Sampling Rare Conformational Transitions of Organic Molecules With Path Sampling

Matteo Bonacini

Abstract

Transition Path Sampling (TPS) is a technique for molecular dynamics simulations that allows us to study *rare events*: events that happen fast, but that are only observed on large time scales. In this text, I will give an introduction on what TPS is and how it is related to Monte Carlo simulations. I will then present a real world application of TPS: the computation of reaction rates for conformational transitions in the alanine dipeptide organic molecule. The results I obtain are comparable to those of previous works.

CONTENTS

1 Introduction

Molecular Dynamics (MD) simulations allow us to sample events with a time step of the scale of $1fs$. Even the most powerful supercomputers are only able to simulate events in the scale of $1ms$. Often, dynamical process of interest occur on time scale that are much longer that the one we have access to; even if the event in itself happens very fast (at scales that we have access to). These are called *rare events*. Rare events are characterised by the presence of a *bottleneck*, in the form of a high energy barrier that separates two (or more) meta-stable states. Some examples of rare events include the nucleation of first order phase transitions, chemical reactions in solutions and protein folding. Transition Path Sampling (TPS) was developed as a methodology to study such events.

TPS uses a statystical mechanics approach: instead of looking for a single, well defined reaction path, the idea is to construct a collection of many different valid paths: the *transition path ensemble*. The strength of this method is that it does not require any a priori knowledge on the reaction process; the only thing required is a precise definition of the stable states. Moreover, the elements of the ensemble are *real physical trajectories*, not subject to any unphysical bias. Methods available prior to the introduction of TPS would require a deeper a priori knowledge of the system at hand, as they would require either a precise knowledge of the free energy surface or some knowledge of a reaction coordinate that could be used to force the system towards taking a reactive trajectory.

There are many useful results that can be obtained with TPS. The most common example is the computation of reaction rate constants which, in principle, are quantities that can be measured experimentally. My goal for this text is to show how to do just that. First, I will explain how the TPS method works and then, I will show how to apply it to the alanine dipeptide molecule to obtain reaction rate constants between stable conformational states.

2 The Monte Carlo method with importance sampling

In this section I will briefly explain the main concepts of Monte Carlo with importance sampling. These concepts are the core on which TPS is based on.

2.1 Why do we need the Monte Carlo method?

The Monte Carlo method was originally developed by S. Ulam [1] as a general method for solving a certain class of integro-diffential equations[1]. In the context of statistical mechanics, the need for the Monte Carlo method arises when we need to compute ensemble averages.

Let us consider a system whose partition function is given by

$$Q = \int_{\mathcal{M}} e^{-\beta H} d\Omega, \tag{1}$$

where $\beta = (kT)^{-1}$, $\mathcal{M} \ni (q, p)$ is the phase-space of the system, $H = H(q, p)$ is the Hamiltonian operator of the system and $d\Omega$ is the integration measure in phase space, usually defined as $d\Omega = (h^{3N} N!)^{-1} d^{3N}q \; d^{3N}p$ for a system of N identical particles. Given an observable $A = A(q, p)$ of the system, we can compute its ensemble average using the formula

$$\langle A \rangle = \frac{1}{Q} \int_{\mathcal{M}} A e^{-\beta H} d\Omega. \tag{2}$$

Now, if A is a function of the momenta only, we can separate off the integral in (2) over each momenta and over the coordinates and perform the computation analitically[2]. If A is a function of the coordinates, however, we are left with an expression of the form

$$\langle A \rangle = \frac{\int A e^{-\beta V(q)} \; d^{3N}q}{\int e^{-\beta V(q)} \; d^{3N}q}, \tag{3}$$

for which we have to resort to numerical computation.

Traditional numerical integration techniques resort to subdividing the integration space into a lattice of a fixed number of points m for each coordinate and momenta. The integrand function then needs to be evaluated at each of the m^{3N} different points and the results are added up to get the final value of the integral. The problem here is that the computational complexity goes with $\mathcal{O}(e^N)$; that is, it scales exponentially in the number of particles of the system. This type of problem is intractable by modern classical computers even for small systems[3], and thus we resort to the Monte Carlo method.

2.2 The Monte Carlo Method and its limitations

The basic idea behind the Monte Carlo method is the following: let us suppose that we want to compute an integral of the form

$$I = \int_V f(x) \; dx. \tag{4}$$

We use the average value theorem to rewrite I as

$$I = V \; \langle f(x) \rangle_x, \tag{5}$$

and the Monte Carlo method consists in computing the average over x in (5) *by choosing a random set of points $x \in V$ and taking the average of $f(x)$ over those points*.

The sampling points do not need to be uniformly distributed; it is usually more useful to perform the sampling according to a different distribution $w(x)$. To do so, we start by rewriting the integral in (4) as

$$I = \int_V w(x) \frac{f(x)}{w(x)} \; dx.$$

Now, we assume that $w(x)$ can be expressed as the divergence of a vector function $W(u)$. This allows us to

1. Metropolis et. al., in their 1953 paper [2], state that the method was concurrently developed by J. E. Mayer. However, I was not able to find the publication they were referring to.

2. We can do this since H is a quadratic function of the momenta.

3. For example, a system of $N = 25$ particles in three dimensions would need to be evaluated at $\sim e^{75}$ points. Modern processors run at a clock frequency of the order of $1GHz$. If a processor could perform a function evaluation for every clock cycle, we would need to wait for $\sim 10^6$ years to get the result of the full computation.

perform a change of variables, so that we are left with the following expression for I:

$$I = \int_{U \ni u} \frac{f(u)}{w(u)} du. \tag{6}$$

In (6) the integration in du rather than dx tells us that, when computing Monte Carlo averages of the type $\langle f/w \rangle_u$, we must sample the points according to the distribution $w(x)$ and not uniformly.

Now, the problems arise when we compute the variance σ_I^2 of (6). As a matter of fact, one can show that, after sampling N points, a reasonable estimate for the variance is given by

$$\sigma_I^2 \approx \frac{V^2}{N} \left(\langle f^2/w^2 \rangle_x - \langle f/w \rangle_x^2 \right). \tag{7}$$

This equation tells us that, if we want to compute I exactly, we would need to sample points according to a distribution that makes f/w a constant. In most cases, it is impossible to find such a distribution, and we are left with an error estimate that goes with the inverse square root of N. Press et. al. [3] go in great detail in showing how to obtain an optimal w and how it is paradoxical to assume that we can obtain it, as doing so would require knowing the integral that we want to compute. They also describe techniques to reach convergence faster, but still in polinomial time.

The problem with polinomial scaling is that phase spaces have very large volumes. For example, a system of 100 noninteracting spins has 2^{100} possible configurations. This would mean that to get a variance of the order of unity, we would need to sample 2^{100} points. Essentially, we would be wasting time by computing the partition function of the system explicitly. This problem led to the development of the Metropolis criterion [2] which, under some circumstances, enables us to perform computations without the need for computing the partition function of the entire system.

2.3 The Metropolis Criterion

The Metropolis Criterion (also called *importance sampling*) is a strategy that allows us to efficiently sample ensemble averages that are expressed as a *ratio* of two integrals, in the form of (3). The strategy relies on the fact that states in the canonical ensemble are weighted according to the Boltzmann distribution, and this allows us to compute averages even if we do not know the partition function of the system. We start from (3), and rewrite both integrals using (5):

$$\begin{aligned} \langle A \rangle &= \frac{\langle A e^{-\beta V} \rangle}{\langle e^{-\beta V} \rangle} \\ &= \frac{\sum_{i=1}^{N} A_i e^{-\beta V_i}}{\sum_{i=1}^{N} e^{-\beta V_i}} \end{aligned} \tag{8}$$

where N is the number of sampling points. Now, if we sample the points according to the Boltzmann distribution, we need them to be weighted accordingly. The ensable average is thus computed as:

$$\langle A \rangle = \frac{\sum_{i'=1}^{N} \frac{1}{P_i} A_i e^{-\beta V_i}}{\sum_{i'=1}^{N} \frac{1}{P_i} e^{-\beta V_i}},$$

where I wrote i' instead of i to indicate that the sampling points are distributed differently from (8). However, since $P_i = e^{-\beta V_i}$, the exponential terms cancel out:

$$\begin{aligned} \langle A \rangle &= \frac{\sum_{i'=1}^{N} A_i}{\sum_{i'=1}^{N} 1} \\ &= \frac{1}{N} \sum_{i'=1}^{N} A_i. \end{aligned} \tag{9}$$

This last equation does not depend on the partition function of the system. This means that, *if we can sample states according to the Boltzmann distribution, we can compute ensemble averages easily!*

Sampling states according to the Boltzmann distribution

There are many ways of sampling states according to the Boltzmann distribution. In their original paper. Metropolis et. al. [2] proved that the following algorithm can be used:

1) Chose a starting configuration, denoted by r^N.
2) Generate an arbitrarily small random displacement, called *trial move*, δr^N.
3) Compute the change of energy ΔV that happens when the system is displaced by δr^N.
4) If $\Delta V < 0$, we accept this displacement and the current configuration becomes $r^N + \delta r^N$. Otherwise, we accept the displacement only with probability $e^{-\beta \Delta V}$. If the displacement is rejected, the current configuration remains the same: r^N.
5) Whether the move was accepted or not, we consider the resulting configuration as a new one, for the purpose of taking averages[4].
6) We repeat these steps until we have generated a satisfactory number of trial moves.

This algorithm can be generalised for system that are not described with the canonical ensemble, provided that one can find a quantity S analogue to $\beta \Delta V$ such that the states in the ensemble one wants to use are distributed proportionally to e^{-S}.

2.4 Monte Carlo sampling in different ensembles

As said before, when performing calculations using the Monte Carlo method, we are not required to work in the canonical ensemble of equation (1). As a matter of fact, Transition Path Sampling relies on the *path ensemble* and on the *transition state ensemble*. A sufficient condition that allows us to obtain a valid Monte Carlo sampling is the *detailed balance condition*[5][4].

The condition of detailed balance reads

$$\phi(o \to n) = \phi(n \to o), \tag{10}$$

where ϕ is the flow between two configurations, given by

$$\phi(o \to n) = \rho(o) \cdot P_{gen}(o \to n) \cdot P_{acc}(o \to n), \tag{11}$$

4. That is, we compute A_i in (9) using the current configuration, whether the move was accepted or not.

and where $\rho(o)$ is the probability of being in configuration o, $P_{gen}(o \to n)$ is the probability of generating the configuration n starting from o and $P_{acc}(o \to n)$ is the probability of accepting the move from o to n.

Procedure to obtain a valid algorithm

The general procedure that we can use to design a valid Monte Carlo algorithm is as follows:

1) Decide which distribution ρ we want to sample.
2) Impose the condition of detailed balance (10).
3) Determine $P_{gen}(o \to n)$
4) Find any acceptance rule $P_{acc}(x \to y)$ that satisfies equation (11).

Frenkel et. al. [4] show various examples on how this procedure can be used to check for the validity of Monte Carlo algorithm. In subsection 3.2 I will use this procedure to show how one can derive the acceptance rule in the path ensemble.

3 Transition Path Sampling

The TPS method was originally proposed by Dellago, Bolhuis et. al. in a series of papers [5], [6], [7], [8]. The core content of those papers is the same; they differ only by the level of formalism used and the different applications they proposed. The contents of this section will contain a discussion of what was proposed in those papers. I will also give an overview of Multiple State Transition Interface Sampling (MSTIS).

3.1 Constructing the Path Ensemble

Let us start by giving some definitions.

Definition 1. A *trajectory* of time duration T

$$\chi(T) = (\chi_0, \chi_{\Delta t}, \chi_{2\Delta t} \dots, \chi_T)$$

is a cronological sequence of phase space points (snapshots) generated by the application of a dynamical propagation rule.

We will denote with $P(\chi(T))$ the probability density to observe a given path from the ensemble of all possible ones. An expression for $P(\chi(T))$ can be derived for different process dynamics, and later we will see how, but we need to introduce some other concepts first.

Defining stable states and reactive pathways

In our study, we are concerned only with *reacting pathways*, that is, pathways that start from a *reactant region* A and end in a *product region* B. These regions are defined through their characteristic functions:

5. The detailed balance condition is sufficient but not necessary. A less stringent necessary and sufficient condition is the balance condition. In practice, it is usually more convenient to impose the detailed balance condition and then, eventually, relax it to the balance condition only if it can lead to faster computation [4].

Definition 2. A region in phase space $A \ni \chi$ is defined by its characteristic function $h_A(\chi)$, defined as

$$h_A(\chi) = \begin{cases} 1 & \iff \chi \in A \\ 0 & \iff \chi \notin A \end{cases}.$$

Usually, the characteristic functions are defined using the value of a low dimensional *order parameter* q. Although some general approaches have been proposed [9], the choice of such parameter is not trivial and it may require a significant degree of experimentation. Some basic requirements that guide the choice of q are the following:

- A and B must be large enough to account for thermal fluctuations at equilibrium, otherwise we might end up discarding potentially valid trajectories.
- The two regions must be located entirely within their respective basins of attraction; that is, if we take a starting point inside the basin of attraction of a region A, its trajectory should relax into the stable state A.

An bad choice of q will yield incorrect results, as described in Figure 1.

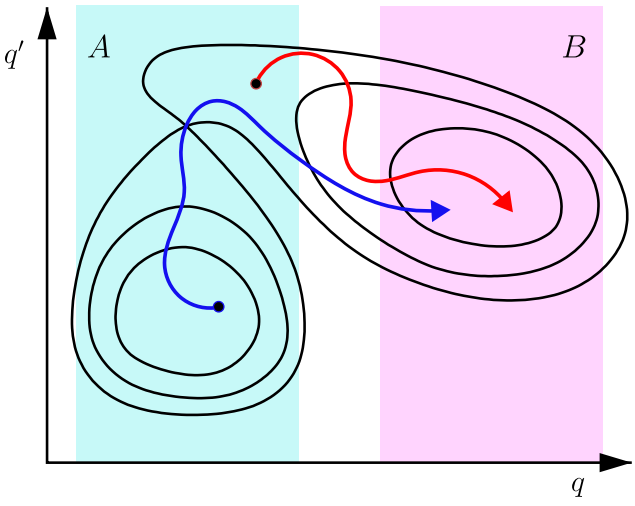

Figure 1: Example of a bad choice for an order parameter. Here, the stable states A and B are defined with the values of q that correspond to the cyan and magenta regions, respectively. A trajectory starting from the basin of attraction in A will need to cross a free energy barrier in order to reach B (blue trajectory). However, there are some points in A that lie outside of its basin of attraction, and that will reach B without crossing a free energy barrier (red trajectory). In this example, a correct choice of an order parameter would need to take the value of q' into consideration as well.

Once we have defined the reactant and product regions, we can restrict the path ensamble to contain only the reactive trajectories. We can do so by introducing a different path probability density:

$$P_{AB}(\chi(T)) = \frac{1}{Q_{AB}(T)} h_A(\chi_0) P(\chi(T)) h_B(\chi_T). \quad (12)$$

In (12), the product of the two characteristic functions h_A and h_B ensures that all the path that do not start in A and end in B will have a zero probability. This reduces the size of the ensamble, and thus we need to normalize the distribution by imposing

$$Q_{AB}(T) = \int h_A(\chi_0) P(\chi(T)) h_B(\chi_T)\, D\chi(T). \quad (13)$$

We will refer to $Q_{AB}(T)$ as the path ensemble partition function.

In the last expression, the notation $D\chi(T)$ is borrowed from path integral theory, and it implies a summation over all possible pathways:

$$D\chi(T) = d\chi_0 d\chi_{\Delta t} \cdots d\chi_T.$$

Different path probability expressions can be obtained for different dynamics. Let us now see some examples.

Markovian dynamics

For a Markov process, the probability of observing the state χ_{k+1} after a time step Δt, depends only on the previous state, χ_k. This means that the total probability of observing a given path is given by the product of single time step transition probabilities:

$$P(\chi(T)) = \rho(\chi_0) \prod_{i=0}^{T/\Delta t - 1} p(\chi_{i\Delta t} \to \chi_{(i+1)\Delta t}).$$

Here, $\rho(\chi_0)$ is the probability distribution of the initial conditions. For a Markov chain, it can be obtained by computing the left eigenvector[6] with unit eigenvalue of the corresponding Markov matrix. Figure 2 provides an explicit example of a path ensemble based on a Markov process.

Hamiltonian dynamics

The time evolution of a Hamiltonian system is deterministic, and can be expressed with the use of the *system propagator* ϕ_t:

$$\chi_t = \phi_t(\chi_0).$$

For such systems, we can write the time step transition probabilities with the help of the Dirac delta function:

$$p(\chi_t \to \chi_{t+\Delta t}) = \delta\left[\chi_{t+\Delta t} - \phi_{\Delta t}(\chi_t)\right]$$

and thus, the total path probability is

$$P(\chi(T)) = \rho(\chi_0) \prod_{i=0}^{T/\Delta t - 1} \delta\left[\chi_{(i+1)\Delta t} - \phi_{\Delta t}(\chi_i \Delta t)\right].$$

For Hamiltonian dynamics we can actually write the partition function of the path ensemble in a neat way, by integrating over all but the first states of the pathway using (13). The result is an expression which depends only on the initial conditions:

$$Q_{AB}(T) = \int \rho(\chi_0) h_A(\chi_0) h_B(\chi_T) d\chi_0. \quad (14)$$

6. A row vector π^* is a left eigenvector of matrix M if and only if $\pi^* M = \pi^* \lambda$, with λ being a scalar.

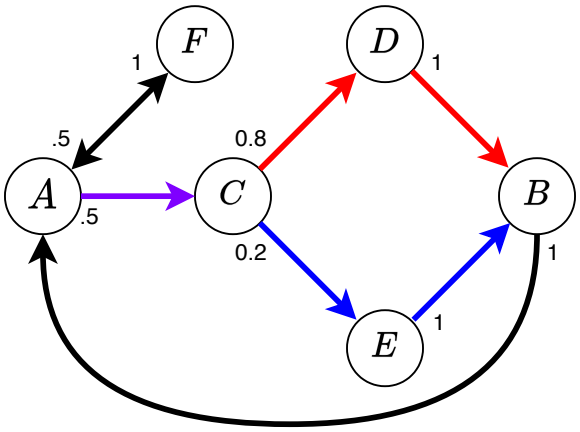

Figure 2: Example construction of path ensemble in a Markov chain. The numbers next to the links indicate the probability of an outgoing transition. Here, we want to consider only the path of length 4 that start in A and end in B. There are only two possible paths (highlighted in red and blue; the link $A \to C$ is taken by both paths) that satisfy those constraints. The partition function of the path ensemble would be $Q = \sum \rho(\chi_1) \times p(\chi_1 \to \chi_2) \times p(\chi_2 \to \chi_3) \times p(\chi_3 \to \chi_4)$, where the summation is over $\chi_1 = \{A\}, \chi_2, \chi_3 = \{C, D, E, F\}, \chi_4 = \{B\}$. Of course, all of the zero probabilities cancel out, and thus the summation reduces to $Q = \rho(A) \times p(A \to C) \times [p(C \to D) \times p(D \to B) + p(C \to E) \times p(E \to B)] = 1/3 \times 1/2 \times (1/5 + 4/5) = 1/6$.

3.2 Sampling the Path Ensemble

Sampling the Path Ensemble means collecting a number of reactive trajectories, weighted accordingly. We can do so by using the Monte Carlo method with the Metropolis criterion. Each Monte Carlo step consists in generating a new path $n = \chi^{new}(T)$, starting from an old one, $o = \chi^{old}(T)$ and then deciding whether to accept or reject the new path.

Path acceptance criterion

In order to obtain a correct sampling, we start by imposing the detailed balance condition. Namely, we require that the probability of observing a move from an old path to a new one is the same as the probability of observing the opposite move:

$$P_{AB}(o)\, \pi(o \to n) = P_{AB}(n)\, \pi(n \to o). \quad (15)$$

In (15), $\pi(o \to n)$ is the probability to perform a move from the old to the new path, which is given by the product of the probability of generating the new path and the probability of accepting it, as explained before in subsection 2.4:

$$\pi(o \to n) = P_{gen}(o \to n)\, P_{acc}(o \to n). \quad (16)$$

Thus, by combining (15) and (16), we get the condition that must be obeyed by the acceptance probability:

$$\frac{P_{acc}(o \to n)}{P_{acc}(n \to o)} = \frac{P_{AB}(n)\, P_{gen}(n \to o)}{P_{AB}(o)\, P_{gen}(o \to n)}. \quad (17)$$

Now, we are allowed to chose any acceptance probability between two paths $P(x \to y)$ that satisfies (17). The most common choice is the one made by Metropolis [2]:

$$P_{acc}(x \to y) = \min\left\{1, \frac{P_{AB}(y)\ P_{gen}(y \to x)}{P_{AB}(x)\ P_{gen}(x \to y)}\right\}. \quad (18)$$

One can easily verify that (18) satisfies (17) by substituting it in the left hand side of the latter[7].

Now, we substitute (12) into (18) and, by using the fact that the old trajectory is reactive and by using $n_i = \chi_i^{new}$, we get the final expression for the acceptance probability:

$$P_{acc}(o \to n) = \\ = h_A(n_0)h_B(n_T)\ \min\left\{1, \frac{P(n)\ P_{gen}(n \to o)}{P(o)\ P_{gen}(o \to n)}\right\} \quad (19)$$

In practice, we can implement (19) in the following way:

1) We generate a new path from the current path (we will see how in the next section).
2) If the new path *is not* reactive, it is rejected.
3) If the new path *is* reactive, we compute the ratio $P(n)\ P_{gen}(n \to o)/P(o)\ P_{gen}(o \to n)$. If it is equals to 1 or greater, we accept the new path. If it is less than one, then we randomly accept or reject it with probability of acceptance given by the ratio itself.
4) If, in any steps, the path was rejected, then we count the old configuration again and start over.
5) If the path was accepted, then we count the new path, we set it as current path and we start over.

3.3 Path generation

Up until now, we assumed that

- we can generate new reactive paths starting from an old one, and
- we have at least one reactive path to start with.

We will now see how these assumptions are justified.

Generating new paths

There are numerous robust methods that can generate new reactive paths. Dellago et. al. originally proposed a series of algorithms based on MD simulations [10], which are broadly used. Given that MD simulations are a computationally intensive task, more recent proposals are moving away from MD altogether, looking to work in a lower dimensional embedding space to generate new pathways [11], [12]. Quantum computation algorithms[8]have been proposed as well [12], [13].

7. Then it's just a matter of applying the definition of min and checking both possible cases.

8. In reality, the goal of quantum mechanical algorithms is not so much of generating *new* trajectories, but it is of generating *uncorrelated* trajectories. We will see more about that in a later section.

Shooting and shifting moves

The *shooting move* and *shifting move* are the two most common ways that can be used to generate a new path. I will only give a brief introduction; for a more in-depth discussion, please refer to [10] and [14].

The basic idea of the shooting move is to take a random point in a valid trajectory and, from that, shoot off a new trajectory by slightly altering the dynamics and integrating the altered trajectory both forwards and backwards in time. The probability $P_{gen}(o \to n)$ for a shooting move is given by the product of three terms

$$P_{gen}(o \to n) = p_{gen}(o_{\bar{t}} \to n_{\bar{t}}) \quad (20a)$$
$$\times \prod_{i=\bar{t}/\Delta t}^{T/\Delta t - 1} p(n_{i\Delta t} \to n_{(i+1)\Delta t}) \quad (20b)$$
$$\times \prod_{i=1}^{\bar{t}/\Delta t - 1} p'(n_{i\Delta t} \to n_{(i-1)\Delta t}), \quad (20c)$$

where (20a) represents the probability to obtain the *shooting point* $n_{\bar{t}}$ by perturbing a starting point $o_{\bar{t}}$, (20b) is the dynamical path weight for the new forward trajectory and (20c) is the dynamical path weight for the backwards trajectory. Please note that in (20c) we need to use an inverse transition probability p', which is, in general, different from p. Inserting (20) into (19) gives us the acceptance probability for paths generated by shooting.

Shooting moves are usually complemented by shifting moves. In this move, a new path is generated from an old one by *shifting* the starting and end point of a trajectory. This is done by simply discarding the first points of a trajectory, and then integrating the end point of the trajectory, so that the total lenght remains the same. One can prove [10] that, for hamiltonian dynamics with canonical or microcanonical distribution of initial conditions, the acceptance probability for shifting moves is 1, provided that the newly generated trajectory is reactive (and 0 otherwise).

Generating an initial path

The matter of generating an initial path is more complex. The textbook approach would be to perform a long MD simulation until we find the first valid path, but this is not always a viable strategy: given that the transition we want to study is a rare event, we may never observe a reactive trajectory. Thus, the most common approach is to bias the system towards taking a reactive trajectory. The biased trajectory can be created, for eaxmple, by simulating the system at a different temperature from the one we want to study it at [15]. Other approaches are based on umbrella sampling [16], in which the trajectories are subject to non physical biasing potentials. A more recent approach [17] uses human intuition to generate biased initial trajectories, by providing a virtual reality environment in which one can visualize the molecule object of study.

If the initial trajectory is biased or non physical, an *equilibration* procedure is then needed to transform it

into one that is more representative of the path ensembemble. This ensures that the trajectories that will be generated from this one are going to be real, physical, unbiased trajectories of the system.

3.4 Computing reaction rates

Let us give a brief definition of what a reaction rate is:

Definition 3. Consider the unimolecular reaction

$$\mathrm{A} \rightleftharpoons \mathrm{B}.$$

We denote with $c_i(t) = N_i(t)/V$ the concentration of chemical species i. Also, the total number of molecules is constant: $N_A + N_B = N$. Provided that the solution is sufficiently diluted, the evolution of the concentrations can be described by a phenomenological rate equation:

$$\dot{c}_A = -k_{AB} c_A + k_{BA} c_B. \tag{21}$$

The quantities k_{AB} and k_{BA} used above are called, respectively, the *forward and backward reaction rate constants*.

It is important to note that equation (21) is a *phenomenological*, *macroscopic* law. In order to calculate the reaction rates from a path sampling simulation, we need to find a way to relate the *microscopic* quantities we obtain from path sampling to the macroscopic law (21).

In their original papers [5], [7], [6], [8], Dellago, Bolhuis et. al. proposed a way of calculating reaction rates starting from a two-state transition path ensemble. More recently, Van Erp. et. al. proposed the new, faster method of *Transition Interface Sampling (TIS)* [18], [19], which works even for multiple state TPS. In this section I will give a brief overview of both methods.

Direct approach from TPS

Here, the formula that connects reaction rates with molecular simulations can be obtained in two ways: either in the context of *linear response theory*, as a consequence of the *fluctuation-dissipation theorem* [20], [21], or in the large perturbations regime [14]. Both approaches lead to the same result:

$$\frac{Q_{AB}(t)}{Q_A} = Q_B \left(1 - e^{-t/\tau}\right), \tag{22}$$

where τ is the time scale of the reaction, Q_{AB} is given by (13)[9] and Q_i is the time-indipendent[10] partition function for the ensemble of all the paths that start in i and end anywhere, namely:

$$Q_i = \int P(\chi(t)) h_i(\chi_0)\, D\chi(t).$$

In (22) the left-hand side is purely microscopic, while the right hand side is macroscopic. Under some assumptions[11], it is possible to show [14] that, for $0 \ll t \ll \tau$, the right-hand side of(22) can be approximated to

$$\frac{Q_{AB}(t)}{Q_A} \approx t k_{AB}$$

and thus *the reaction rate can be computed by taking the time derivative of the quantity in the left-hand side*. This quantity is usually referred to as a *time correlation function*, and it is denoted with $C(t)$:

$$C(t) = \frac{Q_{AB}(t)}{Q_A}. \tag{23}$$

We can actually assign a physical meaning to $C(t)$ by taking the logarithm on both sides of (23):

$$\ln C(t) = \ln \frac{Q_{AB}(t)}{Q_A} = -\Delta F(t).$$

Essentially, the logarithm of the time correlation function is, in modulus, equal to the free energy difference between the two ensembles with partition functions Q_{AB} and Q_A. ΔF is the work that we need to do in order to *compress* the path ensemble and confine to B the endpoints of the path starting from A.

Now, how do we compute $C(t)$? First, we need to assume that we can define an order parameter $\lambda(\chi)$ that can distinguish between the product and reactant state:

$$\int_{\lambda^i_{\min}}^{\lambda^i_{\max}} \delta[\bar{\lambda} - \lambda(\chi)]\, d\bar{\lambda} = h_i(\chi),\ i = \{A, B\}, \tag{24}$$

$$\lambda^A_{\max} < \lambda^B_{\min}.$$

Using (24), and by changing the order of integration, we can rewrite (23) as

$$C(t) = \int_{\lambda^B_{\min}}^{\lambda^B_{\max}} P(\bar{\lambda}, t)\, d\bar{\lambda}, \tag{25}$$

with $P(\bar{\lambda}, t)$ being the probability of finding the system at time t with a certain value of the order parameter $\lambda(\chi_t) = \bar{\lambda}$. It is given by

$$P(\bar{\lambda}, t) = \frac{1}{Q_A} \int h_A(\chi_0) P(\chi(t')) \delta[\bar{\lambda} - \lambda(\chi_t)]\, D\chi(t').$$

Knowing the distribution of $P(\bar{\lambda}, t)$ would allow us to integrate (25) and obtain the time correlation function for different values of t, which could then be used to compute its time derivative numerically. There are ways to speed up this computation [8], but we will not go into detail here.

In order to obtain the $P(\bar{\lambda}, t)$ distribution efficiently, we need to use an algorithm based on umbrella sampling. Without going into too much detail, the idea is that we want to obtain a series of distributions for $P(\bar{\lambda}, t)$ by sampling a series of path ensembles that get progressively squeezed to the Q_{AB} ensemble (as described in Figure 3). After obtaining the series of distributions, we can *match* the different distributions together to obtain the result that we want (see Figure 4). Some methods for matching the different distributions are presented in [16].

It is important to note that, even if we are using an umbrella sampling approach, all the trajectories that

9. With an appropriate change of notation: $T \to t$.

10. In the integral that defines Q_i the time appears as an argument of $\chi(t)$. However, we are integrating on $D\chi(t)$, thus it gets integrated out.

11. Namely, we need to assume that the time scales of molecular motion are comparable with t and much smaller than τ. We also need to assume that the system stays in either A or B states most of the time: $\langle h_A \rangle + \langle h_B \rangle \approx 1$.

we obtain are *real physical trajectories*. By progressively changing the path ensemble we are preserving the real dynamics of the system without adding any unphysical bias.

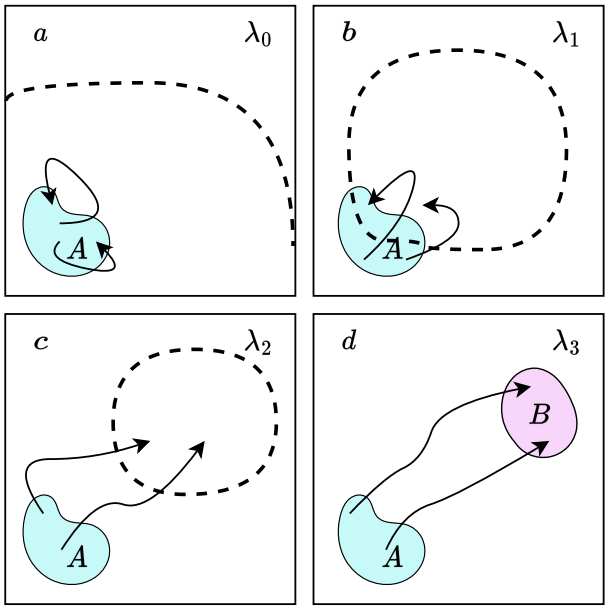

Figure 3: Example of how the path ensemble gets progressively squeezed to the correct one. a) We sample the ensemble of paths that start in A and end almost anywhere on the phase space. b, c) We sample the ensemble of paths that start from A and end in progressively smaller regions of the phase space which converge towards B. d) Finally, we sample the ensemble of paths that start in A and end exactly inside B.

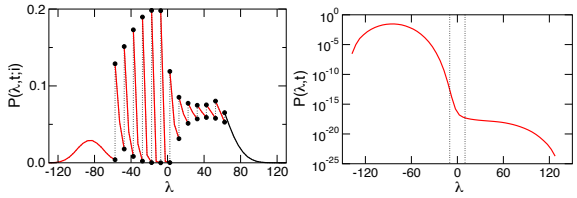

Figure 4: Reprinted from [14]. Example of how histogram matching leads to the sought after distribution of $P(\bar{\lambda}, t)$. Left: series of overlapping $P(\bar{\lambda}, t)$ computed for different ensembles. Right: final histogram obtained through matching.

Transition Interface Sampling approach

The method we just discussed has two main drawbacks which result in wasted computational power:

1) It is designed to work only with paths of a fixed length.
2) $C(t)$ has an oscillatory behaviour which stabilizes only after a long time has passed.

The TIS approach allows us to get rid of these limitations and speed up computing times. We start with the following definition:

Definition 4. A point χ in phase space is defined to be a part of the *overall region* $\mathcal{A}$ of A if and only if a trajectory started in χ and followed backwards in time reaches A before reaching B. The same definition applies for the overall region $\mathcal{B}$.

It can be shown [18] that, if we introduce the (new) time correlation function

$$\mathcal{C}(t) = \frac{Q_{\mathcal{AB}}}{Q_{\mathcal{A}}},$$

then we obtain the following, equivalent expression for the reaction rate k_{AB}:

$$\begin{aligned} k_{AB} &= \dot{\mathcal{C}}(t=0) \\ &= \frac{1}{Q_{\mathcal{A}}} \int h_{\mathcal{A}}(\chi_0) P(\chi(t')) \dot{h}_{\mathcal{B}}(\chi_{t=0})\, D\chi(t'). \end{aligned} \quad (26)$$

The notation I used here indicates that the time derivatives need to be taken at t=0. The integral term in (26) is called *effective positive flux* and it is indicated with $\langle \phi_{AB} \rangle$; it is the average flux into B from trajectories that come from A, without counting recrossings (i.e. trajectories that enter, exit and re-enter B are not counted).

Again, to compute $\langle \phi_{AB} \rangle$ efficiently, we need to use a strategy similar to umbrella sampling. We start from defining an order parameter $\lambda(\chi)$, with the same characteristics as in the previous section. Now, note that any given value $\bar{\lambda}$ of the order parameter defines an (hyper)surface on the phase space.

We now consider a series of n different surfaces[12] $\{\lambda_0, \lambda_1, \ldots, \lambda_n\}$ such that $\lambda(\chi) = \lambda_0$ corresponds the boundary of region A and $\lambda(\chi) = \lambda_n$ corresponds to the boundary of region B. We refer to these surfaces as *interfaces*. For each pair of interfaces, we define the *crossing probabilities* $P_A(\lambda_j|\lambda_i)$:

Definition 5.

$P_A(\lambda_j|\lambda_i)$ = Probability that a path coming from region A and crossing surface λ_i for the first time reaches surface λ_j before coming back to A.

Using this definition, the computation of the reaction rate reduces to the computation of the flux through the first hypersurface (which can be done easily by MD simulations) and the computation of the different crossing probabilites (which can be done easily using TPS):

$$k_{AB} = \frac{\langle \phi_{A1} \rangle}{Q_{\mathcal{A}}} \prod_{i=1}^{n-1} P_A(\lambda_{i+1}|\lambda_i). \quad (27)$$

The TPS computation can be done by defining an ensemble of pathways of variable lengths that satisfy the definition of $P_A(\lambda_j|\lambda_i)$.

Overall, the TIS approach can lead to convergence times which are on order of magnitude faster with respect to the TPS approach [18].

12. Please note that here λ_i indicates both the hypersurface and the value of the order parameter that corresponds to that hypersurface.

3.5 Multiple state Path Sampling

The methods we have discussed up until now work great if there are only two metastable states. The presence of additional metastable states could slow down significantly the computations, as newly generated paths may fall in one of those states and never reach the target state. Rogal and Bolhuis [22] developed the Multiple State Transition Path Sampling (MSTPS) technique to overcome this problem. MSTPS is based on TIS; the main idea behind the technique is to define a new path ensemble that includes trajectories which connect *any two stable states*:

$$P_{\text{MSTPS}}(\chi(T)) = \sum_{i,j\neq i} P_{ij}^{\text{TPS}}(\chi(T)).$$

MSTPS works with path of variable length, and thus the definition of $P_{ij}^{\text{TPS}}(\chi(T))$ varies slightly from the one in (12):

$$P_{ij}^{\text{TPS}}(T) = \frac{1}{Q} h_i(\chi_0) P(\chi(T)) h_j(\chi_T) \prod_k \bar{h}_k(\chi(T)). \tag{28}$$

The factors $\bar{h}_k$ in (28) are needed to specify that all the snapshots of $\chi(T)$, except for the first and the last one, are not within any of the stable states. It can be defined as

$$\bar{h}_k(\chi(T)) = \begin{cases} 0 & \iff \chi_t \in k \\ 1 & \iff \chi_t \notin k \end{cases} \quad \forall t \in]0, T[.$$

From this, we can now generalize what we had seen in section 3.4 to the case of multiple states. The final result for the rate equation between two states will be very similar to (27):

$$k_{ij} = \langle \phi_{mi} \rangle P_i(\lambda_{0,j} | \lambda_{m,i}) \tag{29}$$

Here, each interface is specified by two indexes: for each stable state i, there will be a series of $m+1$ interfaces such that $\lambda_{0,i} = i$ (it coincides with the whole i) and $\lambda_{m,i}$ is the outermost interface relative to i. $\langle \phi_{mi} \rangle$ is the flux through the $\lambda_{m,i}$ interface and $P_i(\lambda_{0,j}|\lambda_{m,i})$ is the probability of reaching the stable state j before re-entering into $\lambda_{m,i}$ (where the path is coming from).

Again, the idea behind (MS)TIS is to break down the computation of (29) into smaller problems. Akin to what we have discussed in section 3.4, we can write

$$\langle \phi_{mi} \rangle = \langle \phi_{0i} \rangle \prod_{s=0}^{m-1} P_i(\lambda_{s+1,i} | \lambda_{s,i}) \tag{30}$$

and for the last term of (29), we can estimate it by counting the total number of pathways starting in i and ending in j and dividing it by the total number of pathways starting in i and crossing the $\lambda_{m,i}$ interface. We write

$$P_i(\lambda_{0,j} | \lambda_{m,i}) \approx \frac{n_{ij}}{\sum_j n_{ij}}. \tag{31}$$

Equations (29), (30) and (31) are all that we need to compute the reaction rates for a system with multiple stable states within a path sampling simulation.

4 Real-world application: Alanine Dipeptide

In this chapter I will start by giving a brief description of the molecule alanine dipeptide. Then, I will show how we can use the methods described in section 3 to perform a path sampling simulation on the molecule in order to obtain reaction rates for its conformational transitions.

The code I used for the simulations is available on my GitHub[13].

4.1 A brief description of Alanine Dipeptide

The *alanine dipeptide* (α-formylaminopropanamide, AD) is one of the primary models used to study theoretical models of backbone conformational equilibria in proteins and organic molecules [23]. The structural formula of the molecule is shown in Figure 5.

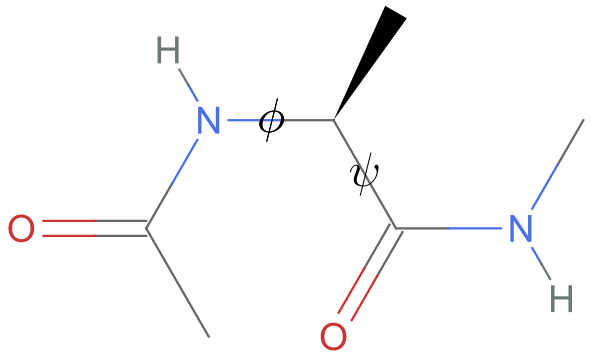

Figure 5: Structural formula of AD. Some hydrogen atoms are omitted. In the picture, the ϕ and ψ angles are highlighted.

The molecule backbone conformation features two degrees of freedom, expressed in terms of the ϕ and ψ torsion angles around the central carbon atom. The molecule also features CH and CO groups, capable of participating in hydrogen bonds with each other or with the solvent molecules. These features make the free energy surface of the molecule complex enough to be used as a benchmark for conformational analysis techniques, yet simple enough to be easily computed.

The free energy surface depends not only on the state of the molecule itself, but also on the external conditions. Factors that can alter the calculations are:

- whether the molecule is in vacuum or in a solution,
- simulation parameters (e.g. simulation temperature),
- simulation force field.

Figure 6 shows a free energy plot I obtained for the AD in vacuum using the MMFF94 [24] force field. The result is qualitatively consistent with prevuious work [23], [25], [26], as it features three deep meta-stable states, surrounded by high energy barriers.

13. https://github.com/P2-718na/pathsampling-project

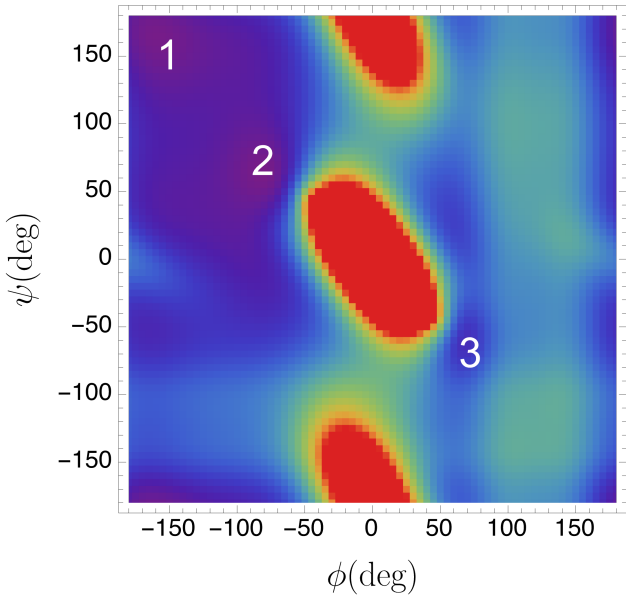

Figure 6: Free energy surface of AD in vacuum using the MMFF94 force fields. The colors of the plot go from red (high energy configurations), to purple (low energy configurations). There are three deep metastable states (indicated with numbers $1-3$), which can be recognized by their blue/purple color.

When the AD is in solution, the free energy surface changes. In this chapter, I will study AD in solution (TIP3P water molecules [27] in an AMBER96 force field) and thus we expect the free energy surface to be like the one in Figure 7. Here we have six deep meta-stable states, defined with regions in the $\phi \times \psi$ space according to Table 1.

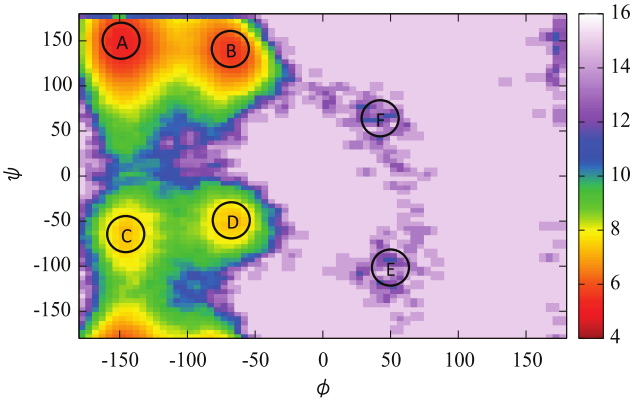

Figure 7: Reprinted from [28]. Free energy surface of AD at $300K$ using AMBER96 force field and TIP3P water molecules. The color gradient is the opposite of Figure 6. Here we have six different stable states, denoted with letters $A-F$.

4.2 Running the experiment

There are two main Python libraries that allow us to perform a path sampling simulation: OpenPathSampling (OPS) [29], [30] and PyRETIS [31], [32], [33]. In the following work, I chose to use OPS, as it is more straightforward to set-up[14]. OPS works in conjunction with the MD simulation library OpenMM [34].

State	Center ϕ	Center ψ
A	-150	150
B	-70	135
C	-150	-65
D	-70	-50
E	50	-100
F	40	65

Table 1: Center location of the stable states for AD, expressed in terms of ϕ and ψ angles (in degrees). The stable states are defined as the circles of radius 10 (degrees) around the centers.

Obtaining a PDB file

Before running the simulation, we need a `PDB` file that contains information on the position of the AD molecule solvated in a water solution. This can be generated quite easily within OpenMM[15]. We start from a `PDB` file of the alanine dipeptide, then we add water molecules and we run a small simulation to equilibrate the system. In my case, I ran the simulation with the parameters from Table 2.

Setting up the simulation

We now need to set up simulation parameters for both OpenMM (which will take care of integrating the equations of motion) and for OPS (which will take care of generating and accepting/rejecting new trajectories).

In my work, I set up two different OpenMM `VVVRIntegrators` [35]. One, with a higher temperature, is used to get an initial path that passes through all the states; the other, with a lower temperature, is used during the actual simulation. The parameters I used are summarized in Table 3 and Table 4. It is also worthy of note that OPS is built with GPU support. This means that, in principle, it can use the compute capabilities of a video card to speed up its computations. By enabling GPU support, I was able to speed up the simulation by up to two orders of magnitude[16].

With regards to OPS, we need to specify which moves to use (the *move scheme*), the order parameters, and the stable state definitions (this last thing I already did in Table 1). For the move scheme, I used OPS's `DefaultScheme`, which includes the moves I described in subsection 3.3 alongside with *replica exchange* and *minus* moves. Finally, for the order parameters I used

14. However, OPS does appear to be less well maintained than PyRETIS, as I ran into some incompatibility issues with Python and module versions.

15. http://docs.openmm.org/latest/userguide/application/03_model_building_editing.html

16. Running the simulation on my laptop (Intel Core i5-7360U), I was able to generate samples at 5 seconds per sample. Using a GPU (Nvidia RTX 2070), I was able to reach about .05 seconds per sample. However, the speed usually slowed down and stabilized at about 1 second per sample. I believe there could be some bottleneck in the implementation of OPS that prevents a full usage of the GPU.

the ϕ and ψ angles (described in subsection 4.1). The λ interfaces are chosen as a series of concentric circles around the stable states, as defined in Table 5 and Figure 8. This definition of the interfaces is taken from [28].

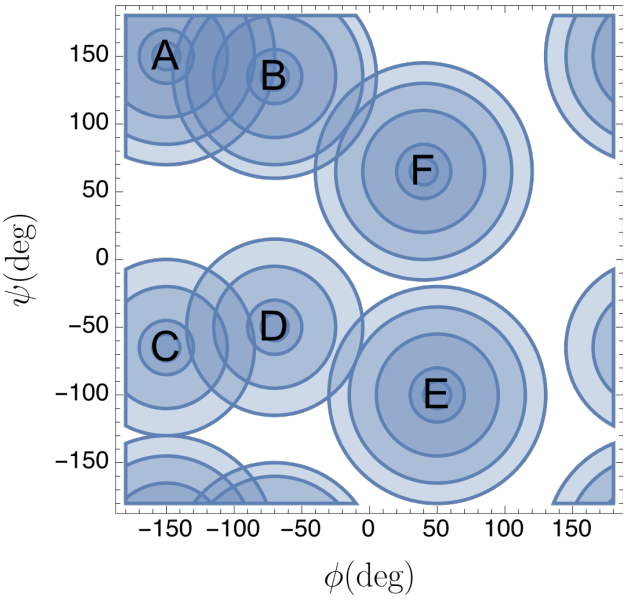

Figure 8: Lambda interfaces of alanine dipeptide plotted in periodic $\phi - \psi$ space. Each concentric circle corresponds to an interface.

Obtaining an initial trajectory and equilibrating

This step is mostly technical, as we have already set up everything that is needed to continue. What we need to do now is to use the OpenMM integrator at high temperature to find an initial trajectory that reaches all states. Then, by splitting this trajectory, we create an initial sample for each ensemble[17]. Finally, we *equilibrate* each initial sample; that is, we use it as a starting point to progressively generate new paths, saving only the last accepted path to memory. In the end, we are left with a sample set that (hopefully) is representative of the different path ensembles.

All of this process is explained well in OPS's documentation[18].

Sampling new paths

Again, this step is mostly technical. Once we have a set of equilibrated trajectories and we have chosen all the parameters for the simulation, OPS takes care of path sampling for us. The algorithm used by OPS is the one described in subsection 3.2. Each accepted path is saved to a file, so that it can be used later for calculations.

17. Remember that, by using different λ interfaces, we are effectively working with different path ensembles.

18. http://openpathsampling.org/latest/topics/simulation_setup.html.

Obtaining reaction rates

Finally, after having sampled a sufficiently large number of paths, we can use MSTIS to obtain the reaction rates. OPS does this automatically for us by automatically computing the value of equation (29).

4.3 Results

I ran the experiment with the parameters of the previous section. Figure 9 is a model of the solvated AD molecule in water. Figure 10 is the initial trajectory I generated using the high temperature integrator. After running the simulation, I got the results reported on Table 6. The reaction rate constants are mostly compatible with those of previous works [28], within the margin of error provided by the other authors. In their case, instead of running the TPS simulation in a single, large batch, they split it up in a series of smaller batches so that they could obtain an estimate for the uncertainty of the reaction rates. I was not able to do this due to time constraints.

Additional information about the simulations can be obtained with OPS. I will not get into detail here, but everything is well explained in the official documentation[19]. One thing which can be interesting to look at is how new paths are generated from old ones (Figure 11). Other insightful numerical data about the results of the experiment is available in Table 7.

19. http://openpathsampling.org/latest/

Parameter	Value
Integrator	`VVVRIntegrator`
Temperature	300K
Force field	AMBER96 with TIP3P water
Water molecule count	620
Time step	2fs
Iteration count	1000

Table 2: Parameters used for solvation of the AD molecule in water.

Parameter	Value
Integrator	`VVVRIntegrator`
Temperature	1000K
Force field	AMBER96 with TIP3P water
Time step	2fs

Table 3: Parameters for the high-temperature OpenMM engine.

Parameter	Value
Integrator	`VVVRIntegrator`
Temperature	300K
Force field	AMBER96 with TIP3P water
Time step	2fs

Table 4: Parameters for the low-temperature OpenMM engine. These parameters are the same as of those in Table 2.

State	λ_0	λ_1	λ_2	λ_3	λ_4
A	10	20	45	65	80
B	10	20	45	65	75
C	10	20	45	60	/
D	10	20	45	60	/
E	10	20	45	65	80
F	10	20	45	65	80

Table 5: λ interfaces around the stable states. Each interface is defined as a circle centered in one of the stable states, with increasing ragius. In the table the different radii are reported(in degrees).

/	A	B	C	D	E	F
A	/	77.59	3.16	3.26	<.01	<.01
B	198.87	/	1.57	2.25	<.01	<.01
C	79.01	17.69	/	183.74	<.01	<.01
D	17.62	3.75	124.22	/	<.01	<.01
E	1.11	1.88	.31	30.95	/	<.01
F	6.53	168.37	<.01	.29	38.89	/

Table 6: Rate matrix k_{ij} of the Alanine Dipeptide molecule, expressed in ns^{-1}. Most results agree with previous work, and all results are correct in terms of order of magnitude. OPS reported null reaction rates for some transitions; this is probably due to the fact that the E and F states are rarely visited, and my simulation was too short to obtain sufficient data. Previous works report them to be in the order of $10^{-2} - 10^{-3} ns^{-1}$. The uncertainty can be estimated by taking the square root of the values. The self reaction rates are undefined and left blank.

Result	Value
Initial trajectory generation computing time	$1min$
Equilibration steps	500
Equilibration computing time	$10min$
Simulation steps	10^5
Simulation computing time	$9h5min$
Path acceptance ratio	61.7%
Rate matrix computing time	$20min$
Amount of data generated	100GB

Table 7: Additional numerical information regarding the experiment.

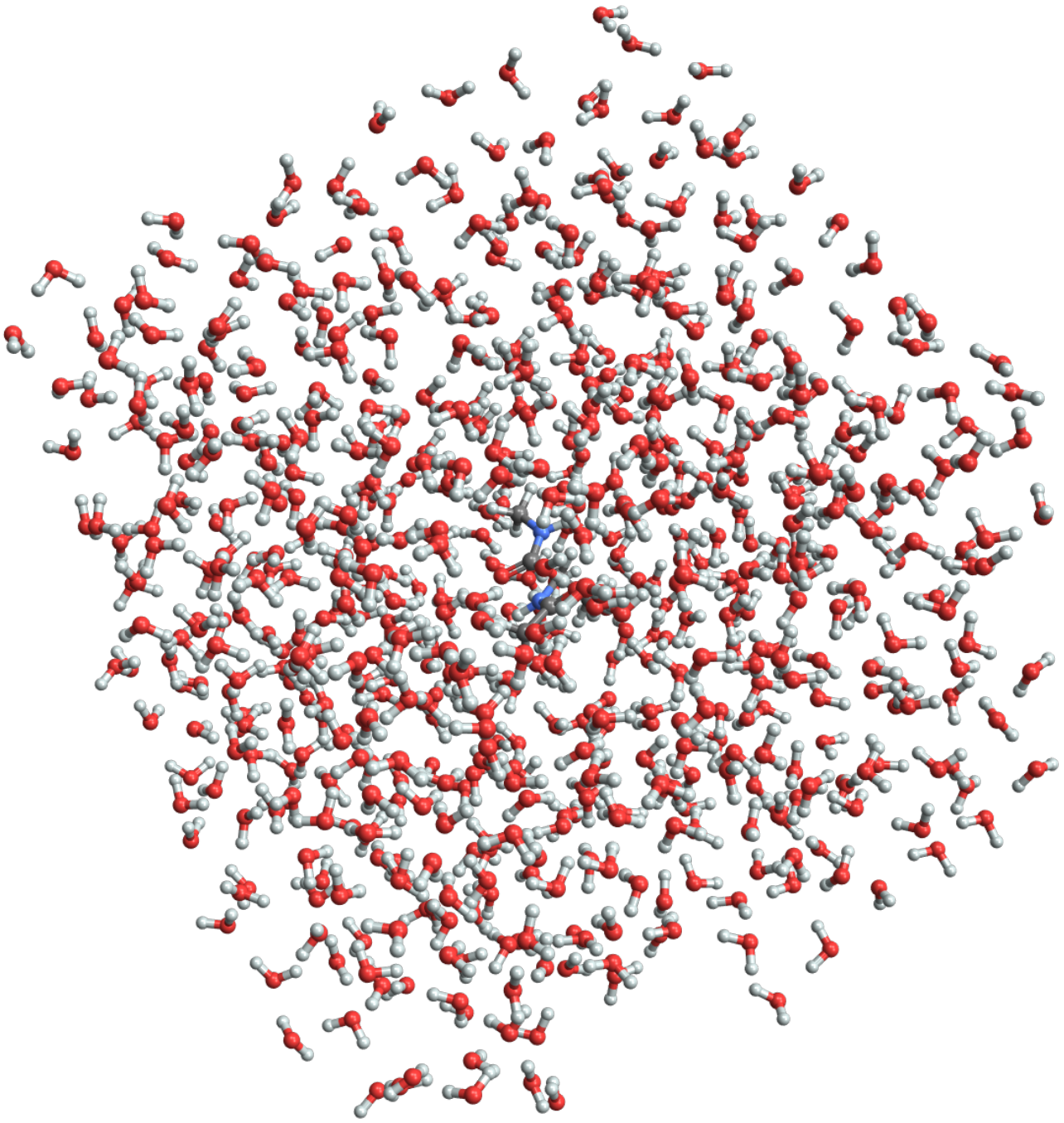

Figure 9: 3D view of a solvated AD molecule in water, obtained with the equilibration procedure described in subsection 4.2. The molecule is at the center of the cubic volume.

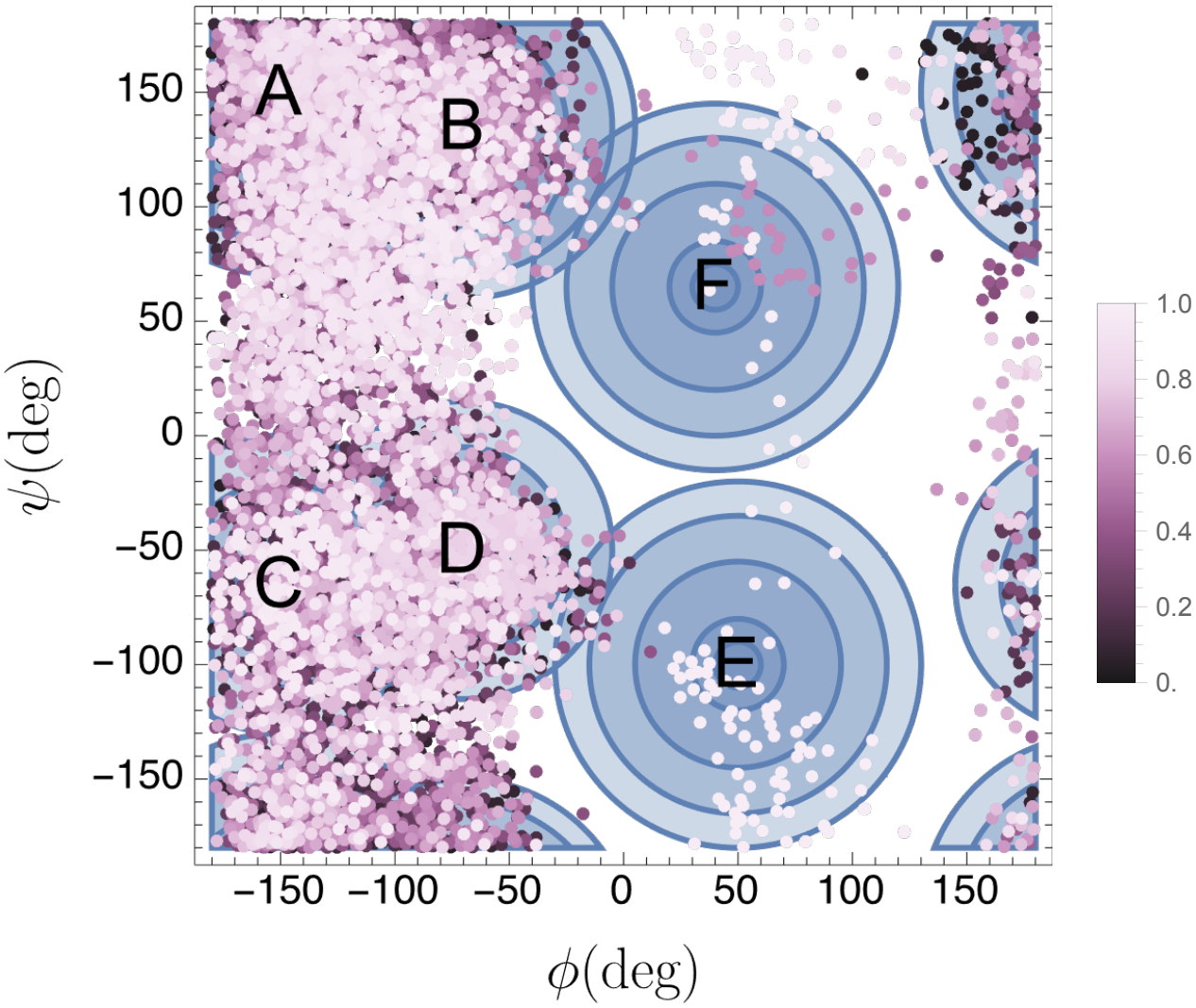

Figure 10: Initial trajectory in $\phi - \psi$ space generated using the high temperature engine. Each coloured point represents a snapshot of the trajectory. The colour indicates the order in which the snapshots were generated. Using the colour, we can see that the state F was the last one visited, with a trajectory coming from E. It is also easy to see that the states E and F are visited rarely compared to A, B, C and D.

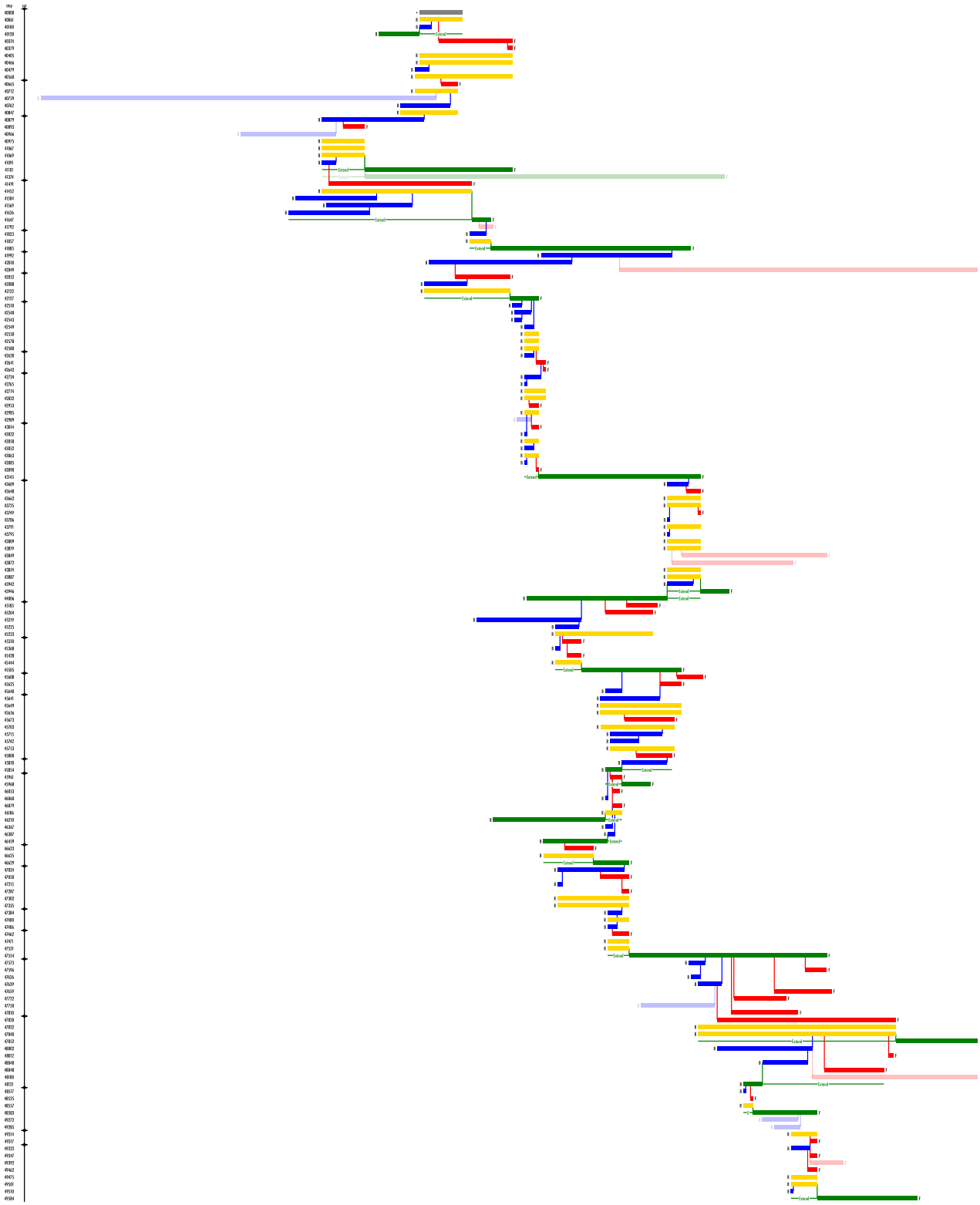

Figure 11: Move tree from the path sampling simulation (cropped). Each row indicates a new move performed in the simulation. Semi-transparent rows indicate rejected moves; the other lines indicate accepted moves. The color indicates the type of move that was performed: Blue: backwards shooting move, red: forward shooting move, green: minus move, yellow: path reversal, gray: starting trajectory. Vertical lines indicate the shooting point for shooting and minus moves. On the left side, the first column indicates the move number and the second column indicates when a new trajectory is uncorrelated from past trajectories. The right side of the image has been cropped to fit into the page and some path endpoints are not visible.

5 Applicability and limitations of the current methods

The methods discussed up until now have numerous applications and limitations. In conclusion to this text, I will now try to give a brief overview on the matter. A more complete collection of topics is found in [36], [37] and [38]. The information in this section is taken from those articles.

5.1 Applications

The Reweighted Path Ensemble

From TIS, we can extract even more information. If we reweigh each path by its computed crossing probability, we can obtain an approximation of the total, unbiased path ensamble (which we will call *Reweighted Path Ensemble, RPE)*. This ensemble contains all the information of the reaction process under study. From this, one can obtain free energy landscapes of the process under study and information on the reaction coordinates.

Reaction coordinate analysis

Probably the most general application of TPS is the possibility to use the path ensemble (more precisely, the RPE) to obtain good reaction coordinates. The ideal reaction coordinate is the *committor*, which is the probability of a given configuration to reach a target state.

Chemical reactions

The study of chemical reactions aims to find the reaction mechanism and/or the free energy barrier. These reactions happen on the quantum scale and additional computational power is needed. A subset of chemical reactions is the application to enzymes and molecular conformational changes in proteins [39] and molecules of life.

Physical systems

TPS also enables the study of some physical systems. Mainly, it allows to study phase transitions. One example is the study of phase transitions in glassy systems. It can also be applied to study small quantum systems.

5.2 Limitations

Finding an initial trajectory

As we have already discussed (subsection 3.3), finding an initial trajectory can be a problem. Usually, unphysical biases have to be used. This is not a problem per se, but if the biased trajectories are too different from the real ones, the equilibration process might not converge.

Defining stable states and order parameters

A correct definition of the stable states is mandatory to get a correct result. An incorrect definition of the stable states might misclassify paths in the ensemble and lead to wrong results. The same applies for the choice of a correct order parameter. This is a lesser problem with respect to the definition of a reaction coordinate (which is needed in methods alternative to TPS).

Generating Uncorrelated trajectories

In order to obtain a good path ensemble, we must make sure that the new trajectories decorrelate over time. Generating new, uncorrelated trajectories can be difficult. There is a quantum-computing approach that aims to resolve this problem [12] by using the intrinsic random nature of a quantum annealer to generate uncorrelated paths.

References

[1] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[2] 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, 06 1953.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

[4] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications*, volume 1 of *Computational Science Series*. Academic Press, San Diego, second edition, 2002.

[5] 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, 02 1998.

[6] Peter G. Bolhuis, Christoph Dellago, and David Chandler. Sampling ensembles of deterministic transition pathways. *Faraday Discuss.*, 110:421–436, 1998.

[7] 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, 06 1998.

[8] Christoph Dellago, Peter G. Bolhuis, and David Chandler. On the calculation of reaction rate constants in the transition path ensemble. *The Journal of Chemical Physics*, 110(14):6617–6625, 04 1999.

[9] Eliodoro Chiavazzo, Roberto Covino, Ronald R. Coifman, C. William Gear, Anastasia S. Georgiou, Gerhard Hummer, and Ioannis G. Kevrekidis. Intrinsic map dynamics exploration for uncharted effective free-energy landscapes. *Proceedings of the National Academy of Sciences*, 114(28), June 2017.

[10] Christoph Dellago, Peter G. Bolhuis, and Phillip L. Geissler. *Transition Path Sampling*, chapter 1, pages 1–78. John Wiley & Sons, Ltd, 2002.

[11] Michael Plainer, Hannes Stärk, Charlotte Bunne, and Stephan Günnemann. Transition path sampling with boltzmann generator-based mcmc moves, 2023.

[12] Danial Ghamari, Philipp Hauke, Roberto Covino, and Pietro Faccioli. Sampling rare conformational transitions with a quantum computer. *Scientific Reports*, 12(1), Sep 2022.

[13] Danial Ghamari, Roberto Covino, and Pietro Faccioli. Sampling a rare protein transition with a hybrid classical-quantum computing algorithm, 2023.

[14] C. Dellago, P.G. Bolhuis, and P.L. Geissler. Transition path sampling methods. *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*, page 349–391, 2006.

[15] Christopher N. Rowley and Tom K. Woo. Generation of initial trajectories for transition path sampling of chemical reactions with ab initio molecular dynamics. *The Journal of Chemical Physics*, 126(2), Jan 2007.

[16] Johannes Kästner. Umbrella sampling. *WIREs Computational Molecular Science*, 1(6):932–942, 2011.

[17] Michael O'Connor, Helen M. Deeks, Edward Dawn, Oussama Metatla, Anne Roudaut, Matthew Sutton, Lisa May Thomas, Becca Rose Glowacki, Rebecca Sage, Philip Tew, Mark Wonnacott, Phil Bates, Adrian J. Mulholland, and David R. Glowacki. Sampling molecular conformations and dynamics in a multiuser virtual reality framework. *Science Advances*, 4(6):eaat2731, 2018.

[18] Titus S. van Erp, Daniele Moroni, and Peter G. Bolhuis. A novel path sampling method for the calculation of rate constants. *The Journal of Chemical Physics*, 118(17):7762–7774, May 2003.

[19] Titus S. van Erp and Peter G. Bolhuis. Elaborating transition interface sampling methods. *Journal of Computational Physics*, 205(1):157–181, May 2005.

[20] David Chandler. Statistical mechanics of isomerization dynamics in liquids and the transition state approximation. *The Journal of Chemical Physics*, 68(6):2959–2970, 03 1978.

[21] D. Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.

[22] Jutta Rogal and Peter Bolhuis. Multiple state transition path sampling. *The Journal of chemical physics*, 129:224107, 01 2009.

[23] Charles. Brooks and David A. Case. Simulations of peptide conformational dynamics and thermodynamics. *Chemical Reviews*, 93(7):2487–2502, 1993.

[24] Thomas A. Halgren. Merck molecular force field. i - v. *Journal of Computational Chemistry*, 17(5-6):490–641, 1996.

[25] Paul E. Smith. The alanine dipeptide free energy surface in solution. *The Journal of Chemical Physics*, 111(12):5568–5579, 09 1999.

[26] Anshuman Kumar, Pablo Arantes, Aakash Saha, Giulia Palermo, and Bryan Wong. Gpu-enhanced dftb metadynamics for efficiently predicting free energies of biochemical systems. *Molecules*, 28:1277, 01 2023.

[27] William L. Jorgensen, Jayaraman Chandrasekhar, Jeffry D. Madura, Roger W. Impey, and Michael L. Klein. Comparison of simple potential functions for simulating liquid water. *The Journal of Chemical Physics*, 79(2):926–935, 07 1983.

[28] Wei-Na Du, Kristen A. Marino, and Peter G. Bolhuis. Multiple state transition interface sampling of alanine dipeptide in explicit solvent. *The Journal of Chemical Physics*, 135(14):145102, 10 2011.

[29] David W. H. Swenson, Jan-Hendrik Prinz, Frank Noe, John D. Chodera, and Peter G. Bolhuis. OpenPathSampling: A Python framework for path sampling simulations. 1. Basics. *Journal of Chemical Theory and Computation*, 15(2):813–836, 2019. PMID: 30336030.

[30] David W. H. Swenson, Jan-Hendrik Prinz, Frank Noe, John D. Chodera, and Peter G. Bolhuis. OpenPathSampling: A Python framework for path sampling simulations. 2. Building and customizing path ensembles and sample schemes. *Journal of Chemical Theory and Computation*, 15(2):837–856, 2019.

[31] Anders Lervik, Enrico Riccardi, and Titus S. van Erp. Pyretis: A well-done, medium-sized python library for rare events. *Journal of Computational Chemistry*, 38(28):2439–2451, 2017.

[32] Enrico Riccardi, Anders Lervik, Sander Roet, Ola Aarøen, and Titus S. van Erp. Pyretis 2: An improbability drive for rare events. *Journal of Computational Chemistry*, 41(4):370–377, 2020.

[33] Wouter Vervust, Daniel T. Zhang, An Ghysels, Sander Roet, Titus S. van Erp, and Enrico Riccardi. Pyretis 3: Conquering rare and slow events without boundaries. *Journal of Computational Chemistry*, n/a(n/a).

[34] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):1–17, 07 2017.

[35] David A. Sivak, John D. Chodera, and Gavin E. Crooks. Time step rescaling recovers continuous-time dynamical properties for discrete-time langevin integration of nonequilibrium systems. *The Journal of Physical Chemistry B*, 118(24):6466–6474, Mar 2014.

[36] Peter Bolhuis, David Chandler, Christoph Dellago, and Phillip Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual review of physical chemistry*, 53:291–318, 02 2002.

[37] Peter G. Bolhuis and David W. H. Swenson. Transition path sampling as markov chain monte carlo of trajectories: Recent algorithms, software, applications, and future outlook. *Advanced Theory and Simulations*, 4(4):2000237, 2021.

[38] P. G. Bolhuis and C. Dellago. Practical and conceptual path sampling issues. *The European Physical Journal Special Topics*, 224(12):2409–2427, Jun 2015.

[39] Jarek Juraszek, Jocelyne Vreede, and Peter G. Bolhuis. Transition path sampling of protein conformational changes. *Chemical Physics*, 396:30–44, 2012. Experimental and Theoretical Studies of Protein Dynamics and Function: From Femtoseconds to Milliseconds.