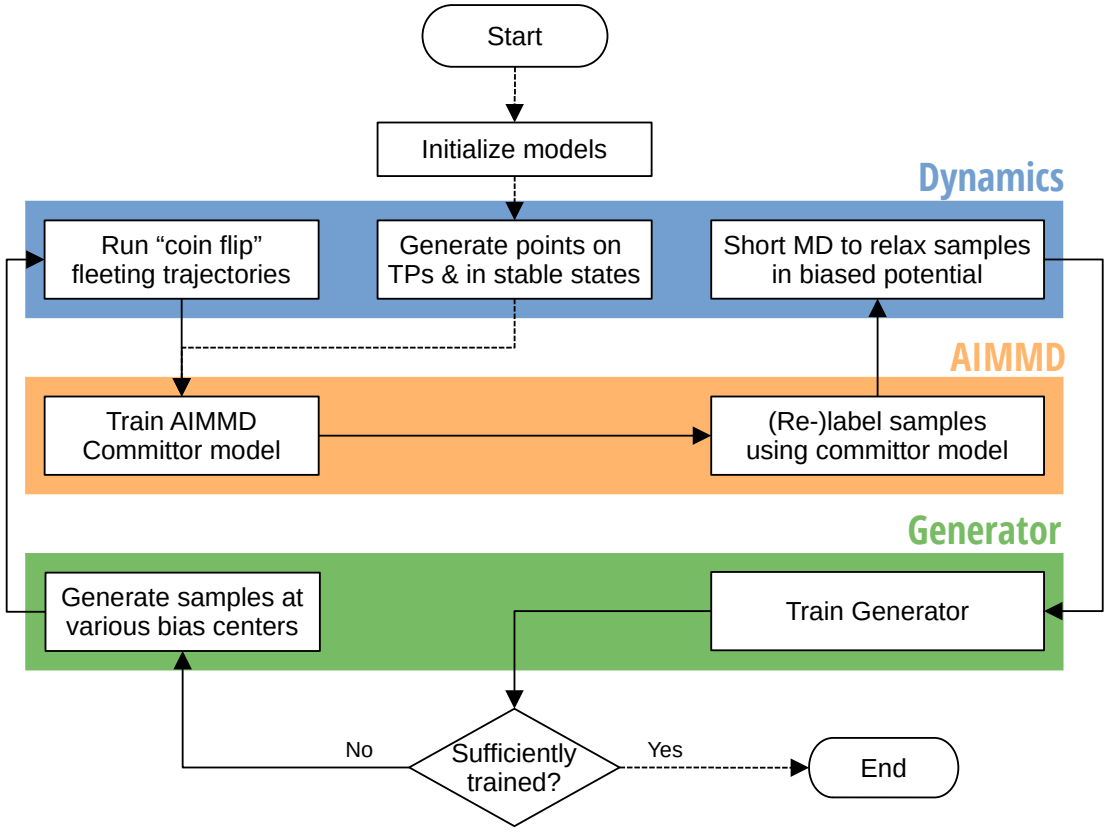


Figure 3.1.: Overview of the GenAIMMD algorithm. The three coloured boxed categorize each step, where *Dynamics* (blue) includes steps where MD or MCMC is performed, *AIMMD* (orange)-labelled steps use or train the committor model described by the AIMMD algorithm and *Generator* (green) refers to steps that use the Boltzmann Generator, either to train it or to produce new training samples. Solid lines connecting algorithm steps indicate the training loop, while steps connected by dashed lines are outside the loop.

Inside the training loop defined by solid lines in figure 3.1, the first step is to start two fleeting trajectories from each point in the current training set. The outcome of each trajectory, i.e., whether it first reached state A or state B, is recorded. Since this constitutes a Bernoulli experiment as discussed in section 2.4.2, which can be pictured as flipping an unfair coin where the respective probabilities of observing the outcomes "heads" or "tails" are not equal, these trajectories are referred to as *coin-flip* trajectories. This step is skipped for the set of initial samples, since there the outcomes are known, either because the sample is already inside one of the stable states or because it is a shooting point from a previous TPS simulation for which the outcomes were recorded. The configurations x in the training set along with shooting outcomes $s = \{s_i\}$ with $s_i \in \{-1, 1\}$ are then used to train the AIMMD committor model according to the procedure

described in section 2.4.2 and using the loss function $L_{\text{AIMMD}}(\theta|x,s)$ in equation 2.32. This is done for a pre-defined number of epochs that is large enough for the committor model's accuracy to benefit from the new data yet small enough to not spend too much compute time on this step.
For the Boltzmann Generator training in a later step (training by example to be precise), a set of samples from different bias windows with bias centres $\{\hat{\xi}_i\}$ according to the distribution associated with equation 2.11 is needed. As discussed before, the bias centres serve as conditions that are passed to each affine coupling layer of the generator and are further required when evaluating the potential energies of configurations. For this reason, the samples in the training set need to be labelled with bias window they are most likely a part of. To achieve this, they are fed into the current committor model that was updated in the previous step and are assigned a label based on the model's prediction for their committor, as will be discussed below.
There are now two approaches to assigning this label that are explored in more detail in the following sections. In the first one, a set of discrete bias windows with corresponding bias centres is chosen prior to starting the algorithm. Each sample is then labelled with the value of the closest bias centre according to the committor predicted for it by the committor model. In the second approach, the predicted committor is directly used as the value of the bias centre, placing the sample at the centre of the associated bias window. While each sample is now associated with some bias window, this does not mean that it can be regarded as having been independently drawn from the probability distribution induced by the corresponding biased potential (equation 2.11). Indeed, in the second approach, each sample is placed at the centre of the corresponding bias window, which would reproduce the correct distribution only in the limit of $k_{\text{bias}} \to \infty$. Also the first approach would not result in the correct distribution of the samples, since, by associating each sample with the closest bias window, one might end up with a disproportionately large number of high-energy configurations.
This necessitates performing the next step, which involves relaxing the samples in their respective potential, which can be achieved through a short MD or MCMC simulation. It should be noted that, like all the steps included in the *dynamics* category of the algorithm, this propagation can be completely parallelized in order to efficiently utilize all available computational resources.
With a set of correctly labelled training data at hand, one can now train the Boltzmann Generator, both via training by example and training by energy. Since this is generally computationally more expensive than training the committor model, one has to be especially careful in choosing the number of epochs to train for, where sufficient training is usually indicated by a plateau of the training losses.
At this point in the algorithm, one should evaluate whether self-consistency has been reached. One such exit condition can be a plateau of the Relative Effective Sample Size (RESS) as a function of algorithm cycles. Given a set of N samples $\{x_i\}_{i=1,\dots,N}$ with corresponding weights $\{w_i\}_{i=1,\dots,N}$, the effective sample size estimates the corresponding number of equally-weighted and independent samples that would have the same statistical accuracy as the $\{x_i\}$ [74]. If the effective sample size is then divided by the sample size N,

the relative effective sample size is obtained. An estimate for the RESS is given by [75]

$$\mathrm{RESS} = \frac{1}{N} \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2}. \tag{3.1}$$

This metric can be calculated for the samples generated by the Boltzmann Generator, whose weights are given by equation 2.55. Values of RESS $\simeq 1$ indicate that the generated distribution of the samples is close to the target distribution, while values close to zero mean poor sampling efficiency.
In case consistency is not achieved and the training continues, the last step in each cycle is to generate new training configurations that will replace part of the current training set. This is again either done by passing uniformly distributed bias centres (approach two from above) or pre-defined discrete bias centres (approach one from above) to the generator. Since these samples might have high energies, especially during the first few training cycles, it is sometimes necessary to relax them using MCMC. This step highlights the importance of providing initial samples outside of the stable states as an input to the algorithm. Without them, tests have shown that the generator performs linear interpolation in latent space to resolve the transition region, resulting in unphysical samples with extremely high energies. An example of this linear interpolation happening in the 2D system can be seen in appendix A.1.
Once the Boltzmann Generator is sufficiently trained, it can be used to generate shooting points for TPS simulations, ideally with a committor bias of 0.5. As discussed before, the shooting points coming out of the generator need to be re-sampled into the correct distribution given by

$$\rho'_{\mathrm{SP}}(x) = \frac{1}{Z_{\mathrm{SP}}} e^{-\beta U'_i(x)} \tag{3.2}$$

using the weights given in equation 2.55 before they can be used. Here, the $U'_i(x)$ is the biased potential given in equation 2.11 with committor bias centre $\hat{\xi}_i = 0.5$. Once re-sampled and propagated, the resulting reactive paths then also need to be re-weighted. Starting with the general form of the path weights given in equation 2.56, one can derive an expression for the special case where the shooting point distribution takes the above form by calculating

$$\begin{aligned}
\Omega'[X(\tau)] \propto \left[\sum_{i=1}^{N} \frac{\rho'_{\mathrm{SP}}(x_i)}{\rho(x_i)}\right]^{-1} &= \left[\sum_{i=1}^{N} \frac{\frac{1}{Z_{\mathrm{SP}}} e^{-\beta U'_j(x_i)}}{\frac{1}{Z_{NVT}} e^{-\beta U(x_i)}}\right]^{-1} \\
&= \frac{Z_{\mathrm{SP}}}{Z_{NVT}} \left[\sum_{i=1}^{N} \frac{e^{-\beta[U(x_i)+U_B^{\hat{\xi}_j}(x_i)]}}{e^{-\beta U(x_i)}}\right]^{-1} \propto \left[\sum_{i=1}^{N} \frac{e^{-\beta U(x_i)} e^{-\beta U_B^{\hat{\xi}_j}(x_i)]}}{e^{-\beta U(x_i)}}\right]^{-1} \\
&= \left[\sum_{i=1}^{N} e^{-\beta U_B^{\hat{\xi}_j}(x_i)]}\right]^{-1} = \frac{1}{\sum_{i=1}^{N} e^{-\beta \frac{k_{\mathrm{bias}}}{2}[\xi(x_i)-\hat{\xi}_j]^2}}.
\end{aligned} \tag{3.3}$$

These path weights depend solely on the harmonic bias potential, which is analytically known and cheap to evaluate. Since one is only interested in the relative path weights

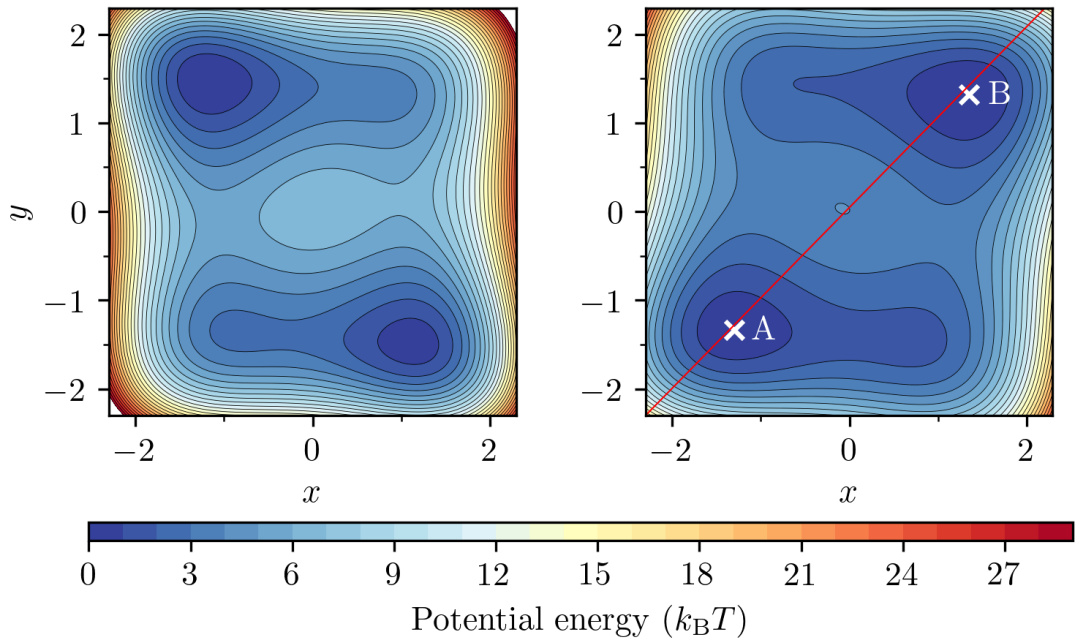


Figure 3.2.: The Wolfe-Quapp potential in its original (left) and rotated and scaled (right) form, shifted such that the global minimum is at zero in both cases. The modified version also marks the locations of the minima of the stable states A and B. It also includes the line $y(x) = x$ that is used to rotate the potential, aligning the stable states and the top of the barrier with it.

$\Omega'_{\mathrm{rel}}[X_k(\tau)]$ of the K paths $\{X_k(\tau)\}_{k=1,\dots,K}$, the proportionality constant drops through the normalization

$$\Omega'_{\mathrm{rel}}[X_k(\tau)] = \frac{\Omega'[X_k(\tau)]}{\sum_{i=1}^{K} \Omega'[X_i(\tau)]}. \tag{3.4}$$

3.2. The 2D model

The first tests are performed using a simple two-dimensional potential with two stable states A and B. There are many functional forms for such a potential that have been used in the past, but the one chosen for this work is the Wolfe-Quapp potential [76, 77]

$$U_{\mathrm{WQ}}(x, y) = x^4 + y^4 - 2x^2 - 4y^2 + xy + 0.3x + 0.1y. \tag{3.5}$$

This original form of the potential, which is depicted in figure 3.2 on the left-hand side, is rotated by approximately $\pi/25$ in order to align the two minima and the top of the barrier with the line defined by $y(x) = x$. This results in the potential energy function

$$\begin{aligned} U_{\mathrm{WQ,\,rot}}(x, y) =& c_1 \cdot x_1^4 + c_2 \cdot x_1^3 x_2 + c_3 \cdot x_1^2 + c_4 \cdot x_1^2 x_2^2 + c_5 \cdot x_1 \\ &+ c_6 \cdot x_1 x_2 + c_7 \cdot x_1 x_2^3 + c_8 \cdot x_2 + c_9 \cdot x_2^2 + c_{10} \cdot x_2^4 + c_{11} \end{aligned} \tag{3.6}$$

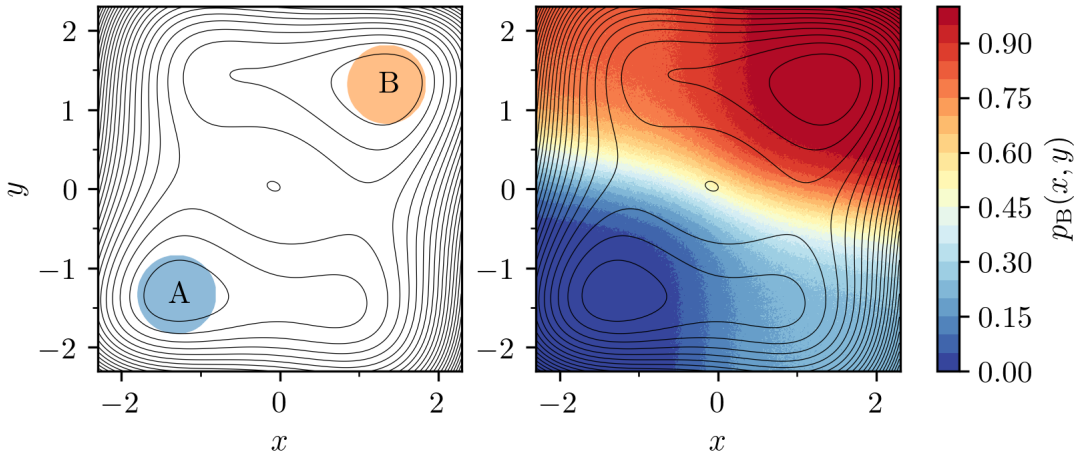


Figure 3.3.: Definition of the stable states A (blue) and B (orange) in the 2D model (left) and resulting committor $p_{\mathrm{B}}(x, y)$ (right). The committor estimates are obtained by starting multiple fleeting trajectories from each point in a fine grid over the depicted range of x and y and recording how many entered state B first.

with coefficients $c_1 = 0.969233$, $c_2 = -0.480614$, $c_3 = -2.155285$, $c_4 = 0.184602$, $c_5 = -0.285146$, $c_6 = -1.46487$, $c_7 = 0.480614$, $c_8 = 0.136719$, $c_9 = -3.84471$ and $c_{10} = 0.969233$, the constant $c_{11} = 6.76245$ shifting the global minimum to zero.
The potential is then multiplied with a scaling factor $s = h/c_{11}$ in order to make the height of the barrier variable using the parameter h in units of $k_{\mathrm{B}}T$. This results in the final form

$$U'_{\mathrm{WQ}}(x, y) = s \cdot U_{\mathrm{WQ,\,rot}}(x, y) \tag{3.7}$$

of the 2D potential, which is also depicted in figure 3.2 on the right-hand side with a barrier height of $h = 4\ k_{\mathrm{B}}T$.

3.2.1. State definition and committor estimation

In order to define the stable states A and B, an order parameter in the form of a bimodal Gaussian function centred at either minimum of the potential energy

$$\chi_{\mathrm{WQ}}(\vec{r}) = a \cdot \left[e^{-b_1 ||\vec{r} - \vec{r}_{\mathrm{B}}||^2} - e^{-b_2 ||\vec{r} - \vec{r}_{\mathrm{A}}||^2} \right] \tag{3.8}$$

with empirically chosen parameters $a = 100$ and $b_1, b_2 = 10$ and $\vec{r} = (x, y)^{\mathrm{T}}$ is defined, where $\vec{r}_{\mathrm{A}}, \vec{r}_{\mathrm{B}}$ are the positions of the minima. Configurations with $\chi_{\mathrm{WQ}}(\vec{r}) < q_A = -8.2085$ or $\chi_{\mathrm{WQ}}(\vec{r}) > q_B = 8.2085$ are then considered to be inside the stable state A or B respectively. This construction effectively results in state definitions that are discs with radius 0.5 centred at the respective minima, which can be seen on the left-hand side in figure 3.3.

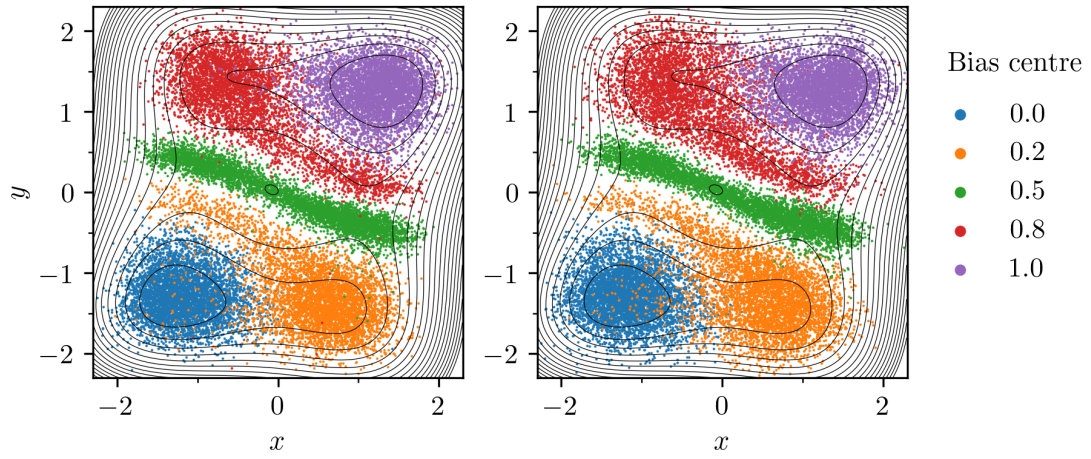


Figure 3.4.: Samples used for training a Boltzmann Generator in the 2D model (left) and configurations produced by the trained generator (right). The samples belong to different committor bias centres, which are listed on the far right. The training samples were obtained via Umbrella Sampling and the generated samples have not yet been re-sampled into the correct distribution, yet they already agree well with the US data.

Using these state definitions, the committor $p_{\mathrm{B}}(x, y)$ can be estimated by starting a large number of fleeting trajectories from each point of a fine grid and recording the relative number of trajectories that entered state B before state A. In this case, 1000 trajectories are used per point and the points are equally spaced on a quadratic grid with lattice constant 0.01 over the range $-2.3 \leq x, y \leq 2.3$. The results of this committor estimation are depicted on the right-hand side of figure 3.3.

3.2.2. Conditioning a Boltzmann Generator

Before running GenAIMMD, the possibility of conditioning a Boltzmann Generator on the committor of the 2D system as well as the correctness of the point- (equation 2.55) and path-weights (equation 2.56) needs to be verified. To this end, the committor estimates obtained as outlined in the previous section by starting fleeting trajectories on a square lattice are interpolated linearly between lattice sites and this interpolation is used as the reaction coordinate for conditioning the generator. This serves as a test of what to expect from a fully trained committor model.
In order to achieve this, a set of training configurations at different committor bias centres is needed, which is produced using Umbrella Sampling in five bias centres. An excerpt from the training set is plotted on the left-hand side of figure 3.4. After the training is done, the generator is used to produce a set of configurations that can be used for testing, which can be seen in the right of the same figure.
As discussed before, these samples are not distributed according to the distribution

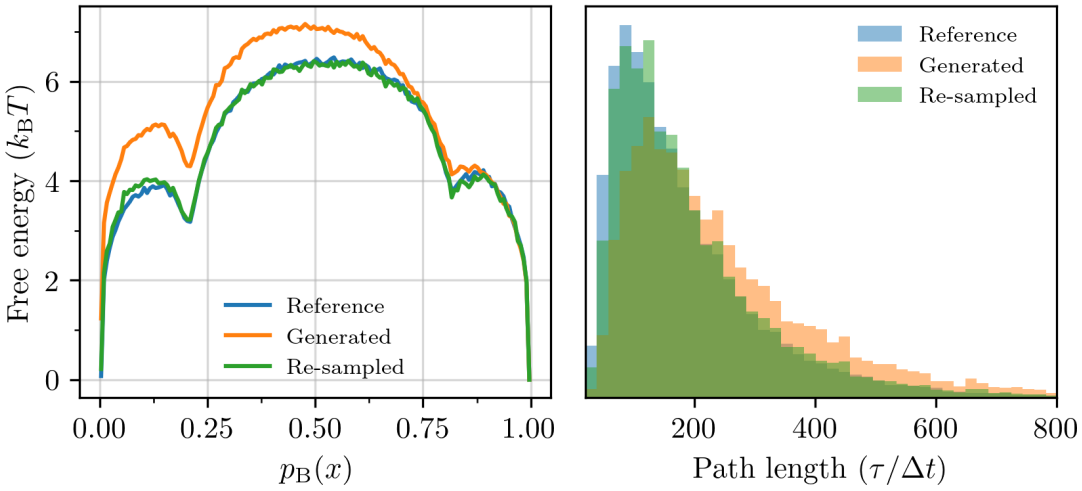


Figure 3.5.: Left: Free energy as a function of the committor estimated using WHAM on different committor-biased configurations from 11 bias windows of the 2D system. Right: Distribution of path lengths $\tau/\Delta t$ in units of MD steps of different sets of transition paths in the 2D system. In both plots, the different colours refer to the means by which the underlying samples were obtained. *Reference* (blue) is a result from reference data, MD samples obtained via Umbrella Sampling (US) in case of the free energy estimates on the left and TPS samples in case of the path length distributions on the right. *Generated* (orange) signifies that the samples involved were not re-sampled into the correct distribution. For the free energies, this means that the configurations produced by the Boltzmann Generator were used without modification. In case of the path length distribution, the shooting points were re-sampled but the resulting reactive paths were not. *Re-sampled* (green) finally indicates that the samples (configurations for the free energies/transition paths for the path length distribution) were re-sampled into their respective target distribution. Therefore, the results should match those produced by the reference data.

associated with the biased potential, which is why they need to be re-weighted according to equation 2.55. To verify that this procedure works, the free energy as a function of the committor is calculated using the method described in section 2.3.1 for both the re-weighted samples produced by the generator and the configurations in the training set obtained via Umbrella Sampling, the latter serving as a reference. To illustrate the necessity of the re-weighting, the free energy resulting from the generated configurations prior to re-weighting is also computed. These three free energies are plotted on the left-hand side of figure 3.5.
Having confirmed that the generated samples can be re-weighted to follow the appropriate distribution, the next step is to do the same for the transition paths propagated from

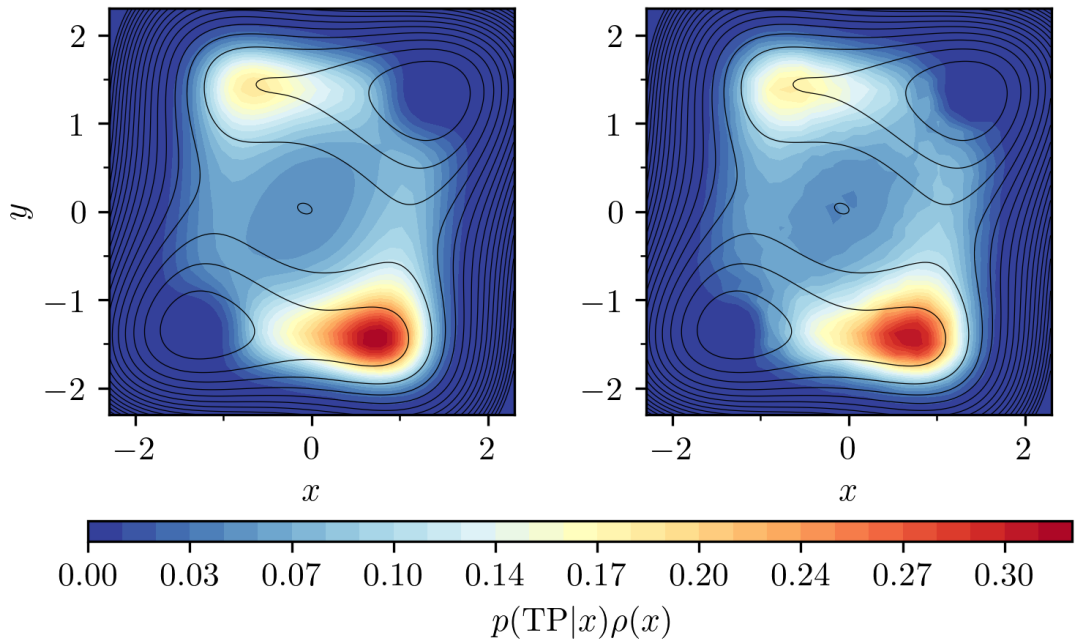


Figure 3.6.: The distribution $p(\mathrm{TP}|x)\rho(x)$ obtained by inserting the interpolated numerical committor estimates into equation 2.21 and multiplying with $\rho(x)$ (left) which serves as a reference to the 2D histogram of points on reactive paths, which were propagated from generated and re-sampled shooting points and then themselves re-sampled into the correct path distribution (right).

the configurations generated at committor bias 0.5. These configurations, which serve as shooting points for the paths, need to be re-sampled into the $\rho_{\mathrm{SP}}(x)$ distribution from equation 2.56. Otherwise, the resulting path weights would be incorrect. This is done by randomly picking configurations with replacement according to their weight from equation 2.55.
After the paths are generated, the same re-sampling procedure is applied using the path weights, ideally resulting in correctly distributed transition paths. In order to verify this, two checks are performed. First, a 2D histogram of all points along transition paths is created, which are distributed according to $p(\mathrm{TP}|x)\rho(x)$. Since the committor is known from simulation on the quadratic grid in the region of interest, a reference for this distribution can by obtained by inserting it into equation 2.21 and multiplying with the equilibrium Boltzmann distribution $\rho(x)$ evaluated at the grid points. The two plots are depicted in figure 3.6 and show good agreement. In order to perform the second check, reference TPS data is acquired. The distribution of the path lengths $\tau/\Delta t$ of this reference data is then compared to those of the generated paths before and after re-sampling. This comparison can be seen to be successful on the right-hand side of figure 3.5.

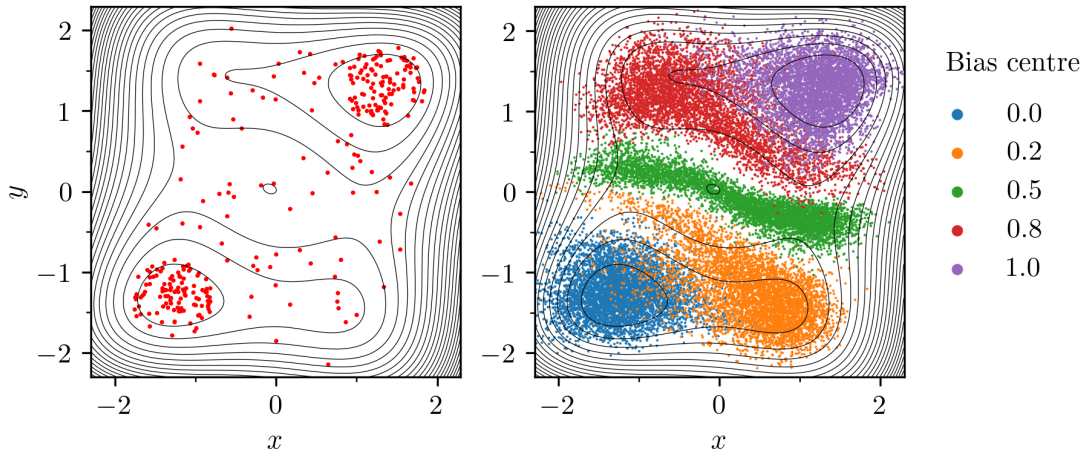


Figure 3.7.: Configurations in stable states and on transition paths that are used to initiate GenAIMMD in the 2D system (left) and configurations generated at different committor bias centres using the Boltzmann Generator that was trained through GenAIMMD (right).

3.2.3. Testing GenAIMMD

The results presented so far show that both the configuration as well as the path reweighting is successful and the correct distributions can be recovered, provided that the high-probability regions are sufficiently sampled. The next step is to test GenAIMMD on the 2D model. The hyperparameters of the Boltzmann Generator and the committor model as well as the training parameters for GenAIMMD are listed in appendix A.2.
From the checks performed in the previous section, the initial data in the stable states and along transition paths required to start GenAIMMD is already at hand and depicted on the left-hand side of figure 3.7. The total number of initial samples used is 300, 200 of which are in the stable states and the remaining 100 are along transition paths. Using this, GenAIMMD manages to successfully train both the Boltzmann Generator and the committor model within just 20 cycles, at which point the relative effective sample size averaged over all bias windows plateaus, which can be seen on the left-hand side of figure 3.8, and further training results in no increase in sampling accuracy. It should also be noted that the RESS already starts relatively high in the first cycle, indicating good training performance in just one cycle for the simple 2D system. Samples that were generated at various bias centres using the trained Boltzmann Generator can be seen on the right-hand side of figure 3.7 and should be compared to the reference data obtained via Umbrella Sampling on the left-hand side of figure 3.4.
The training results for the committor model are depicted on the right-hand side of figure 3.8 and match the reference data on the left-hand side of figure 3.3.

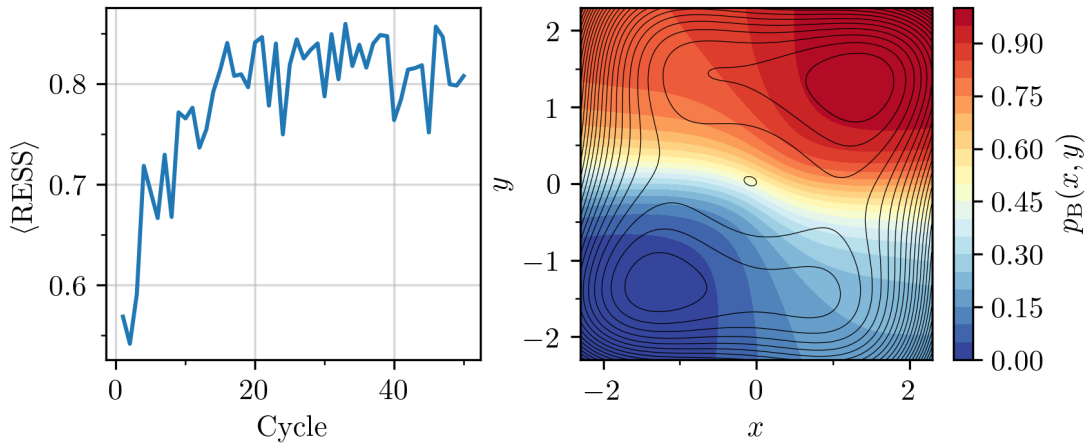


Figure 3.8.: Relative Effective Sample Size (RESS) averaged over different bias windows as a function of algorithm training cycles (left) and committor $p_B(x, y)$ of the 2D system learned in parallel to the Boltzmann Generator training using GenAIMMD (right).

3.3. The polymer model

Having shown excellent performance in the two-dimensional system, GenAIMMD is now applied to a more sophisticated model with higher dimensions. To this end, the same model of a polymer consisting of $N = 7$ monomers in two dimensions that was also studied by Falkner et al. [46] was chosen. After applying an internal coordinate transformation described below, the total number of degrees of freedom that the generator as well as the committor model have to work with is 11, which constitutes a significant increase in complexity compared to the Wolfe-Quapp model.
In this model, the total potential energy is given by a sum of three potentials: a Lennard-Jones (LJ) potential

$$U_{\mathrm{LJ}} = 4\varepsilon \sum_{i=1}^{N} \sum_{j>i}^{N} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \tag{3.9}$$

with $\sigma = \varepsilon = 1$ which models the pairwise interactions of the monomers and prevents them from overlapping, a harmonic bond stretching potential

$$U_{\mathrm{bond}} = \frac{k_{\mathrm{bond}}}{2} \sum_{i=1}^{N-1} (r_{i,i+1} - r_{\mathrm{ref}})^2 \tag{3.10}$$

that defines a preferred bond length $r_{\mathrm{ref}} = \sqrt[6]{2}\sigma$ and an angular potential

$$U_{\mathrm{angle}} = \frac{k_{\mathrm{angle}}}{2} \sum_{i=1}^{N-2} [1 - \cos(\varphi_{i,i+1,i+2} - \varphi_{\mathrm{ref}})] \tag{3.11}$$

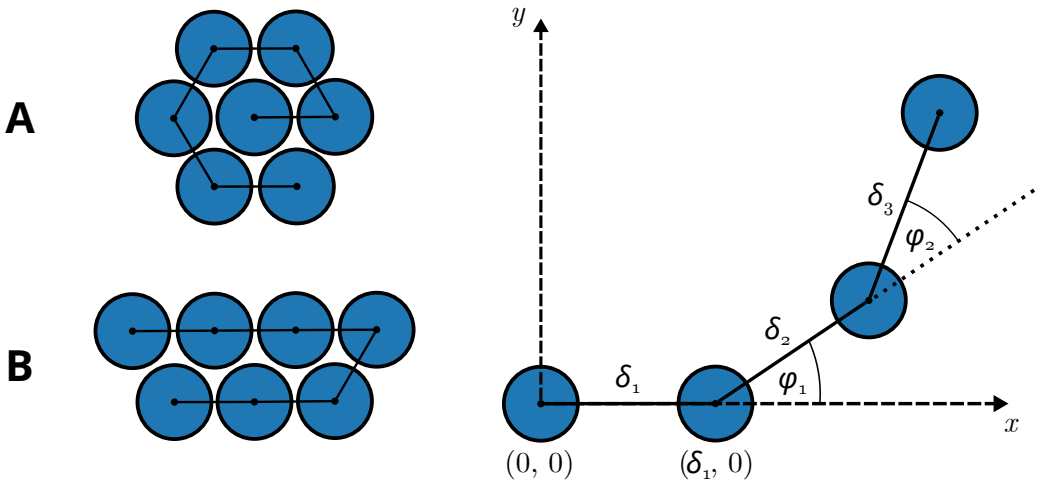


Figure 3.9.: Compact (A) and extended (B) stable state conformations of the polymer model (left) and definition of the internal coordinate system for a symbolic polymer with only 4 monomers (right).

that has the same effect as U_{bond} but for the angle $\varphi_{i,i+1,i+2}$ between three consecutive monomers with a preferred angle $\varphi_{\text{ref}} = \pi$ rad. Here, $r_{i,j}$ denotes the distance between two monomers i and j and $k_{\text{bond}} = 5\,\varepsilon\sigma^{-1}$ and $k_{\text{angle}} = 1.4\,\varepsilon$ rad^{-1} define the strengths of the bond and angle potentials respectively. Therefore, the total energy of the polymer model is given by

$$U_{\text{polymer}} = U_{\text{LJ}} + U_{\text{bond}} + U_{\text{angle}}. \tag{3.12}$$

3.3.1. State definition and internal coordinates

The polymer system has two stable states A and B, which are depicted in figure 3.9. They are defined using the radius of gyration

$$R_{\text{G}}(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\vec{x}_i - \langle x\rangle\,||^2} \tag{3.13}$$

as an order parameter, where $x = (x_1, y_1, \ldots, x_N, y_N)^{\text{T}}$ is a conformation of the system in Cartesian coordinates, $\vec{x}_i = (x_i, y_i)^{\text{T}}$ is the position of monomer i and $\langle x\rangle$ is the centre of mass of that conformation. Therefore, $R_{\text{G}}(x)$ is a measure for the spatial extent of the polymer. Since state A consists of curled up conformations (low $R_{\text{G}}(x)$) while the conformations in state B are extended (large $R_{\text{G}}(x)$, see figure 3.9) and the transition is characterized by the polymer either extending or contracting, Falkner et al. [46] used the radius of gyration combined with the value of the Lennard-Jones contribution U_{LJ} to the total potential as a reaction coordinate. In this work, only the radius of gyration is used to

define the two stable states as $R_G(x) < q_A = 1.05\,\sigma$ for state A and $R_G(x) > q_B = 1.2\,\sigma$ for state B.
In order to reduce the problem to the relevant degrees of freedom and account for rotational and translational invariances, a coordinate transformation between Cartesian and internal coordinates is performed, such that the Boltzmann Generator and the committor model are trained on internal coordinates. These internal coordinates consist of the $N-1$ bond lengths and the $N-2$ angles between three consecutive monomers, resulting in a total of $2N-3 = 11$ degrees of freedom compared to the $2N = 14$ in Cartesian coordinates. A conformation in internal coordinates is then given by $a = (\delta_1, \delta_2, \ldots, \delta_{N-1}, \varphi_1, \varphi_2, \ldots, \varphi_{N-2})^T$ with $\delta_i := r_{i,i+1}$ and $\varphi_i := \varphi_{i,i+1,i+2}$. When performing the transformation from internal to Cartesian coordinates, the three missing degrees of freedom are accounted for by choosing the frame of reference such that the first monomer is placed at the origin, i.e., $\vec{x}_1 = (0,0)^T$, and the second monomer is placed on the x-axis, such that $\vec{x}_2 = (\delta_1, 0)^T$. A visualization of the definition of the internal coordinate system can be seen in figure 3.9.
The Jacobian determinant of the transformation from internal to Cartesian coordinates is given by [46]

$$\det J = \prod_{i=2}^{N-1} \delta_i. \tag{3.14}$$

This needs to be included when calculating the Jacobian determinant of the transformation performed by the Boltzmann Generator, both in the forward and in the backward direction.

3.3.2. Modifications to the GenAIMMD implementation

The endeavour of training the generator on higher dimensional data necessitates some additions to the naive implementation used for the 2D model. As discussed in the previous section, one of these is the inclusion of a transform layer mapping between Cartesian and internal coordinates, which already accounts for translational and rotational invariances. To further reduce complexity, each configuration in the training set is constrained to have a positive first angle φ_1, meaning that, should the third monomer have a negative y-coordinate, the entire polymer is mirrored along the x-axis, which has no effect on the potential energy of the configuration. The internal coordinates are also passed through a normalization layer before they are shown to the generator, which subtracts the mean and divides by the standard deviation of each coordinate in the training set.
During training, especially for the first cycles, the generator produces configurations with extremely high energy, which causes the loss function to become numerically unstable. To mitigate this, the energy is clamped by regularizing it using a function [45]

$$U'_{\text{polymer}} = \begin{cases} U_{\text{polymer}} & \text{if } U_{\text{polymer}} < U_{\text{clamp}} \\ U_{\text{clamp}} + \log(U_{\text{polymer}} - U_{\text{clamp}} + 1) & \text{if } U_{\text{clamp}} \leq U_{\text{polymer}} < U_{\max} \\ U_{\text{clamp}} + \log(U_{\max} - U_{\text{clamp}} + 1) & \text{if } U_{\text{polymer}} \geq U_{\max}, \end{cases} \tag{3.15}$$

where U_{clamp} is the threshold for energy clamping and $U_{\max}$ is the energy that defines the location of the function's plateau.

Although their impact on the loss function can be softened, these high energy samples still pose a challenge when it comes to obtaining committor estimates through fleeting trajectories due to the large forces acting on them, which make it impossible to propagate such configurations using MD. For this reason, the samples produced by the generator in the last step of GenAIMMD are relaxed using a short MCMC simulation before they are added to the training set and used to estimate committors, since MCMC is mostly unaffected by initially high energies.
In the 2D model, the Boltzmann Generator was trained using constant training parameters (e.g. the learning rate η, the training by energy/example weights λ_{KL}, λ_{ML} and the clamp/maximum energies U_{clamp}, U_{max}). In the polymer model, this approach is not applicable, since the first epochs of training require more carefully chosen parameters to ensure stability while later epochs benefit from a more aggressive training scheme. Therefore, the introduction of a training scheduler that gradually changes the training parameters greatly increases training efficiency.
Finally, recall that AIMMD models the committor $p_{\mathrm{B}}(x)$ using equation 2.29, i.e., as the standard logistic function applied to the output $q(x|\theta) := q(x)$ of the neural network. For the polymer model, this transformation $q(x) \to p_{\mathrm{B}}(x)$ is dropped, and $q(x)$ is instead used as a reaction coordinate and a bias for the Boltzmann Generator, which is mathematically equivalent due to the bijective nature of the standard logistic function, for which the inverse, which is called the *logit* function, is given by

$$q(x) = -\log\left[\frac{1}{p_{\mathrm{B}}(x)} - 1\right]. \tag{3.16}$$

Obviously, the relations

$$\lim_{p_{\mathrm{B}}\to 0} q[p_{\mathrm{B}}] = -\infty, \quad \lim_{p_{\mathrm{B}}\to 1} q[p_{\mathrm{B}}] = \infty \quad \text{and} \quad p_{\mathrm{B}}[q=0] = 0.5 \tag{3.17}$$

hold. The reason for this change is that, in equilibrium, the high probability regions of configuration space correspond to values of p_{B} close to 0 or 1, meaning that the majority of configurations are squeezed into the small regions $p_{\mathrm{B}}(x) \simeq 0$ and $p_{\mathrm{B}}(x) \simeq 1$. The effects of this can also be seen in the free energy plot on the left in figure 3.5. Using $q(x)$ instead of $p_{\mathrm{B}}(x)$ expands these regions, increasing the resolution of statistics performed there and helping with numerical stability.

3.3.3. Conditioning on the radius of gyration

Like in the 2D model, before using GenAIMMD to learn the committor while simultaneously training the Boltzmann Generator, a generator is trained in an initial test using the radius of gyration as the condition. The training sample set is again obtained via Umbrella Sampling in 10 equally spaced bias windows between values of 1.013 and 1.25 for the radius of gyration with $20,000$ samples per window and a temperature of $k_{\mathrm{B}}T = 0.1$. Figure 3.10 shows two histograms, one of the radius of gyration of the training set and one of the radius of gyration of the samples generated using the trained Boltzmann Generator.

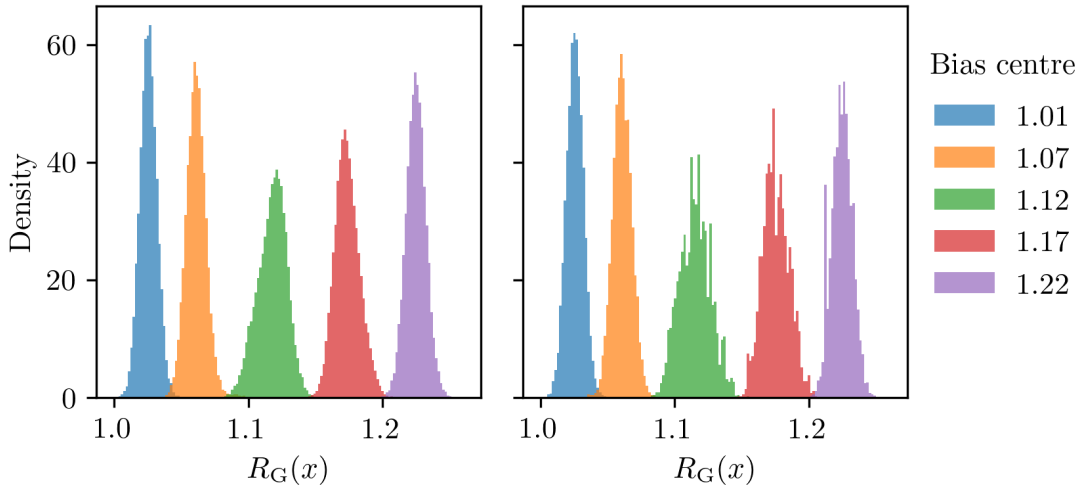


Figure 3.10.: Left: Distribution of the radius of gyration $R_G(x)$ of the samples used for training a Boltzmann Generator on the polymer model with $R_G(x)$ as the condition. The samples are obtained via Umbrella Sampling in the bias centres listed on the right of the two plots. Right: The distribution of $R_G(x)$ of the re-weighted samples produced by the trained generator using the same bias centres.

To complete the analysis, the free energy as a function of the radius of gyration is calculated. To this end, reference data is produced via US using the same parameters as before but for 22 bias windows, which is used to compute the reference free energy. This is then compared to the free energy obtained from samples produced by the generator in the same bias windows. Figure 3.11 shows this comparison and the results match those obtained by Falkner et al. [46] for the same system.

3.3.4. Testing GenAIMMD

Knowing that the conditioning works, GenAIMMD can now be tested for the polymer model. As before, the parameters used for training are listed in appendix A.2. To verify that the Boltzmann Generator was trained successfully, samples are generated at different bias centres and the re-weighted distribution of the corresponding committor values is compared to that of the reference data. The latter is again obtained using Umbrella Sampling biased by the learned committor in the same bias windows. The results of this analysis can be seen in figure 3.12 and show that the Boltzmann Generator training was successful.
Testing the performance of the committor model is not as straightforward as in the 2D model, where this was done by visually inspecting the committor predictions at each site of a square lattice and comparing them to reference data obtained by starting multiple trajectories from each lattice site. This is obviously not possible in the polymer model

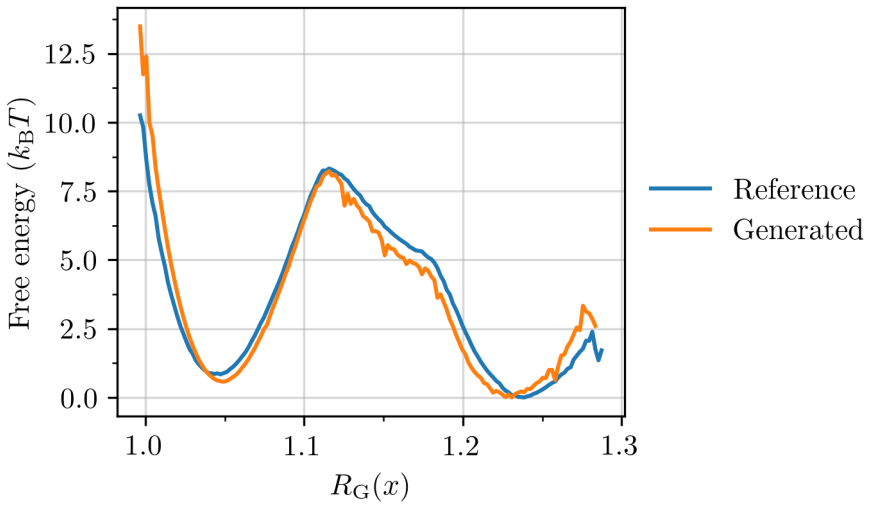


Figure 3.11.: Free energy as a function of the radius of gyration $R_G(x)$ for the polymer model, calculated from Umbrella Sampling samples (Reference) and from samples generated by the trained Boltzmann Generator (Generated). The same 22 bias windows are used in both cases.

because of its larger number of degrees of freedom. Therefore, a different approach is chosen, where the testing configurations are not obtained from a grid but from Umbrella Sampling simulations which use the radius of gyration as a bias instead of the committor, which would theoretically constitute a better reaction coordinate as discussed before. The reason for this choice is that involving the learned committor in producing data that is then used to check the correctness of the learned committor would introduce a bias, making the results less meaningful. The thus obtained configurations are then shown to the committor model, whose predictions are subsequently plotted against the reference committor estimates obtained analogously to those in the 2D model, i.e., by propagating multiple trajectories from each configuration and observing their outcomes. This plot can be seen on the left-hand side of figure 3.13 and it shows the committor predictions to be rather accurate.
Lastly, the relative effective sample size of samples produced by the Boltzmann Generator after each training cycle of GenAIMMD can be investigated. As for the 2D model, samples are produced in different bias windows and the RESS averaged over all windows is computed. Then, the averaged RESS is plotted against the algorithm cycle, which can be seen on the right-hand side of figure 3.13. Initially, the RESS grows, indicating that the quality of the samples produced by the generator increases as the training progresses. Ultimately, it plateaus at a value rather smaller compared to the 2D model, which is due to the larger number of degrees of freedom of the polymer model and therefore the increase in complexity of the sampling. As a result, more sample need to be generated in order to achieve the same level of accuracy as in the 2D model. However, due to the fact

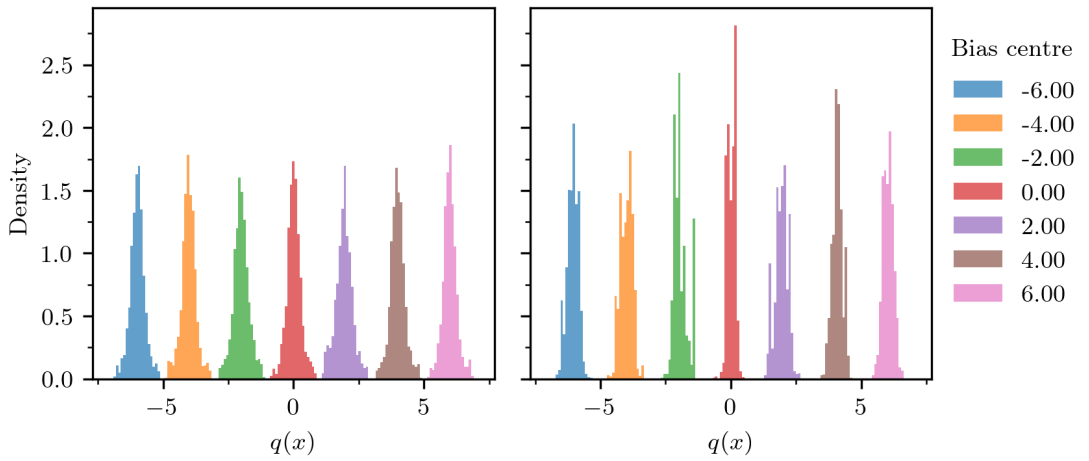


Figure 3.12.: Distribution of the committor values of Polymer configurations produced at different bias centres listed on the right via Umbrella Sampling (left) and generated and re-weighted using a Boltzmann Generator trained using GenAIMMD (right). The function $q(x)$ is related to the committor $p_{\mathrm{B}}(x)$ through equation 3.16.

that the sampling and re-weighting is computationally inexpensive, this poses no great challenge. This concludes the analysis of the performance of GenAIMMD.

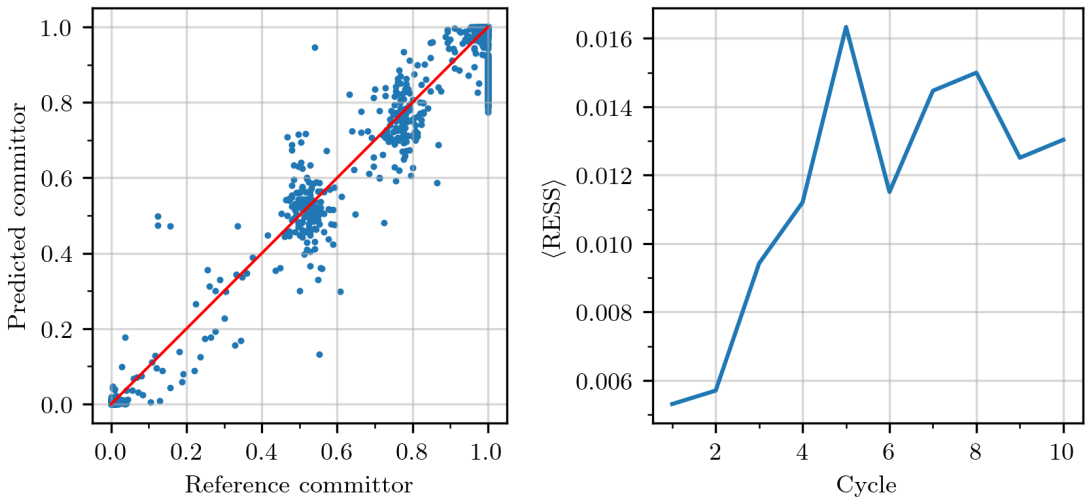


Figure 3.13.: Left: Predicted and reference committor for polymer configurations obtained via Umbrella Sampling using the radius of gyration as a bias coordinate. The red line indicates perfect agreement between the predictions made by the committor model and the committor estimated by starting multiple trajectories at each testing configuration and recording their outcomes. Right: Relative effective sample size of polymer samples produced by the Boltzmann Generator averaged over multiple bias centres as a function of algorithm training cycles.

4. Discussion and conclusion

Boltzmann Generators [45] enable one to draw uncorrelated samples from a complex Boltzmann distribution by learning a transformation between an easy-to-sample-from prior distribution and the target Boltzmann distribution. It was shown by Falkner et al. [46] that it is possible to use the samples generated by a Boltzmann Generator as shooting points for Transition Path Sampling simulations. The latter can thus be parallelized and produce uncorrelated transition paths. In this approach, the efficiency of the simulation can be maximized by steering the generator to produce samples preferentially in the high-efficiency region defined by some reaction coordinate $\xi(x)$. However, it was discussed that finding a good reaction coordinate is highly non-trivial or even impossible in complex systems, often requiring physical, chemical or biological insight into the mechanisms governing the reaction.

It was noted that the committor $p_{\mathrm{B}}(x)$ of a system constitutes the ideal reaction coordinate, providing a purely statistical description of the reaction. However, it is an intractable function that quickly becomes computationally expensive to estimate, especially in higher dimensional systems. This can be addressed by harnessing the predictive capabilities of artificial neural networks, as was done by Jung et al. [47] who introduced an algorithm called AIMMD that learns the committor of a system in parallel to a TPS simulation, biasing shooting point selection towards the $p_{\mathrm{B}}(x) \simeq 0.5$ region and utilizing readily available shooting outcomes for training the model. Since this algorithm relies on TPS, however, is is inherently sequential and the training data suffers from correlations.

Ideally, one would like to combine the two methods, generating uncorrelated samples using a Boltzmann Generator and using them to train the committor model. This, however, results in a circular dependence, since the Boltzmann Generator in turn requires the committor in order to sample the high-efficiency region.

Resolving this was the main goal of this thesis, the result being an algorithm called Generative AIMMD (GenAIMMD) that simultaneously and self-consistently trains a committor model of a system and uses it to condition a Boltzmann Generator. This algorithm was tested using two systems: a simple two-dimensional bi-stable potential and a more complex model of a polymer consisting of seven monomers in two dimensions. The tests have shown that, in both systems, GenAIMMD was successful in training both the committor model and the Boltzmann Generator. However, it should be mentioned that no tests were performed that compare the efficiency of TPS enhanced using the proposed algorithm to that of the standard shooting algorithm.

However, going from the two dimensions of the simple system to the eleven dimensions of the polymer system proved to be no trivial matter and required careful choices of the coordinate system and the parameters of the training, largely aided by the previous work of Falkner et al. [46]. The number of initial samples also had to be increased, although it

was still small enough to not cause any significant increase in computational overhead. Even with these measures in place, the quality of the samples produced in the complex system is vastly inferior to those in the 2D system.
This is likely not a result of GenAIMMD performing badly, but rather a more fundamental issue with Boltzmann Generators struggling to learn higher-dimensional target distributions [24]. To confirm this, a Boltzmann Generator was conditioned independently using the fully trained committor model, the resulting sampling efficiency being equivalent to that of a generator trained via GenAIMMD. In the past, efforts have been made to improve the architecture and the training procedure of Boltzmann Generators, showing promising results when applied to medium-sized proteins [78]. Ultimately, a central limiting factor is the extensivity of the system energy leading to a divergence of two systems with different energies that grows exponentially with their size. This is problematic in the re-weighting procedure, where the distribution produced by the Boltzmann Generator needs to be re-weighted into the target Boltzmann distribution. Here, even slight differences between the distributions may lead to the re-weighting becoming unstable for large systems. Another problem specifically related to Boltzmann Generators that hinders their efficiency is the large number of model parameters required to provide the necessary expressivity, which is a consequence of the invertibility requirement of the transformation. However, simply increasing the number of parameters in the S and T networks of the affine coupling layers may still not result in the desired expressivity, which is due to the simple nature of the affine transformations in the Real NVP architecture. To resolve this, other coupling functions like splines [79, 80] may be considered.
In recent years, the sampling capabilities of a different class of generative models called *diffusion models* [81, 82] have been explored and even applied to Transition Path Sampling directly [83]. These models have shown to be more robust and expressive than normalizing flows and can successfully be applied to learning tasks involving highly complex target distributions [24]. This makes them a promising alternative to flow-based models.
The algorithm developed in this thesis is completely modular. This means that the generative part currently implemented using a Boltzmann Generator can be easily exchanged, should a different approach that still offers a numerically tractable re-weighting factor prove to be better suited. Currently, diffusion models seem to be most likely to fulfil that role, showing better performance in higher-dimensional systems compared to Boltzmann Generators.
It should further be mentioned that the success of GenAIMMD in training the Boltzmann Generator strongly depends on the quality of the initial samples. During tests, the presence of unphysical configurations in the sample set revealed that such configurations are then reproduced by the Boltzmann Generator, which continuously feeds them back into the training loop in case they are not sufficiently relaxed in the MD propagation. This can be detrimental to the sampling performance of the generator, which is why it is essential to ensure that the initial samples are correctly distributed.

Bibliography

[1] Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids*. Oxford University PressOxford, 2 edition, June 2017. ISBN 9780198803195 9780191841439. doi: 10.1093/oso/9780198803195.001.0001. URL https://academic.oup.com/book/27866.

[2] M. Volmer and A. Weber. Keimbildung in übersättigten Gebilden. *Zeitschrift für Physikalische Chemie*, 119U(1):277–301, January 1926. ISSN 2196-7156, 0942-9352. doi: 10.1515/zpch-1926-11927. URL https://www.degruyter.com/document/doi/10.1515/zpch-1926-11927/html.

[3] R. Becker and W. Döring. Kinetische Behandlung der Keimbildung in übersättigten Dämpfen. *Annalen der Physik*, 416(8):719–752, January 1935. ISSN 0003-3804, 1521-3889. doi: 10.1002/andp.19354160806. URL https://onlinelibrary.wiley.com/doi/10.1002/andp.19354160806.

[4] S. Jungblut and C. Dellago. Pathways to self-organization: Crystallization via nucleation and growth. *The European Physical Journal E*, 39(8):77, August 2016. ISSN 1292-8941, 1292-895X. doi: 10.1140/epje/i2016-16077-6. URL http://link.springer.com/10.1140/epje/i2016-16077-6.

[5] G.M. Torrie and J.P. Valleau. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23 (2):187–199, February 1977. ISSN 00219991. doi: 10.1016/0021-9991(77)90121-8. URL https://linkinghub.elsevier.com/retrieve/pii/0021999177901218.

[6] Robert H. Swendsen and Jian-Sheng Wang. Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters*, 57(21):2607–2609, November 1986. ISSN 0031-9007. doi: 10.1103/PhysRevLett.57.2607. URL https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

[7] Charles J. Geyer. Markov Chain Monte Carlo Maximum Likelihood. 1991. URL https://hdl.handle.net/11299/58440.

[8] Koji Hukushima and Koji Nemoto. Exchange Monte Carlo Method and Application to Spin Glass Simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, June 1996. ISSN 0031-9015, 1347-4073. doi: 10.1143/JPSJ.65.1604. URL http://journals.jps.jp/doi/10.1143/JPSJ.65.1604.

[9] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314(1-2):141–151, November 1999. ISSN

00092614. doi: 10.1016/S0009-2614(99)01123-9. URL https://linkinghub.elsevier.com/retrieve/pii/S0009261499011239.

[10] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences*, 99(20):12562–12566, October 2002. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.202427399. URL https://pnas.org/doi/full/10.1073/pnas.202427399.

[11] Christoph Dellago, Peter G. Bolhuis, and David Chandler. Efficient transition path sampling: Application to Lennard-Jones cluster rearrangements. *The Journal of Chemical Physics*, 108(22):9236–9245, June 1998. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.476378. URL https://pubs.aip.org/jcp/article/108/22/9236/477081/Efficient-transition-path-sampling-Application-to.

[12] OpenAI. ChatGPT, 2024. URL https://chat.openai.com.

[13] Anthropic. Claude, 2025. URL claude.ai.

[14] OpenAI. Sora, 2025. URL https://openai.com/sora/.

[15] Leonardo Interactive Pty Ltd. Leonardo, 2025. URL https://leonardo.ai/.

[16] Adam C. Mater and Michelle L. Coote. Deep Learning in Chemistry. *Journal of Chemical Information and Modeling*, 59(6):2545–2559, June 2019. ISSN 1549-9596, 1549-960X. doi: 10.1021/acs.jcim.9b00266. URL https://pubs.acs.org/doi/10.1021/acs.jcim.9b00266.

[17] O. Anatole Von Lilienfeld and Kieron Burke. Retrospective on a decade of machine learning for chemical discovery. *Nature Communications*, 11(1):4895, September 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-18556-9. URL https://www.nature.com/articles/s41467-020-18556-9.

[18] Balaranjan Selvaratnam and Ranjit T. Koodali. Machine learning in experimental materials chemistry. *Catalysis Today*, 371:77–84, July 2021. ISSN 09205861. doi: 10.1016/j.cattod.2020.07.074. URL https://linkinghub.elsevier.com/retrieve/pii/S0920586120305502.

[19] Nicolae Sapoval, Amirali Aghazadeh, Michael G. Nute, Dinler A. Antunes, Advait Balaji, Richard Baraniuk, C. J. Barberan, Ruth Dannenfelser, Chen Dun, Mohammadamin Edrisi, R. A. Leo Elworth, Bryce Kille, Anastasios Kyrillidis, Luay Nakhleh, Cameron R. Wolfe, Zhi Yan, Vicky Yao, and Todd J. Treangen. Current progress and open challenges for applying deep learning across the biosciences. *Nature Communications*, 13(1):1728, April 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29268-7. URL https://www.nature.com/articles/s41467-022-29268-7.

[20] Matthew D. Schwartz. Modern Machine Learning and Particle Physics. *Harvard Data Science Review*, March 2021. doi: 10.1162/99608f92.beeb1183. URL https://hdsr.mitpress.mit.edu/pub/xqle7lat.

[21] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih. Machine Learning in the Search for New Fundamental Physics, 2021. URL https://arxiv.org/abs/2112.03769.

[22] Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. Machine Learning Force Fields. *Chemical Reviews*, 121(16):10142–10186, August 2021. ISSN 0009-2665, 1520-6890. doi: 10.1021/acs.chemrev.0c01111. URL https://pubs.acs.org/doi/10.1021/acs.chemrev.0c01111.

[23] Guanjie Wang, Changrui Wang, Xuanguang Zhang, Zefeng Li, Jian Zhou, and Zhimei Sun. Machine learning interatomic potential: Bridge the gap between small-scale models and realistic device-scale simulations. *iScience*, 27(5):109673, May 2024. ISSN 25890042. doi: 10.1016/j.isci.2024.109673. URL https://linkinghub.elsevier.com/retrieve/pii/S2589004224008952.

[24] Shams Mehdi, Zachary Smith, Lukas Herron, Ziyue Zou, and Pratyush Tiwary. Enhanced Sampling with Machine Learning. *Annual Review of Physical Chemistry*, 75(1):347–370, June 2024. ISSN 0066-426X, 1545-1593. doi: 10.1146/annurev-physchem-083122-125941. URL https://www.annualreviews.org/content/journals/10.1146/annurev-physchem-083122-125941.

[25] Ao Ma and Aaron R. Dinner. Automatic Method for Identifying Reaction Coordinates in Complex Systems. *The Journal of Physical Chemistry B*, 109(14):6769–6779, April 2005. ISSN 1520-6106, 1520-5207. doi: 10.1021/jp045546c. URL https://pubs.acs.org/doi/10.1021/jp045546c.

[26] Baron Peters and Bernhardt L. Trout. Obtaining reaction coordinates by likelihood maximization. *The Journal of Chemical Physics*, 125(5):054108, August 2006. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.2234477. URL https://pubs.aip.org/jcp/article/125/5/054108/909111/Obtaining-reaction-coordinates-by-likelihood.

[27] Pavan Ravindra, Zachary Smith, and Pratyush Tiwary. Automatic mutual information noise omission (AMINO): generating order parameters for molecular systems. *Molecular Systems Design & Engineering*, 5(1):339–348, 2020. ISSN 2058-9689. doi: 10.1039/C9ME00115H. URL https://xlink.rsc.org/?DOI=C9ME00115H.

[28] Ferry Hooft, Alberto Pérez De Alba Ortíz, and Bernd Ensing. Discovering Collective Variables of Molecular Transitions via Genetic Algorithms and Neural Networks. *Journal of Chemical Theory and Computation*, 17(4):2294–2306, April 2021. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.0c00981. URL https://pubs.acs.org/doi/10.1021/acs.jctc.0c00981.

[29] Frank Noé and Feliks Nüske. A Variational Approach to Modeling Slow Processes in Stochastic Dynamical Systems. *Multiscale Modeling & Simulation*, 11(2):635–

655, January 2013. ISSN 1540-3459, 1540-3467. doi: 10.1137/110858616. URL http://epubs.siam.org/doi/10.1137/110858616.

[30] Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. Identification of slow molecular order parameters for Markov model construction. *The Journal of Chemical Physics*, 139(1):015102, July 2013. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.4811489. URL https://pubs.aip.org/jcp/article/139/1/015102/192538/Identification-of-slow-molecular-order-parameters.

[31] Maxwell I. Zimmerman and Gregory R. Bowman. FAST Conformational Searches by Balancing Exploration/Exploitation Trade-Offs. *Journal of Chemical Theory and Computation*, 11(12):5747–5757, December 2015. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.5b00737. URL https://pubs.acs.org/doi/10.1021/acs.jctc.5b00737.

[32] Zahra Shamsi, Kevin J. Cheng, and Diwakar Shukla. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. *The Journal of Physical Chemistry B*, 122(35):8386–8395, September 2018. ISSN 1520-6106, 1520-5207. doi: 10.1021/acs.jpcb.8b06521. URL https://pubs.acs.org/doi/10.1021/acs.jpcb.8b06521.

[33] Adrià Pérez, Pablo Herrera-Nieto, Stefan Doerr, and Gianni De Fabritiis. AdaptiveBandit: A Multi-armed Bandit Framework for Adaptive Sampling in Molecular Simulations. *Journal of Chemical Theory and Computation*, 16(7):4685–4693, July 2020. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.0c00205. URL https://pubs.acs.org/doi/10.1021/acs.jctc.0c00205.

[34] Thomas Stecher, Noam Bernstein, and Gábor Csányi. Free Energy Surface Reconstruction from Umbrella Samples Using Gaussian Process Regression. *Journal of Chemical Theory and Computation*, 10(9):4079–4097, September 2014. ISSN 1549-9618, 1549-9626. doi: 10.1021/ct500438v. URL https://pubs.acs.org/doi/10.1021/ct500438v.

[35] Raimondas Galvelis and Yuji Sugita. Neural Network and Nearest Neighbor Algorithms for Enhancing Sampling of Molecular Dynamics. *Journal of Chemical Theory and Computation*, 13(6):2489–2500, June 2017. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.7b00188. URL https://pubs.acs.org/doi/10.1021/acs.jctc.7b00188.

[36] Ashley Z. Guo, Emre Sevgen, Hythem Sidky, Jonathan K. Whitmer, Jeffrey A. Hubbell, and Juan J. De Pablo. Adaptive enhanced sampling by force-biasing using neural networks. *The Journal of Chemical Physics*, 148(13):134108, April 2018. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.5020733. URL https://pubs.aip.org/jcp/article/148/13/134108/75750/Adaptive-enhanced-sampling-by-force-biasing-using.

[37] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014. URL https://arxiv.org/abs/1406.2661.

[38] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2013. URL https://arxiv.org/abs/1312.6114.

[39] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods, June 2020. URL http://arxiv.org/abs/1908.09257. arXiv:1908.09257.

[40] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, March 2010. ISSN 1539-6746, 1945-0796. URL https://projecteuclid.org/journals/communications-in-mathematical-sciences/volume-8/issue-1/Density-estimation-by-dual-ascent-of-the-log-likelihood/cms/1266935020.full.

[41] E. G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, February 2013. ISSN 0010-3640, 1097-0312. doi: 10.1002/cpa.21423. URL https://onlinelibrary.wiley.com/doi/10.1002/cpa.21423.

[42] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP, February 2017. URL http://arxiv.org/abs/1605.08803. arXiv:1605.08803.

[43] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows, 2015. URL https://arxiv.org/abs/1505.05770.

[44] Peter Wirnsberger, Andrew J. Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14):144112, October 2020. ISSN 0021-9606, 1089-7690. doi: 10.1063/5.0018903. URL https://pubs.aip.org/jcp/article/153/14/144112/316574/Targeted-free-energy-estimation-via-learned.

[45] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365 (6457):eaaw1147, September 2019. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aaw1147. URL https://www.science.org/doi/10.1126/science.aaw1147.

[46] Sebastian Falkner, Alessandro Coretti, Salvatore Romano, Phillip L Geissler, and Christoph Dellago. Conditioning Boltzmann generators for rare event sampling. *Machine Learning: Science and Technology*, 4(3):035050, September 2023. ISSN 2632-2153. doi: 10.1088/2632-2153/acf55c. URL https://iopscience.iop.org/article/10.1088/2632-2153/acf55c.

[47] Hendrik Jung, Roberto Covino, A. Arjun, Christian Leitold, Christoph Dellago, Peter G. Bolhuis, and Gerhard Hummer. Machine-guided path sampling to discover mechanisms of molecular self-organization. *Nature Computational Science*, 3(4): 334–345, April 2023. ISSN 2662-8457. doi: 10.1038/s43588-023-00428-z. URL https://www.nature.com/articles/s43588-023-00428-z.

[48] David Landau and Kurt Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 5 edition, July 2021. ISBN 9781108780346 9781108490146. doi: 10.1017/9781108780346. URL https://www.cambridge.org/core/product/identifier/9781108780346/type/book.

[49] Mark E. Tuckerman. *Statistical mechanics: theory and molecular simulation*. Oxford graduate texts. Oxford university press, Oxford, 2nd ed edition, 2023. ISBN 9780198825562.

[50] Hans C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of Chemical Physics*, 72(4):2384–2393, February 1980. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.439486. URL https://pubs.aip.org/jcp/article/72/4/2384/218722/Molecular-dynamics-simulations-at-constant.

[51] Shuichi Nosé and M.L. Klein. Constant pressure molecular dynamics for molecular systems. *Molecular Physics*, 50(5):1055–1076, December 1983. ISSN 0026-8976, 1362-3028. doi: 10.1080/00268978300102851. URL http://www.tandfonline.com/doi/abs/10.1080/00268978300102851.

[52] Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*, 81(1):511–519, July 1984. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.447334. URL https://pubs.aip.org/jcp/article/81/1/511/607222/A-unified-formulation-of-the-constant-temperature.

[53] William G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Physical Review A*, 31(3):1695–1697, March 1985. ISSN 0556-2791. doi: 10.1103/PhysRevA.31.1695. URL https://link.aps.org/doi/10.1103/PhysRevA.31.1695.

[54] B. Leimkuhler and C. Matthews. Rational Construction of Stochastic Numerical Methods for Molecular Sampling. *Applied Mathematics Research eXpress*, page abs010, June 2012. ISSN 1687-1200, 1687-1197. doi: 10.1093/amrx/abs010. URL https://academic.oup.com/amrx/article-lookup/doi/10.1093/amrx/abs010.

[55] The Editors of Encyclopaedia Britannica. Monte Carlo method, March 2025. URL https://www.britannica.com/science/Monte-Carlo-method.

[56] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.1699114. URL https://pubs.aip.org/jcp/article/21/6/1087/202680/Equation-of-State-Calculations-by-Fast-Computing.

[57] Johannes Kästner. Umbrella sampling. *WIREs Computational Molecular Science*, 1(6):932–942, November 2011. ISSN 1759-0876, 1759-0884. doi: 10.1002/wcms.66. URL https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.66.

[58] Shankar Kumar, John M. Rosenberg, Djamal Bouzida, Robert H. Swendsen, and Peter A. Kollman. THE weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *Journal of Computational Chemistry*, 13(8): 1011–1021, October 1992. ISSN 0192-8651, 1096-987X. doi: 10.1002/jcc.540130812. URL https://onlinelibrary.wiley.com/doi/10.1002/jcc.540130812.

[59] Christoph Dellago, Peter G. Bolhuis, Félix S. Csajka, and David Chandler. Transition path sampling and the calculation of rate constants. *The Journal of Chemical Physics*, 108(5):1964–1977, February 1998. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.475562. URL https://pubs.aip.org/jcp/article/108/5/1964/182970/Transition-path-sampling-and-the-calculation-of.

[60] C. Dellago, P.G. Bolhuis, and P.L. Geissler. Transition Path Sampling Methods. In Mauro Ferrario, Giovanni Ciccotti, and Kurt Binder, editors, *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*, volume 703, pages 349–391. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 9783540352709. doi: 10.1007/3-540-35273-2_10. URL http://link.springer.com/10.1007/3-540-35273-2_10.

[61] Peter G. Bolhuis, David Chandler, Christoph Dellago, and Phillip L. Geissler. Transition Path Sampling: Throwing Ropes Over Rough Mountain Passes, in the Dark. *Annual Review of Physical Chemistry*, 53(1):291–318, October 2002. ISSN 0066-426X, 1545-1593. doi: 10.1146/annurev.physchem.53.082301.113146. URL https://www.annualreviews.org/doi/10.1146/annurev.physchem.53.082301.113146.

[62] Christoph Dellago, Peter G. Bolhuis, and Phillip L. Geissler. Transition Path Sampling. In I. Prigogine and Stuart A. Rice, editors, *Advances in Chemical Physics*, volume 123, pages 1–78. Wiley, 1 edition, July 2002. ISBN 9780471214533 9780471231509. doi: 10.1002/0471231509.ch1. URL https://onlinelibrary.wiley.com/doi/10.1002/0471231509.ch1.

[63] Sebastian Falkner, Alessandro Coretti, Baron Peters, Peter G. Bolhuis, and Christoph Dellago. Revisiting Shooting Point Monte Carlo Methods for Transition Path Sampling, 2024. URL https://arxiv.org/abs/2408.03054.

[64] Ajith Abraham. Nature and Scope of AI Techniques. In Peter H. Sydenham and Richard Thorn, editors, *Handbook of Measuring System Design*. Wiley, 1 edition, February 2005. ISBN 9780470021439 9780471497394. doi: 10.1002/0471497398.mm420. URL https://onlinelibrary.wiley.com/doi/10.1002/0471497398.mm420.

[65] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471,

0033-295X. doi: 10.1037/h0042519. URL https://doi.apa.org/doi/10.1037/h0042519.

[66] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14-15):2627–2636, August 1998. ISSN 13522310. doi: 10.1016/S1352-2310(97)00447-0. URL https://linkinghub.elsevier.com/retrieve/pii/S1352231097004470.

[67] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90020-8. URL https://linkinghub.elsevier.com/retrieve/pii/0893608089900208.

[68] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0.

[69] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016. URL https://arxiv.org/abs/1609.04747.

[70] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. ISSN 1533-7928. URL http://jmlr.org/papers/v12/duchi11a.html.

[71] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method, 2012. URL https://arxiv.org/abs/1212.5701.

[72] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014. URL https://arxiv.org/abs/1412.6980.

[73] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation, 2014. URL https://arxiv.org/abs/1410.8516.

[74] Landan Zhang, Sylwia Bujkiewicz, and Dan Jackson. Three new methodologies for calculating the effective sample size when performing population adjustment. *BMC Medical Research Methodology*, 24(1):287, November 2024. ISSN 1471-2288. doi: 10.1186/s12874-024-02412-1. URL https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-024-02412-1.

[75] Leslie Kish. *Survey Sampling*. John Wiley & Sons, New York, January 1965. ISBN 978-0-471-48900-9.

[76] Saul Wolfe, H. Bernhard Schlegel, Imre G. Csizmadia, and Fernando Bernardi. Chemical dynamics of symmetric and asymmetric reaction coordinates. *Journal*

of the American Chemical Society, 97(8):2020–2024, April 1975. ISSN 0002-7863, 1520-5126. doi: 10.1021/ja00841a005. URL https://pubs.acs.org/doi/abs/10.1021/ja00841a005.

[77] Wolfgang Quapp. A growing string method for the reaction pathway defined by a Newton trajectory. *The Journal of Chemical Physics*, 122(17):174106, May 2005. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.1885467. URL https://pubs.aip.org/jcp/article/122/17/174106/922109/A-growing-string-method-for-the-reaction-pathway.

[78] Joseph C. Kim, David Bloore, Karan Kapoor, Jun Feng, Ming-Hong Hao, and Mengdi Wang. Scalable Normalizing Flows Enable Boltzmann Generators for Macromolecules, 2024. URL https://arxiv.org/abs/2401.04246.

[79] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-Spline Flows, 2019. URL https://arxiv.org/abs/1906.02145.

[80] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows, 2019. URL https://arxiv.org/abs/1906.04032.

[81] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

[82] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. URL https://arxiv.org/abs/2006.11239.

[83] Luke Triplett and Jianfeng Lu. Diffusion methods for generating transition paths. *Journal of Computational Physics*, 522:113590, February 2025. ISSN 00219991. doi: 10.1016/j.jcp.2024.113590. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999124008386.

List of Figures

List of Tables

A. Appendix

A.1. Linear latent space interpolation

Figure A.1 shows the effect of choosing initial samples only in the stable states. After the first cycle, the generator performs a linear interpolation between the stable states, since it has no information about this region from the training data. In the 2D system, this does not pose much of a problem, but in higher dimensional systems, these samples would be unphysical and have exceedingly high energies.

A.2. GenAIMMD parameters

The parameters used to test the performance of GenAIMMD are listed in table A.1 for the 2D model and in table A.2 for the polymer model.

Table A.1.: Parameters used for running GenAIMMD in the 2D system

	Boltzmann Generator	**Committor model**	**GenAIMMD**
Hidden layers	3	2	
Nodes per layer	100	20, 10	
Learning Rate	10^{-4}	10^{-4}	
Batch size	100	100	
Epochs per cycle	200	200	
RealNVP blocks	3		
w_{KL}	1		
w_{ML}	1		
Cycles			50
D			1.0
Δt			10^{-2}
MD steps			500
k_{bias}			1500
Queue size			$2 \cdot N_{\mathrm{samples}}$
N_{samples}			300

Table A.2.: Parameters used for running GenAIMMD in the polymer system

	Boltzmann Generator	**Committor model**	**GenAIMMD**
Hidden layers	3	3	
Nodes per layer	200	64, 32, 16	
Epochs per cycle	100, 50, 25	50	
Learning Rate	$5 \cdot 10^{-4}$, 10^{-4}, 10^{-4}	10^{-3}	
Batch size	250	150	
RealNVP blocks	8		
w_{KL}	0, 10^{-4}, 10^{-3}		
w_{ML}	1		
U_{clamp}	1		
U_{max}	10^{20}		
Cycles			40
D			1.0
Δt			10^{-4}
MD steps			3000
k_{bias}			2.5
Queue size			$2 \cdot N_{\mathrm{samples}}$
N_{samples}			6000

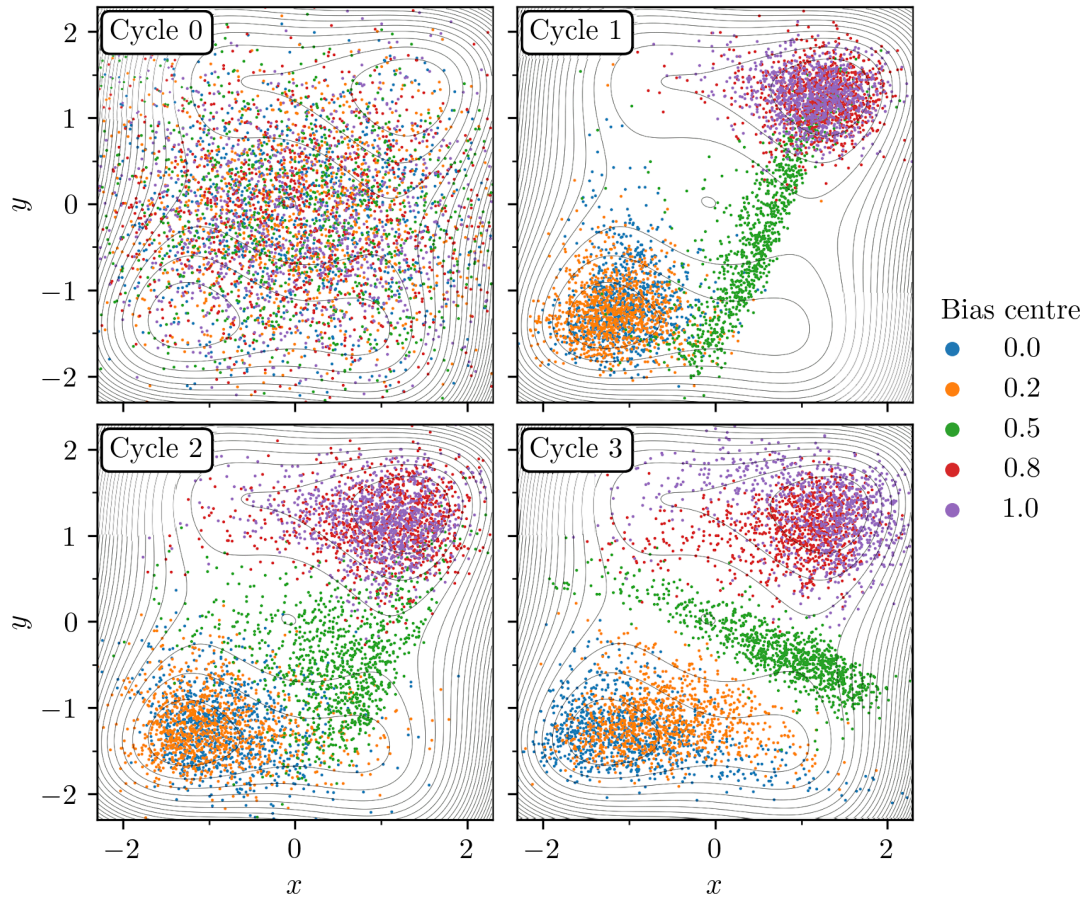


Figure A.1.: Samples produced by the Boltzmann Generator in the 2D system after each training cycle of GenAIMMD. *Cycle 0* refers to the untrained Boltzmann Generator, which produces essentially randomly distributed samples. The initial samples were only chosen in the stable states, which results in the linear interpolation between the two states for bias centre 0.5 in cycle 1.