Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Jan Gustav Frydenlund

A New Initialization Method for RETIS

Master's thesis in Applied Physics and Mathematics
Supervisor: Titus S. van Erp
Co-supervisor: Raffaela Cabriolu

July 2022

NTNU
Norwegian University of
Science and Technology

Jan Gustav Frydenlund

A New Initialization Method for RETIS

Master’s thesis in Applied Physics and Mathematics
Supervisor: Titus S. van Erp
Co-supervisor: Raffaela Cabriolu
July 2022

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Abstract

Monte Carlo (MC) simulations in path space are much more computationally expensive than MC simulations in configuration space since molecular dynamics (MD) trajectories must be generated for each MC move. Furthermore, when the initial state in the Markov chain is poorly chosen, more MC moves need to be ignored when averaging to obtain the desired equilibrium properties, as subsequent moves might still bear the memory of the initial states. A carefully chosen set of initial states is thus far more important in path sampling techniques than in standard Markov chain MC methods that sample configuration space.

This thesis explores a new method to generate initial paths for the RETIS scheme. Moreover, we have written a fully functioning code as an implementation of the new method and integrated it as a Git-branch in the PyRETIS library. The new initialization method is based on an energy criterion. It favors paths that have progressed far along an order parameter and have low path energy, where the path energy is defined as the average total energy of all the constituent points in phase space.

We have tested the method on a single-particle one-dimensional double-well potential system, using PyRETIS's internal Langevin engine, and on a three-dimensional sodium-chloride system consisting of one Na^+ ion, one Cl^- ion, and 908 water molecules, using GROMACS as an external MD engine. In both systems, we show that the new method produces initial paths with far lower energy than the standard 'kick' method in all the path ensembles. Moreover, in the NaCl system, the new method produces paths with lower energy than the system exhibits in equilibrium, suggesting that the energy criterion might be excessive. Nevertheless, we think the new method without the energy criterion will still be better than the 'kick' method.

Furthermore, we investigate the importance of choosing a time-symmetric integration scheme when performing the MC shooting move. Specifically, the velocity Verlet algorithm (time-symmetric) and leap-frog algorithm (time-asymmetric) are studied.

Finally, we compare TIS simulations with initial paths made by 'kick' and the new method, revealing a regression toward the mean for both the methods.

Sammendrag

Monte Carlo-simuleringer (MC) i banerommet (eng. *path space*) er langt mer beregningskrevende enn MC-simuleringer i konfigurasjonsrommet fordi molekylærdynamikkbaner må genereres hvert MC-steg. Hvis initialtilstanden i Markovkjeden er langt unna likevekt må man ignorere flere Monte Carlo-steg når man skal regne ut de ønskede egenskapene til systemet. Det er fordi påfølgende steg fortsatt kan bære minnet av initialtilstanden, som kan forplante seg når man regner ut gjennomsnittet. Derfor er det mye viktigere å bruke en omhyggelig valgt initialtilstand i banesamplingsteknikker enn i standard Markovkjede Monte Carlo-teknikker som sampler konfigurasjonsrommet.

I denne masteroppgaven utforsker vi en ny metode for å generere initialbaner til RETIS. Dessuten har vi skrevet en fullt fungerende kode som implementerer den nye metoden og integrert den som en Git-forgrening i PyRETIS-biblioteket. Den nye initialiseringsmetoden er basert på et energikriterium. Den favoriserer baner som har kommet langt langs en reaksjonskoordinat og har lav baneenergi, der baneenergien er definert som den gjennomsnittlige totalenergien til alle faseromspunktene som banen består av.

Vi har testet metoden på et endimensjonalt énpartikkel-system i et dobbelbrønn-potensial med PyRETIS' interne Langevin-maskin og på et tredimensjonalt natriumkloridsystem bestående av ett Na^+-ion, ett Cl^--ion og 908 vannmolekyler med GROMACS som ekstern maskin. I begge systemene viser vi at den nye metoden produserer baner med langt lavere energi enn den ordinære «kick»-metoden i alle baneensemblene. Dessuten produserer den nye metoden initialbaner til NaCl-systemet med lavere energi enn det systemet viser i likevekt, noe som kan tyde på at energikriteriet kanskje er overflødig. Uansett antar vi at den nye metoden uten energikriteriet likevel vil være bedre enn «kick»-metoden.

Videre investigerer vi viktigheten av å bruke en tidssymmetrisk integrasjonsalgoritme når man utfører det såkalte «shooting»-steget. Her studeres spesifikt «velocity Verlet» (tidssymmetrisk) og «leap-frog» (tids-asymmetrisk).

Til slutt sammenligner vi TIS-simuleringer med initialbaner lagd av «kick» og av den nye metoden, noe som viser en regresjon mot middelverdien for begge metodene.

Contents

Preface

Rare events are events that occur very infrequently and have a significant impact. Earthquakes, pandemics, stock market crashes, and acts of terrorism are all examples of rare events from everyday life. As the statistician, trader, and author Nassim N. Taleb writes in his book *The Black Swan: The Impact of the Highly Improbable*, “The inability to predict outliers implies the inability to predict the course of history.” In this thesis, I address computer simulation techniques for rare events on the molecular scale and how they are crucial in studying, for example, chemical reactions.

My master’s thesis builds on theoretical study and coding that I did in a specialization project, which is part of the Master of Science degree in Applied Physics and Mathematics. Due to overlap in the theoretical background and the scientific goal, this thesis should be viewed as an extension of the previously delivered specialization project [1]. If you, the reader, have any questions, do not hesitate to contact me by email at jg.frydenlund@gmail.com.

Acknowledgements

I want to direct special thanks to Ph.D. Candidate Daniel T. Zhang and my supervisor Professor Titus van Erp for their availability, friendliness, and keen insight throughout the past year. Daniel has helped me with all sorts of technical stuff, such as version control, shell scripting, cluster computing, and understanding the PyRETIS library. Titus has helped me understand the theory, interpret results, and provided me with valuable feedback by carefully proofreading this thesis. I also want to thank Associate Professor Raffaela Cabriolu for helping me quickly get started on the coding. You are indeed the best advisors I could have asked for. Finally, I want to thank my family and friends for moral support and encouragement throughout all these years—you are the best. This thesis would never have reached its current state without your collective help and support. For that, I am truly grateful.

Jan G. Frydenlund
Trondheim, Norway
July, 2022

Nomenclature

List of Symbols and Physical Constants

This list is ordered according to the approximate appearance in the text.

$\mathbf{r}_i = (x_i, y_i, z_i)$	position of particle i (3-dimensional vector)
$\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz})$	velocity of particle i (3-dimensional vector)
$\mathbf{f}_i = (f_{ix}, f_{iy}, f_{iz})$	force on particle i (3-dimensional vector)
m_i	mass of particle i
t	time
$\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t}$	time derivative of x
δt	discrete time step
Ψ	molecular wave function
$\langle ... \rangle$	ensemble average
N	number of particles (i.e. point-particles) in the system
$\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N)$	configuration point ($3N$-dimensional vector)
$\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N)$	velocity vector ($3N$-dimensional vector)
$\mathbf{x} = (\mathbf{R}, \mathbf{V})$	phase point or point in phase space ($6N$-dimensional vector)
$U(\mathbf{R})$	potential energy of configuration point $\mathbf{R}$
$K(\mathbf{V})$	kinetic energy corresponding to velocities $\mathbf{V}$
$E(\mathbf{R}, \mathbf{V})$	total energy $U(\mathbf{R}) + K(\mathbf{V})$
k_B	Boltzmann's constant ($1.380\,649 \times 10^{-23}\,\mathrm{J\,K^{-1}}$)
T	temperature
β	reciprocal temperature ($1/k_\mathrm{B}T$)
n	new configuration
o	old configuration
$\mathcal{P}_\mathrm{acc}(\mathrm{o} \to \mathrm{n})$	acceptance probability of a move from o to n
$\tau_\mathrm{eff}(a)$	CPU efficiency time: minimum computational cost needed to determine a with a relative error equal to 1
τ_eff^{-1}	efficiency
n	total number of simulations used in a simulation series
A, B	stable states: reactant and product, respectively
$\lambda_A = \lambda_0$	interface defining stable state A
$\lambda_B = \lambda_n$	interface defining stable state B
$\mathcal{A}$, $\mathcal{B}$	overall states
f_A	flux through λ_A
k_{AB}	reaction rate
$\lambda(\mathbf{x})$	reaction coordinate at a phase point $\mathbf{x}$

λ_i	value defining interface i
$\mathcal{P}_A(\lambda\|\lambda')$	conditional crossing probability to go from interface λ' to λ
$[i^+]$	path ensemble
$[i^+] \leftrightarrow [(i+1)^+]$	swapping move between two adjacent path ensembles
$r_{\mathrm{ion}} = \|\mathbf{r}_{\mathrm{Na}^+} - \mathbf{r}_{\mathrm{Cl}^-}\|$	Euclidean distance between Na^+ and Cl^-
N_w	number of water molecules
V	volume
F	Helmholtz free energy
S	entropy
ρ	probability density
α	probability to select a point from a ranking list's upper half
n_{del}	number of times a criterion must be fulfilled to delete a path
n_{reac}	number of AB-paths that must be found to terminate the algorithm
AA, BB, AB	categories defining A-to-A, B-to-B, and A-to-B trajectories
p_{AA}, p_{BB}, p_{AB}	probabilities to select categories AA, BB, and AB
p_j	probability to choose point $\mathbf{x}_j$
$\lambda_{\mathrm{min},k}$	minimum order parameter of path k
$\lambda_{\mathrm{max},k}$	maximum order parameter of path k
λ^*_{min}	minimum order parameter in a group of paths
λ^*_{max}	maximum order parameter in a group of paths
E_k	path energy of path k
N_k	number of phase points in path k
C_V	heat capacity at constant volume
N_{A}	Avogadro's number ($6.022\,140\,76 \times 10^{23}\,\mathrm{mol}^{-1}$)
$R = N_{\mathrm{A}} k_{\mathrm{B}}$	molar gas constant

List of Abbreviations and Acronyms

AIMD	*ab initio* molecular dynamics
CPU	central processing unit
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MD	molecular dynamics
OP	order parameter (used interchangeably with RC)
RC	reaction coordinate (used interchangeably with OP)
RETIS	replica exchange transition interface sampling
TIS	transition interface sampling
TPS	transition path sampling
TST	transition state theory

1 Introduction

Traditionally, physics divides into two categories: theoretical physics and experimental physics. Theoretical physics uses mathematical models to explain and predict natural phenomena, whereas experimental physics applies measurements and observations to address the same phenomena. Computational physics is the application of numerical analysis and computer science to solve problems in physics where a theory exists, but the phenomena are too complex to find an exact (or even approximate) analytical solution. Computational physics uses nothing but the laws of (theoretical) physics as a framework, yet it is still sometimes regarded as the third category in physics. The problems solved by computational physics are often referred to as 'computer experiments' or 'simulations.'

Molecular dynamics (MD) is a simulation method for mimicking the physical movements of atoms and molecules using both quantum and classical mechanics. The movement from one configuration of atoms to another is called a path or trajectory. Today's fastest supercomputers can simulate a million atoms at 100 microseconds per day [2]. MD has many applications, such as drug design, protein structure predictions, and understanding allostery [3].

Events that are rare on the molecular time scales dominate the kinetics of many vital processes such as phase transitions, self-assembly, conformational changes, chemical and biological reactions, and protein folding [4]. The limitations on the number of atoms and simulation time make simulating such events impossible using straightforward MD.

Path sampling is a method to attain precisely the same results with respect to rate constants and activation energies as one would get from a (hypothetical) extremely long MD simulation, but orders of magnitude faster [5–7]. Path sampling is essentially a Monte Carlo (MC) procedure where short MD paths are generated. New states are generated by a random modification of the previous state: this is called Markov chain Monte Carlo (MCMC). However, in standard MC, the states are configuration points of a molecular system, whereas, in path sampling, the states are MD trajectories.

Replica exchange transition interface sampling (RETIS) was developed by Van Erp in 2007 and is a highly efficient path sampling method that utilizes a series of path sampling simulations for sampling a set of path ensembles [8, 9]. An order parameter, which is a function of the phase space, measures the progress a path has made from one stable state to another. The path ensembles differ by the minimum order parameter the paths need to reach.

In MCMC simulations, an initial state needs to be established since we always modify the previous state. In RETIS, this implies that a valid path for each path ensemble needs to be generated. In principle, the initial paths do not need to be very representative of the most probable paths—and can even be unphysical—since the MC procedure should ensure that the sampling evolves automatically to the relevant state space. When averaging over the ensembles, one usually ignores a small part at the beginning of the simulation. The ignored states are those that still bear the memory of the initial paths, that is, the paths in the Markov chain that have not reached equilibrium.

However, practice has shown that when the initial paths are very different from the most probable paths, many MC steps are required to bring the sampled paths to equilibrium [10]. Since an MC step in path sampling is much more computationally expensive than a standard MC step in configuration space, a method to generate good initial paths is essential. A simple and commonly used scheme in RETIS is the 'kick' method, which works by forcing a configuration point toward an increasing order parameter. Unfortunately, the 'kick' method is often known to produce initial paths far from equilibrium.

PyRETIS is a Python library that can run RETIS simulations [11, 12]. The latest public version of PyRETIS at the time of writing is 2.5.0, and the latest current version can be found at `www.pyretis.org`.

My scientific goal for this thesis is to develop a method that produces better initial paths to the RETIS method, such that fewer initial Monte Carlo steps must be ignored in the Markov chain. The purpose is ultimately to increase the performance and computational efficiency of rare event molecular simulations. Moreover, I aim to implement the method into the PyRETIS library [13].

For the specialization project, I began implementing the algorithm of the new initialization method and tested it on simple systems after a theoretical study. For this thesis, I have made some changes to the user input parameters, which are now more intuitive to select. Furthermore, I have improved and generalized the code such that it now supports three-dimensional systems and external MD engines. Lastly, I have tested the code on a realistic full-atom system.

Following is a basic explanation of the underlying theory behind RETIS (Chapter 2), a description of the new initialization scheme (Chapter 3), results and discussion (Chapter 4), and finally, a conclusion and further work that is expected to be part of any further development of the new method (Chapter 5).

2 | Theoretical Background

2.1 Rare Events

There is no formal definition of what classifies an event to be *rare*—it depends on the field and the perspective. Generally speaking, rare events occur infrequently and impact systems significantly, such that they might destabilize.

In our case, a rare event can be defined as an important event that takes an extremely long time to occur in a straightforward MD simulation. Observing such an event in a computer experiment can literally take centuries or more.

2.2 Molecular Dynamics

Molecular dynamics (MD) simulations try to mimic the true physical movements of atoms and molecules. MD is a simulation technique for computing the dynamical and equilibrium properties of a classical many-body system consisting of atoms and molecules. The word *classical* means that the particles obey the laws of classical mechanics. The most precise physical description of atoms and molecules we have is, of course, quantum mechanics. Therefore, the 'ideal' way to simulate chemical reactions and other processes would be to use a fully quantum mechanical approach. However, such calculations are very time-consuming for large systems, and classical mechanics is an excellent approximation for a wide range of materials [14]. The trajectories of the particles are obtained by numerically solving Newton's second law,

$$\mathbf{f}_i = m_i \frac{\mathrm{d}^2 \mathbf{r}_i}{\mathrm{d}t^2}, \tag{1}$$

for N particles with constant mass, where $\mathbf{f}_i$ is the force on particle i, and m_i and $\mathbf{r}_i$ are the mass and position of particle i, respectively. The force on each particle depends on its position relative to the other particles. Consequently, the particles' motions are difficult and often impossible to describe analytically. A well-known example is the absence of an analytical solution to the general three-body problem [15].

When molecular dynamics simulations emerged in the late 50s, hard-sphere models were used. In such models, the spheres move at constant velocity in straight lines, exerting perfectly elastic collisions [16]. As computer power increased and clever algorithms were introduced, more realistic models could be used, where the forces update whenever the particles change their position or whenever any of the particles with which it interacts changes position. *Finite difference methods* are used to generate trajectories with continuous potential models. The trajectories are integrated in time with a fixed time step δt. The net force on each particle at time t is calculated by invoking the superposition principle, i.e., by summing the interactions with other particles. The forces are assumed to be constant during one time step, and hence also the acceleration, by Equation (1). The positions and velocities at time $t + \delta t$ are calculated using the acceleration and the positions and velocities at time t. Then the new forces are calculated, resulting in yet new sets of positions and velocities at time $t + 2\delta t$, and so on. A commonly used integrator in MD is the *velocity Verlet* algorithm, although several others exist, such as *Verlet*, *leap-frog*, and *Runge-Kutta*.

The molecular motion can be simulated either by classical MD or by *ab initio* MD (AIMD). Both methods capture the dynamics in terms of classical dynamics and statistical mechanics. In standard MD, the forces are determined by taking the gradient of a classical force field, which is a function of the atomic coordinates, whereas AIMD unifies approximate electronic structure theory and classical MD [17].

Ab initio strictly means "from the beginning" or "from first principles," which implies that such calculations should only require the input of physical constants and the fundamental laws of physics. In practice, however, *ab initio* MD means obtaining the forces by considering a full electronic structure calculation based on the Schrödinger equation using two approximations. First, the Born–Oppenheimer approximation is applied, where one assumes that the electronic and nuclear motions can be separated since the electrons have a much smaller mass than the nuclei [16]. Under the Born–Oppenheimer approximation, the total wave function for a molecule can be written as

$$\Psi_{\text{tot}}(\text{nuclei, electrons}) = \Psi(\text{electrons})\Psi(\text{nuclei}). \tag{2}$$

Second, the nuclei are considered classical particles; only the electrons are treated quantum mechanically.

The force evaluations are the most computationally heavy operations in molecular dynamics, and the two approaches for calculating them differ significantly in terms of computational cost. *Ab initio* methods study system sizes and time scales that are far smaller than systems studied by standard MD because they are the most costly.

The major advantage of molecular dynamics over Monte Carlo methods is the ability to calculate time-dependent properties [16]. This is possible because MD generates system configurations that are connected in time. However, from a rare event perspective, there are two problems with standard MD approaches. First, the time scale of atomistic MD simulations for realistic dimensions ($> 10^5$ particles) is typically limited to a few milliseconds—far lower than the relevant time range of many processes dominated by rare events.

2.3 Monte Carlo

The underlying concept of Monte Carlo (MC) simulations, or Monte Carlo methods, is to determine some properties by using random numbers. This is done by generating configurations of a system by making random changes to, for example, the position or orientation of a molecule. MC simulations can therefore only determine thermodynamic properties, i.e., time-independent properties.

Each new configuration generated by the MC method is accepted according to a special set of criteria. In the end, the desired property A is calculated by simply averaging over all the M accepted configurations,

$$\langle A \rangle = \frac{1}{M} \sum_{i=1}^{M} A(\mathbf{R}_i). \tag{3}$$

The average in MD and MC is not taken from the first sampled configuration because it is generally far from representative of the desired properties. Instead, ignoring the first (say 5–10 %) of the MD or MC steps is good practice. This is because the subsequent steps might still resemble the initial configuration, which could be far from equilibrium (Section 2.6.5).

There are many applications of Monte Carlo methods (and, to a lesser extent, of molecular dynamics) beyond computing equilibrium properties of classical many-body systems. However, the notions of MC and MD will only refer to such uses in this report.

2.3.1 Metropolis Algorithm

The Metropolis algorithm is the most popular way to generate an ensemble of configurations in Monte Carlo simulations of molecular systems [16]. The Metropolis algorithm is a *Markov chain* Monte Carlo (MCMC) method. MCMC is also referred to as *importance sampling* and is a method for sampling from a probability distribution where it is difficult to draw samples. In rare events, for example, the values of interest are very improbable to draw. In a Markov chain, each trial depends only upon the preceding trial and not any previous trials.

The Metropolis algorithm is as follows. Make a random change to the system, for instance by moving an atom. Calculate the new energy of the configuration using the potential energy function. If the energy of the new configuration is lower than the energy of the old configuration, the new configuration is accepted. If the energy of the new configuration is higher, then the *Boltzmann factor* of the energy difference, $\exp\left[-\beta(U_\mathrm{n}(\mathbf{R}) - U_\mathrm{o}(\mathbf{R}))\right]$, is calculated, where 'n' and 'o' denote 'new' and 'old,' respectively. A uniformly random number between 0 and 1 is drawn. If the Boltzmann factor is higher than the random number, the new configuration is accepted and becomes the next state; otherwise it is rejected and the old configuration is retained for the next iteration and recounted. The acceptance probability of going from the old state to the new state, $\mathcal{P}_{\text{acc}}(\text{o} \to \text{n})$, can then be formulated as

$$\mathcal{P}_{\text{acc}}(\text{o} \to \text{n}) = \text{MIN}\left[1, \exp(\beta(U_{\text{o}} - U_{\text{n}}))\right]. \tag{4}$$

Hence, the higher the energy of the new state (i.e., the lower the value of $U_{\text{o}} - U_{\text{n}}$), the lower the probability of accepting.

One needs to have an initial configuration in order to make a first random change to a configuration. How to obtain such an initial configuration will be discussed in Section 2.6.5.

The Metropolis algorithm assumes that the generation probabilities are symmetric, i.e., that the probability of generating the new state from the old equals the probability of generating the old state from the new, $\mathcal{P}_{\text{gen}}(\text{o} \to \text{n}) = \mathcal{P}_{\text{gen}}(\text{n} \to \text{o})$. If this is not the case, the Metropolis-Hastings scheme must be used, which is a generalization of the Metropolis scheme.

2.4 Efficiency

The *CPU time* is the part of the total simulation time spent by the central processing unit (CPU), measured in seconds. The CPU time is thus dependent on the hardware and technical details of the implementation. The *CPU efficiency time*, $\tau_{\text{eff}}(a)$, can be defined as the minimum computational cost needed to determine a with a relative error equal to one [7]. In MD simulations, force calculations are the most computationally expensive operations. Therefore, τ_{eff} can be expressed as an integer representing the number of MD steps. This provides a measure of the CPU efficiency time independent of the computational resources. The lower the value of τ_{eff}, the more efficient the simulation method. The *efficiency* is sometimes defined as the inverse CPU efficiency time, τ_{eff}^{-1}, so that a greater value means a more efficient method.

2.5 Path Sampling

The conventional way to tackle the time-scale problem with rare molecular events is based on *transition state theory* (TST) and separates the problem into two steps [18]. First, the calculation of the free energy barrier as function a *reaction coordinate* (RC), and second, the calculation of the transmission coefficient by sampling trajectories departing from the top of the barrier. The problem, however, with this approach is that choosing a good reaction coordinate can be extremely difficult in certain systems. Furthermore, it usually requires detailed *a priori* knowledge of the transition mechanism. A poorly chosen RC can lead to a statistically inaccurate or immeasurable rate constant (Section 2.6.2).

2.5.1 Reaction Coordinate and Stable States

A reaction coordinate (RC) is chosen to determine what state the system is in. In this thesis, the term 'order parameter' (OP) will be used synonymous with the term 'reaction coordinate.' The reaction coordinate, $\lambda(\mathbf{x})$, is a function of a phase point $\mathbf{x} = (\mathbf{R}, \mathbf{V})$ and is chosen such that the system is in

$$\begin{cases} \text{stable state } A \text{ if } \lambda(\mathbf{x}) < \lambda_A, \text{ and} \\ \text{stable state } B \text{ if } \lambda(\mathbf{x}) > \lambda_B. \end{cases} \tag{5}$$

In practice, λ often depends only on the positions $\mathbf{R}$. In chemistry jargon, the stable states A and B are known, respectively, as the reactant and the product. The transition region is when $\lambda_A \leq \lambda(\mathbf{x}) \leq \lambda_B$. In transition interface sampling (TIS), only the stable states A and B are used to determine the reaction rate (Section 2.6.2).

In TIS and replica-exchange TIS (RETIS), the *overall states* $\mathcal{A}$ and $\mathcal{B}$ are introduced. The overall state $\mathcal{A}$ is defined such that it covers all phase points inside the stable state A and all phase points that have a path, or trajectory, coming directly from A, i.e., without having been to B. The same definition applies for overall state $\mathcal{B}$, with $\{A, B\} \to \{B, A\}$. In this way, the overall states also depend on the history of the paths—not just the position at a specific time—thus eliminating the fluctuations in and out of the stable states [8].

Ideally, an RC is a function of all the collective variables responsible for the reaction dynamics and is alone sufficient to track the progress of the reaction. Finding a proper RC is difficult in practice, especially in high-dimensional complex systems, but it should, in any case, describe the progress of the reaction to some extent [19]. In some cases, the RC can be the distance between two ions, as in the sodium chloride system described in Section 2.10.

2.5.2 Monte Carlo Moves in Path Space

The *shooting move* is a method for generating a new path from A to B or from A to A, using the old path. Using the Metropolis algorithm, each path should only depend upon the previous path. An intuitive way to generate a new path could be to slightly change the initial condition (in A) and use MD to generate a new path toward B. The problem with this approach is that even though the MD trajectories are deterministic, the system might be chaotic: that is, the final position is highly sensitive to changes in the initial conditions. Moreover, since the event is rare, the new trajectory will probably not be much closer to B.

Instead, a random perturbation is made, e.g., by changing the momenta at a randomly chosen phase point of the old path. Then, the equations of motion are integrated forward

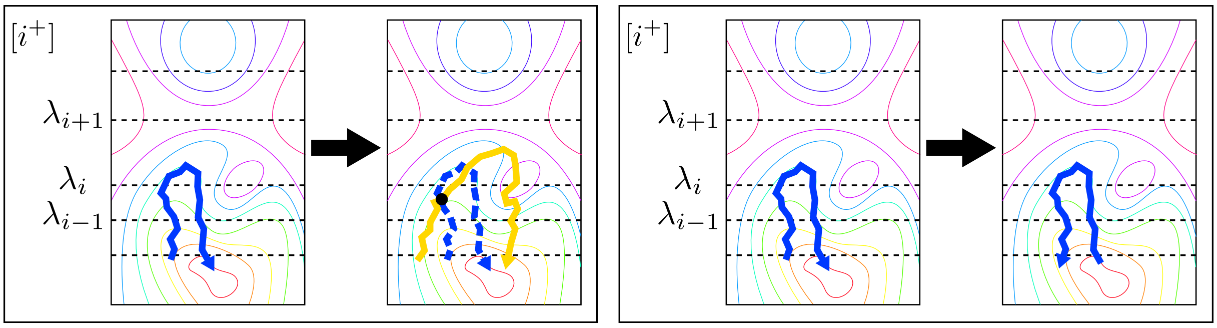


Figure 1: Illustrated examples of the shooting move and the time-reversal move. The arrows represent the paths moving through a free energy landscape in two dimensions; the colored curves are equipotentials of constant energy; the straight, dashed lines are the interfaces. The vertical axis is the order parameter λ; the horizontal axis is an arbitrary variable. *Left panel*: The shooting move on the $[i^+]$ ensemble. A random phase point (black dot) of the old path (dashed, blue arrow) is chosen. Then the equations of motion are integrated forward and backward in time, creating a new path (yellow arrow). The new path is accepted if it too crosses the λ_i interface. *Right panel*: The time-reversal move on the $[i^+]$ ensemble. The path direction is simply changed by reversing the order of the phase points and reversing the velocities.

and backward in time from this point, creating a new path. The shooting move gives a much higher chance of generating a valid path than simply starting from a random point in the phase space [11]. The left panel of Figure 1 illustrates the shooting move. The points on a path can also be referred to as *time slices*, as they are discrete snapshots of the path.

The *time-reversal move* changes the direction of the old path by changing the order of the time slices and reversing the velocities. This move is computationally cheap as it does not require any force calculations. The time-reversal move is illustrated in Figure 1, right panel.

The *swapping move* is also computationally cheap. It is described in the RETIS section (Section 2.7) as it requires the introduction of *interfaces* and *path ensembles*.

2.5.3 Transition Path Sampling

Transition path sampling (TPS) is a method where no prior knowledge of the system is needed. The main difference to TST is that where TST sees the transition of a system as the crossing of a specific point or surface in the free energy landscape, TPS sees the transition as a path (or trajectory) between the reactant and the product state. In TPS, a random walk in path space is performed via the Metropolis scheme, creating an ensemble of unbiased paths between the two stable states. To achieve this, TPS generates a Markov chain of paths, where each new trial path is created by modifying an old path, using a Monte Carlo move like the shooting move.

The reaction rate in TPS, $k_{AB}^{\mathrm{TPS}}(t)$, is determined as the time derivative of a general time correlation function $C(t)$ [20],

$$k_{AB}^{\mathrm{TPS}}(t) = \frac{\mathrm{d}C(t)}{\mathrm{d}t}, \quad C(t) = \frac{\langle h_A(\mathbf{x}_0) h_B(\mathbf{x}_t)\rangle}{\langle h_A(\mathbf{x}_0)\rangle}, \tag{6}$$

where $h_A(\mathbf{x})$ and $h_B(\mathbf{x})$ are the characteristic functions,

$$h_A(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in A \\ 0 \text{ if } \mathbf{x} \notin A, \end{cases} \qquad h_B(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in B \\ 0 \text{ if } \mathbf{x} \notin B, \end{cases} \tag{7}$$

and the stable states A and B are defined as in (5). Note that $h_A(\mathbf{x}) + h_B(\mathbf{x}) \neq 1$.

2.6 Transition Interface Sampling

Transition interface sampling (TIS) is a technique for modeling rare events and an improvement of TPS, developed by Van Erp et al. in 2003 [18]. The results of TIS-based methods are equivalent to the ones that would be obtained by running exceedingly long brute-force molecular simulations, but orders of magnitude faster [11]. The advantage of TIS and replica-exchange TIS (RETIS) over TPS is that they are based on statistical path ensembles with flexible lengths, whereas, in TPS, the paths have a fixed path length. In TIS and RETIS, many paths are shorter than the fixed TPS path length, reducing the number of MD steps per path. Furthermore, sometimes a TIS path is longer than the corresponding TPS path because there is no limiting value. TPS will therefore miss the contribution of the very long trajectories. The TIS/RETIS approach is consequently more accurate and precise than TPS.

2.6.1 Interfaces

In TIS, the phase space is divided by a set of $n-1$ non-intersecting interfaces in between λ_A and λ_B. By defining $\lambda_0 \equiv \lambda_A$ and $\lambda_n \equiv \lambda_B$, the set of interfaces is $\{\lambda_0, \lambda_1, ..., \lambda_n\}$, with $\lambda_i < \lambda_{i+1}$. Each interface is defined as where the reaction coordinate, $\lambda(\mathbf{x})$, equals λ_i, see Figure 2. The number of interfaces and their separation should be chosen to optimize the efficiency. λ_0 should be placed such that it is frequently crossed when an MD simulation is initiated from within the stable state A: this is to calculate the flux in Equation (10).

2.6.2 Reaction Rate

The reaction rate is the most central quantity calculated in TIS simulations and can be formulated as

$$k_{AB} = f_A \mathcal{P}_A(\lambda_B|\lambda_A), \tag{8}$$

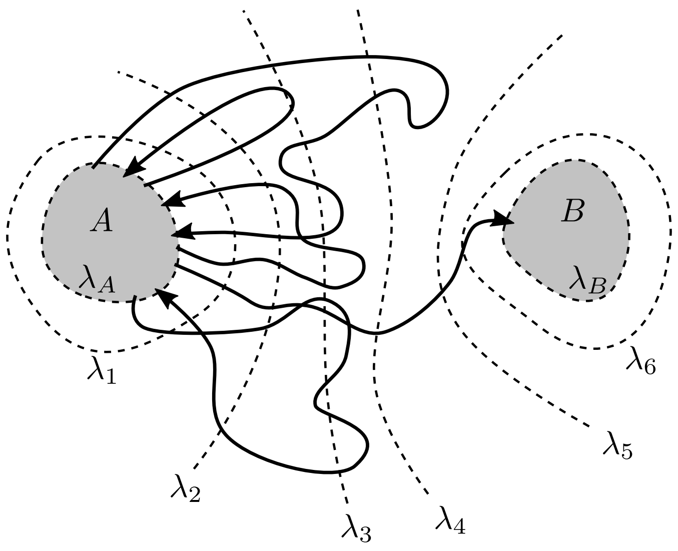


Figure 2: Example of the division of phase space by $n+1=8$ interfaces $\{\lambda_A = \lambda_0, \lambda_1, ..., \lambda_7 = \lambda_B\}$. The interfaces correspond to the calculation of Equation (9) with $n=7$. A and B are the stable state regions where $\lambda(\mathbf{x}) < \lambda_A$ and $\lambda(\mathbf{x}) > \lambda_B$, respectively. Paths belonging to ensemble $[3^+]$ are shown. Here, the corresponding calculation of the interface crossing probability would yield $\mathcal{P}_A(\lambda_4|\lambda_3) = 2/5$. The trajectories in this figure are more similar to the RETIS method (Section 2.7), because there the trajectories are continued after reaching λ_{i+1}. The figure is based on Van Erp et al. [18].

where

$$\mathcal{P}_A(\lambda_B|\lambda_A) = \mathcal{P}_A(\lambda_n|\lambda_0) = \prod_{i=0}^{n-1} \mathcal{P}_A(\lambda_{i+1}|\lambda_i), \tag{9}$$

and f_A is the flux of trajectories through the first interface λ_A. $\mathcal{P}_A(\lambda_B|\lambda_A)$ is the probability that once λ_A is crossed, λ_B will be crossed before any recrossing with λ_A. In a rare event, this *overall* probability is very small, but it can be computed by factorizing it into probabilities $\mathcal{P}_A(\lambda_{i+1}|\lambda_i)$ that each has a much higher value than the overall probability, thus reducing the computational cost. $\mathcal{P}_A(\lambda_{i+1}|\lambda_i)$ is the probability of a path crossing λ_{i+1}, given that it starts at λ_A, crosses λ_i at least once, and ends by crossing either λ_A or λ_B. Consequently, $\mathcal{P}_A(\lambda_{i+1}|\lambda_i)$ is a complicated history-dependent conditional probability. The probabilities are calculated as fractions of the sampled paths that satisfy the conditions.

In TIS, f_A is determined by a straightforward MD simulation,

$$f_A = \frac{N_c^+}{T_{\in\mathcal{A}}}, \tag{10}$$

where N_c^+ is the number of positive crossings (i.e. crossings that increase the RC) with the interface $\lambda_A = \lambda_0$ and $T_{\in\mathcal{A}}$ is the time spent in $\mathcal{A}$. If a transition to $\mathcal{B}$ happens, however unlikely, the simulation is stopped, and another MD simulation is run from A, to avoid waiting for the system to return to A. In the end, the f_A's from the different simulations are averaged.

2.6.3 Path Ensembles

For each interface λ_i $(i = 0, 1, ..., n-1)$, there is a path ensemble $[i^+]$. The $[i^+]$ path ensemble is constructed as a Markov chain by sampling paths that obey the λ_i crossing condition. If a path starts at λ_A, has at least one crossing with λ_i, and ends by either crossing λ_A or λ_B, it satisfies the λ_i crossing condition. In TIS, the sampling is done via the shooting move. An excellent flowchart diagram of the shooting move in TIS can be seen in Van Erp et al. [21].

A new path generated by the shooting move is accepted to ensemble $[i^+]$ if the backward trajectory crosses λ_A, the total trajectory has at least one crossing with λ_i, and the path length remains within the maximum path length determined at the start of the shooting move (see Van Erp [8]). If these criteria are not met, the old path is kept, recounted, and used again to repeat the shooting move for a new random time slice, as described in Section 2.5.2.

The final result follows from a set of independent simulations $\{[\mathrm{MD}], [0^+], [1^+], ..., [(n-1)^+]\}$, where the MD simulation is used to determine f_A in Equation (10).

2.6.4 Placement of the Interfaces

To illustrate the essence of importance sampling in path space, imagine a rare event with an overall crossing probability of $\mathcal{P}_A(\lambda_B|\lambda_A) = 10^{-6}$. MD would then, on average, need to sample one million paths to observe just one crossing. Typically, TIS (and RETIS) aim to sample an equal number of paths in each ensemble. We could, for instance, place $n+1 = 7$ interfaces such that each interface has a crossing probability of $\mathcal{P}_A(\lambda_{i+1}|\lambda_i) \approx 0.1$. We can get an estimate of the overall crossing probability using just ten paths in each of the six ensembles $[0^+], [1^+], ...[5^+]$, by Equation (9). This approach has effectively reduced the minimum number of required paths from 10^6 to 60. Of course, more paths are required to obtain a statistically significant result, but the difference in required CPU time between the two methods is obvious from this back-of-the-envelope calculation.

An optimization problem is finding the number of interfaces, $n+1$, and their placements, λ_i, to minimize the CPU efficiency time. From the above example, it is apparent that the minimum number of interfaces (i.e., two: $\lambda_0 = \lambda_A$ and $\lambda_1 = \lambda_B$) is inefficient because finding the overall crossing probability $\mathcal{P}_A(\lambda_B|\lambda_A)$ will then be similar to an ordinary MD simulation. In the limit of a huge number of interfaces, the crossing probabilities $\mathcal{P}_A(\lambda_{i+1}|\lambda_i)$ will approach one. This is also highly inefficient as the number of simulations will be vast.

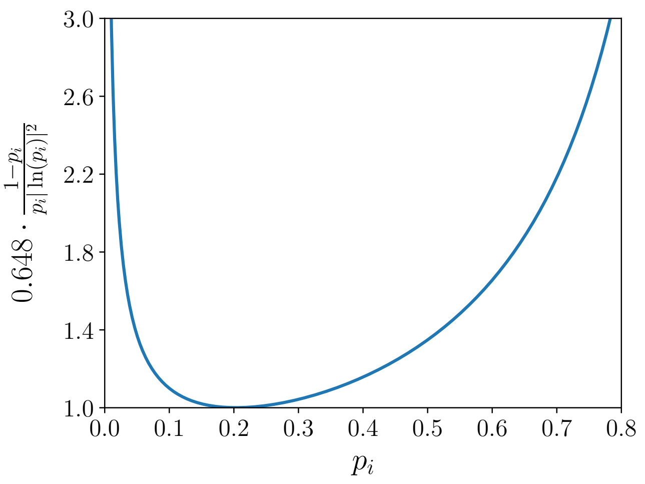


Figure 3: CPU efficiency time τ_{eff} versus crossing probability $p_i \equiv \mathcal{P}_A(\lambda_{i+1}|\lambda_i)$, see Equation (11). The vertical axis is normalized to the minimum at $p_i = 0.2$. Note that several assumptions are made [7, 22]. For example, the crossing probabilities are assumed to be the same for all i.

Van Erp [7] showed that an optimal value for the crossing probabilities in TIS can be found by invoking some assumptions that will not be presented here. The optimal value was calculated by minimizing the CPU efficiency time, which was shown to be

$$\tau_{\text{eff}} \propto \frac{1-p_i}{p_i|\ln p_i|^2}, \tag{11}$$

where the crossing probabilities, $p_i \equiv \mathcal{P}_A(\lambda_{i+1}|\lambda_i)$, are assumed to be the same for all i. The minimum occurs at $p_i = 0.2$, see Figure 3. In RETIS, however, the minimum value is assumed to be somewhat larger [22]. Although the theoretical optimal value of p_i relies on several assumptions that will not be truly fulfilled in a simulation, the plot in Figure 3 can still serve as guidance when deciding the number of interfaces and their placements.

2.6.5 Initialization

An obvious challenge with the Markov chain of paths is that before the first shooting move can be made to generate a new path, a path must already exist in the path ensemble. Each path ensemble $[i^+]$ needs an initial path, i.e., a path that crosses the interface λ_i. The same path can, in principle, be used for several ensembles, provided that it crosses λ_i in all the ensembles it serves as the initial path. The initial paths need to be established using some initialization procedure. Finding an initial reactive path for a rare event is not necessarily trivial, as the number of simulations required to observe just one barrier crossing in a straightforward MD approach is vast (see Section 2.6.4). Although the equilibrium properties of a system do not (or, at least, should not) depend on the initial

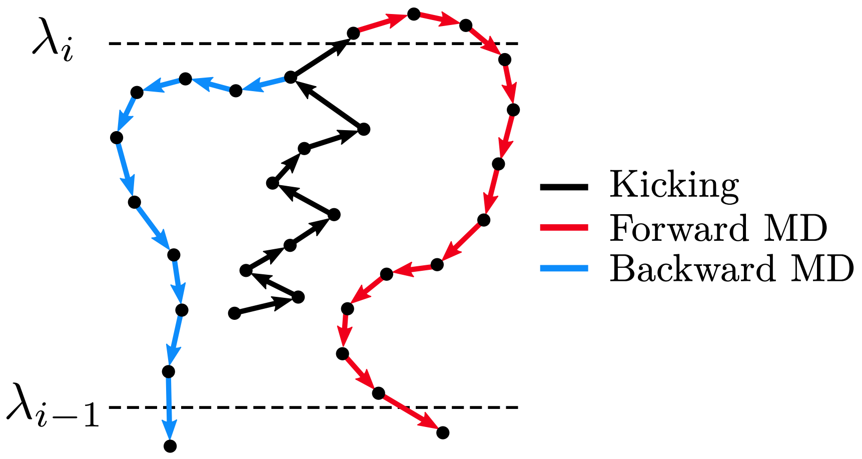


Figure 4: A configuration point is given as a starting point in the' kick' initialization procedure. Here, the starting point is the bottommost dot of the black line. New velocities are drawn randomly from a distribution (e.g., Maxwell–Boltzmann) and given to the point. Then one MD step is executed to find a new point. While kicking (black path), only forward-going steps with respect to the order parameter are accepted. These steps are repeated until a crossing with the interface λ_i occurs. Then, MD forward in time from the point immediately above λ_i (red path) and MD backward in time from the point immediately below λ_i (blue path) is performed, creating a path to the $[i^+]$ ensemble. The final path is then the reverse blue path plus the last segment of the black path plus the red path.

conditions, some initial conditions are more appropriate to choose concerning the physical process and the computational efficiency [14]. In any case, the initial path does not have to be very good and can even be unphysical.

One initialization procedure, called *kick*, works by searching for a crossing with interface λ_i (see Figure 4). Given that the system has some initial starting point, the 'kick' method performs the shooting move by modifying the velocities randomly according to some distribution (e.g., the Maxwell–Boltzmann distribution) and performing one MD step. If the MD step results in a point closer to the interface λ_i as measured by the RC, it is accepted. If it results in a point further away from λ_i, it is rejected. These steps are repeated until the crossing occurs. When the crossing is found, the point immediately before λ_i and the point immediately after the interface are kept. Then, forward MD integration from the point after the interface and backward MD integration from the point before the interface is performed, creating a path belonging to the $[i^+]$ ensemble. This procedure has a disadvantage in that the kicking may move over local energy barriers. In other words, it might try to climb a wall instead of taking a step back and moving beside it. Another initialization procedure, dubbed 'flick,' will be described in great detail in Chapter 3.

2.7 Replica Exchange Transition Interface Sampling

Replica exchange transition interface sampling (RETIS) is an improvement of TIS, developed by Van Erp [9]. The main differences to TIS are the introduction of the $[0^-]$ ensemble and the *swapping move*. The $[0^-]$ ensemble consists of all paths that start at λ_A, then go in the opposite direction of the reaction progress, and end at λ_A again. The reaction rate is calculated as in TIS using Equation (8), but the flux f_A is calculated differently. The $[0^-]$ ensemble replaces the initial MD simulation used in Equation (10). The flux is in RETIS calculated from the average path lengths of the $[0^-]$ and $[0^+]$ ensembles, $\left\langle t_{\text{path}}^{[0^-]} \right\rangle$ and $\left\langle t_{\text{path}}^{[0^+]} \right\rangle$, respectively, as [9]

$$f_A = \left(\left\langle t_{\text{path}}^{[0^-]} \right\rangle + \left\langle t_{\text{path}}^{[0^+]} \right\rangle \right)^{-1}. \tag{12}$$

The swapping move swaps paths between two adjacent ensembles. Or rather, it duplicates the last path in both ensembles, *then* swaps the duplicates (hence replica-exchange) if the acceptance criterion is met. A swap $[i^+] \leftrightarrow [(i+1)^+]$ is accepted if both paths are valid in both ensembles, i.e., if the $[i^+]$-path crosses λ_{i+1}. Such swaps are computationally cheap. The swap $[0^-] \leftrightarrow [0^+]$ is always accepted, and it is the only swapping move that requires force calculations (through MD steps). Such a move generates new paths by integrating backward in time from the first time slice for the $[0^-] \leftarrow [0^+]$ move and forward in time from the last time slice for the $[0^-] \rightarrow [0^+]$ move, see Figure 5. The swapping move significantly increases the sampling efficiency [9].

The RETIS algorithm is as follows. At each step, it is decided by some probability whether a series of shooting, time-reversal, or swapping moves will be performed. If shooting moves are chosen, all ensembles will be extended sequentially by performing the computationally expensive shooting move. If time-reversal moves are chosen, all ensembles will be updated sequentially by performing the computationally cheap time-reversal move. If swapping moves are chosen, an equal probability will decide whether the swaps $\{[0^-] \leftrightarrow [0^+], [1^+] \leftrightarrow [2^+], ...\}$ or the swaps $\{[0^+] \leftrightarrow [1^+], [2^+] \leftrightarrow [3^+], ...\}$ are performed. Whether the last ensemble, $[(n-1)^+]$, is swapped is then dependent on which of the series is chosen and on the parity of n. Each time $[0^-]$ and $[(n-1)^+]$ do not participate, they are left unchanged and recounted, like all the other swapping moves that are not accepted.

While TIS can run all path ensembles embarrassingly parallel, parallelization of RETIS is not so clear-cut. For example, the PyRETIS software implements RETIS fully sequentially [11, 12]. In TIS, the path ensembles are independent of each other, while they are not in RETIS due to the swapping move. Therefore, a straightforward parallelization of RETIS, where a separate computer node treats each ensemble, would lead to many nodes

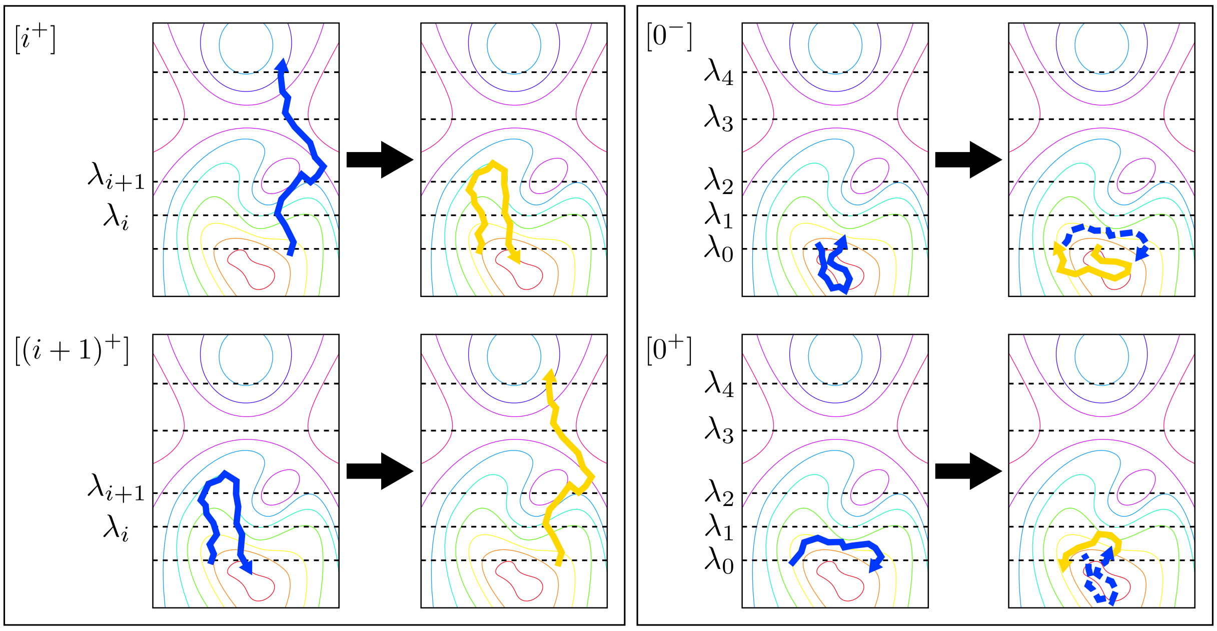


Figure 5: Illustrated example of the swapping move. Notations and labels are the same as in Figure 1. *Left panel*: A $[i^+] \leftrightarrow [(i+1)^+]$ swap. The left-hand side and the right-hand side of the panel show the last counted path before and after the move has been executed, respectively. Here, the $[i^+]$ path happens to cross λ_{i+1}, such that the move is accepted. If the path had not crossed λ_{i+1}, it would not have been accepted, and the old path would have been counted again, like in standard Metropolis Monte Carlo. *Right panel*: A $[0^-] \leftrightarrow [0^+]$ swap. The dashed blue arrows are shown as references. This swapping move is always accepted and generates new paths by integrating backward in time from the first time slice for the $[0^-] \leftarrow [0^+]$ move and forward in time from the last time slice for the $[0^-] \rightarrow [0^+]$ move.

being idle while waiting for longer paths to finish in other ensembles. In addition, the non-flexible path length algorithm mentioned in Section 2.6 results in considerable differences in the CPU time when performing a single path generation move. Nevertheless, a parallelizable RETIS method was recently (May 2022) developed [23].

2.8 PyRETIS: Rare Events in Python

PyRETIS is an open-source Python library for rare event molecular simulations, emphasizing methods based on TIS and RETIS. It utilizes both object-oriented and procedural programming and is developed by a research team at the Theoretical Chemistry Group at NTNU [11, 12, 24].

Figure 6 illustrates the main loop for a typical rare event in a PyRETIS simulation. The user decides the number of interfaces and their placements (Section 2.6.4). RETIS requires a valid initial trajectory for each path ensemble (Section 2.6.5). In PyRETIS, this can either be generated by the internal software or loaded from an existing trajectory.

2.8.1 Input File

The PyRETIS simulations can be set up and carried out by explicitly using the library in a Python script or by using an *input file*. All the user-specified parameters and settings are specified here. See the PyRETIS web-page for a full description of the input file [24].

The one-dimensional double-well potential (Figure 7) is given by

$$U = ax^4 - b(x-c)^2, \tag{13}$$

where x is the position and a, b, and c are parameters for the potential. The double-well potential parameters are specified in the `Potential` section of the input file:

Example potential section

```
Potential
---------
class = DoubleWell
parameter a = 1.0
parameter b = 2.0
parameter c = 0.0
```

Choosing which simulation PyRETIS will run is done in the `Simulation` section:

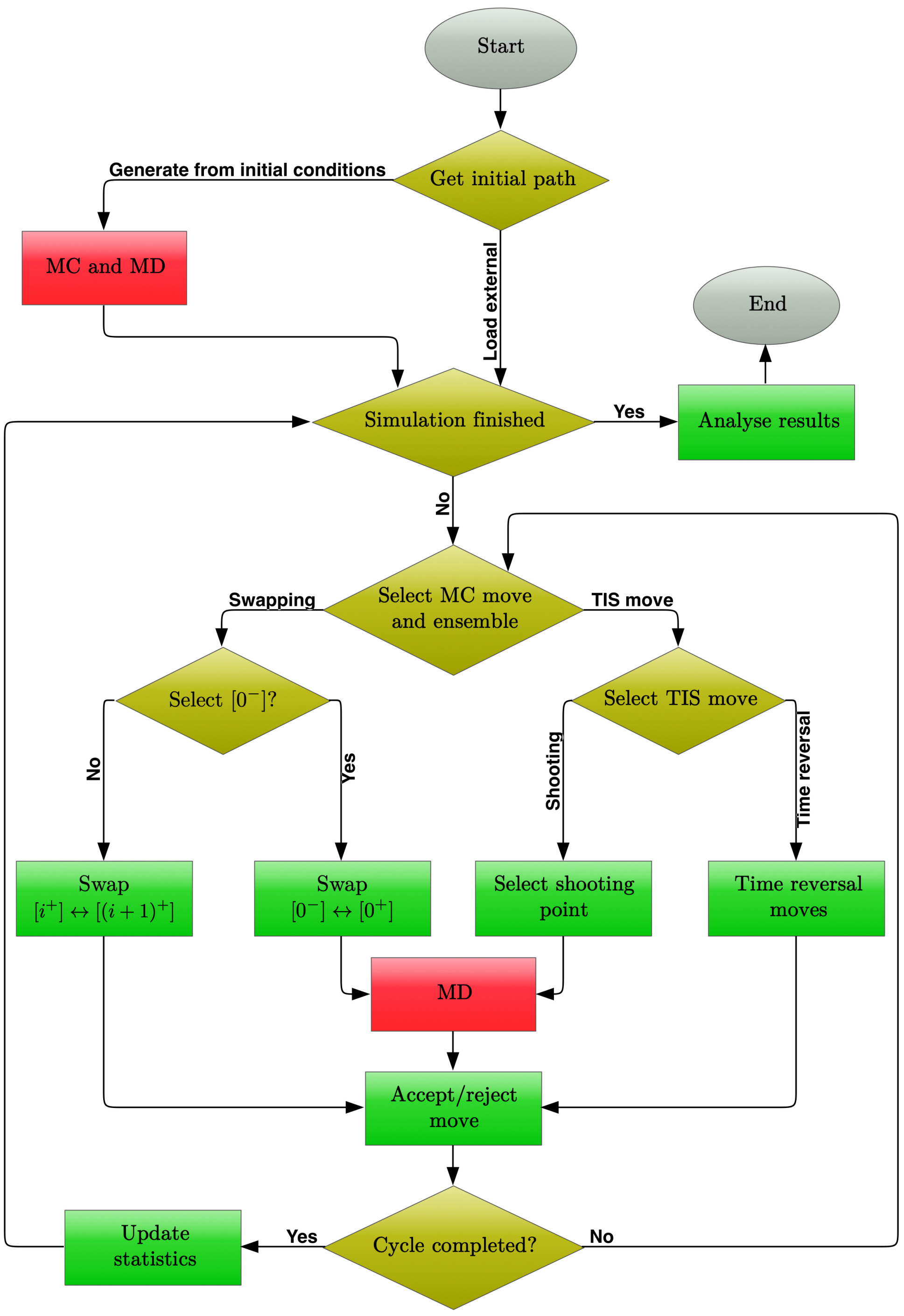


Figure 6: Flowchart of a rare event simulation with PyRETIS. After the initial paths have been generated or loaded, the main loop is entered. In the main loop, new trajectories are generated from previous ones by randomly selecting an MC move (swap, shoot, or time-reversal). The red rectangles are computationally costly operations, while the green ones are cheap. The yellow rhombuses are choices made by the user or by the algorithm. The figure is based on Lervik et al. [11].

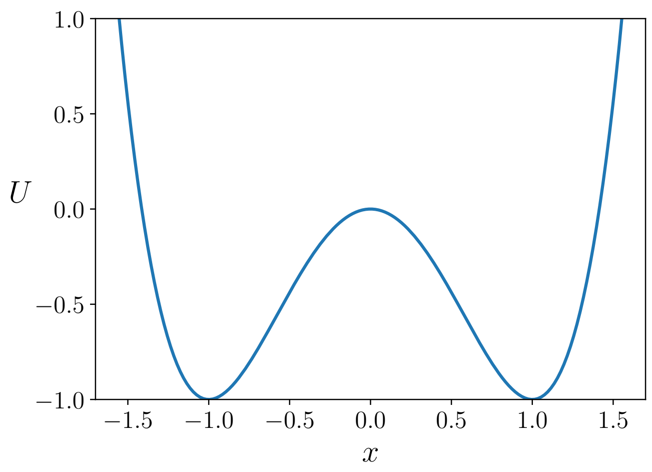


Figure 7: The one-dimensional double-well potential, Equation (13), with $a = 1$, $b = 2$, and $c = 0$. Two stable states are located around $x = -1$ and $x = 1$, which correspond to where the system is likely to be.

Example simulation section

```
Simulation
----------
task = retis
steps = 20000
interfaces = [-0.9, -0.8, -0.7, -0.6,
              -0.5, -0.4, -0.3, 1.0]
```

Note that the leftmost and the rightmost number of the `interfaces` list defines the stable state definitions in (5) (here: $\lambda_A = -0.9$ and $\lambda_B = 1.0$).

The `Initial-path` section specifies how the initial path (Section 2.6.5) for a TIS/RETIS simulation should be generated:

Example initial path section

```
Initial-path settings
---------------------
method = kick
kick-from = previous
```

Three schemes are currently available in PyRETIS: ‘kick,’ ‘load,’ and ‘restart.’ Chapter 3 addresses a new initialization scheme, referred to as ‘flick.’

2.9 GROMACS

Gromacs is a free, open-source software suite for high-performance molecular dynamics and output analysis [25].

2.9.1 GROMACS in PyRETIS

Gromacs can be used as an external engine in PyRETIS by specifying the class `gromacs` in the `Engine` section of the input file (Section 2.8.1):

Example engine section

```
Engine settings
---------------
class = gromacs
gmx = gmx
mdrun = gmx mdrun
input_path = gromacs_input
timestep = 0.002
subcycles = 1
gmx_format = gro
```

The `gromacs2` engine is, at the time of writing, an experimental engine implemented into PyRETIS, which is faster than the standard `gromacs` engine. Note that `gromacs` and `gromacs2` are PyRETIS engines that both use Gromacs, but differently, to integrate the equations of motion.

The `input_path` keyword sets the directory where PyRETIS will look for input files to use with Gromacs. Inside this folder, the following files must be present: `conf.gro` (the initial configuration), `grompp.mdp` (the molecular dynamics parameters), and `topol.top` (the topology for the system). In Section 4.2.2, we discuss one of the keywords in the `grompp.mdp` file, namely `integrator`.

2.10 Dissociation of Sodium Chloride in Water

In Section 4.2, the new initialization algorithm is tested on a realistic full-atom NaCl system consisting of one Na^+ ion and one Cl^- ion immersed in a bath of water molecules, using the Gromacs engine. The ion-ion distance $r_{\text{ion}} = |\mathbf{r}_{\text{Na}^+} - \mathbf{r}_{\text{Cl}^-}|$ is used as reaction coordinate.

Ballard and Dellago [19] have shown that the ion-ion distance alone is a poor reaction coordinate and that the surrounding solvent must be taken into account to describe the progress of the reaction. With the ultimate goal of finding a reaction coordinate for the event, a complete set of solvent variables that jointly account for the dissociation process is still unknown. Ballard and Dellago studied a system consisting of one Na^+ ion, one Cl^- ion and $N_{\text{w}} = 216$ water molecules. They modeled the ion-ion and ion-water interactions using the OPLS force field, which includes short-ranged Lennard-Jones and long-ranged Coulomb terms. Their simulations were performed sampling the NVT ensemble, with constant temperature $T = 300$ K and volume $V = (18.64\ \text{Å})^3$, using Gromacs with time

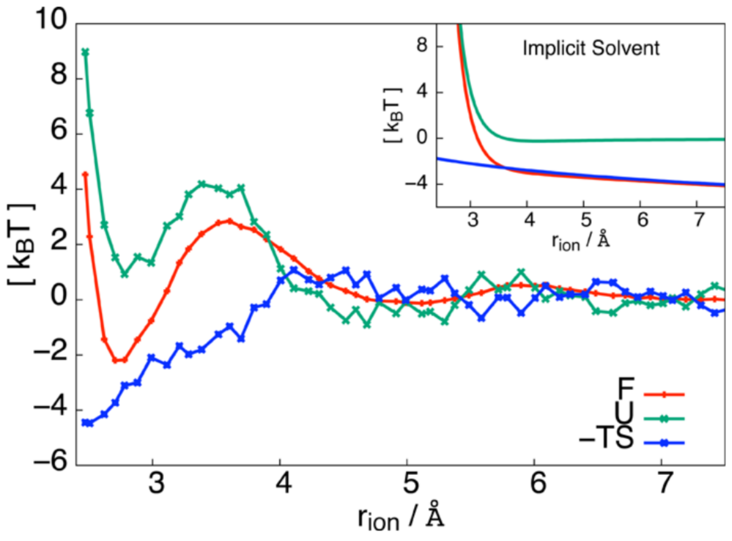


Figure 8: Thermodynamics of ionic dissociation in water. The Helmholtz free energy F (red line) exhibits a stable associated state at $r_{\text{ion}} = 2.7\,\text{Å}$, separated from the dissociated state by a free energy barrier of $5\,k_{\text{B}}T$ with a peak at $r_{\text{ion}} \approx 3.6$ Å. The average energy U (green line) and the negative entropy (blue line) are also plotted. The figure is taken from Ballard and Dellago [19].

step $\delta t = 2$ fs. They calculated the Helmholtz free energy along the interionic distance as

$$F(r_{\text{ion}}) = -k_{\text{B}}T \ln\left(\frac{\rho(r_{\text{ion}})}{\rho_{\text{ref}}}\right), \tag{14}$$

where $\rho(r_{\text{ion}})$ is the probability density and ρ_{ref} is a reference value that will vanish when considering free energy differences. They did the calculation by histogramming r_{ion} from the concatenated trajectories, revealing the red line in Figure 8. From there, they identified the associated state as all configurations with $r_{\text{ion}} < 3.2\,\text{Å}$, and the dissociated state as $r_{\text{ion}} > 4.4\,\text{Å}$. Also plotted is the average energy $U(r_{\text{ion}}) = \langle E\rangle_{r_{\text{ion}}} - \langle E\rangle_{\infty}$ and the entropy S, which is identified from

$$F(r_{\text{ion}}) = U(r_{\text{ion}}) - TS(r_{\text{ion}}). \tag{15}$$

The master's thesis by Konrad Wilke calculates the crossing probabilities of five path ensembles using PyRETIS for a similar NaCl system with 908 water molecules [26]. This is the same system we study in Section 4.2. The interfaces where chosen to be $\{\lambda_i/\text{nm}\} = \{0.32, 0.34, 0.36, 0.38, 0.41, 0.70\}$, where $i = 0, 1, ..., 5$. Using a larger value ($\lambda_B = 0.70$ nm) for the dissociated state than Ballard and Dellago [19] ($\lambda_B = 0.44$ nm) should not matter, as the barrier has been crossed and hence the crossing probability will be flat beyond this point (Figure 8). The crossing probabilities were computed to $\{\mathcal{P}_A(\lambda_{i+1}|\lambda_i)\} \approx \{0.26, 0.45, 0.66, 0.77, 0.62\}$. Using Figure 3 and the discussion in Cabriolu et at. [22] as

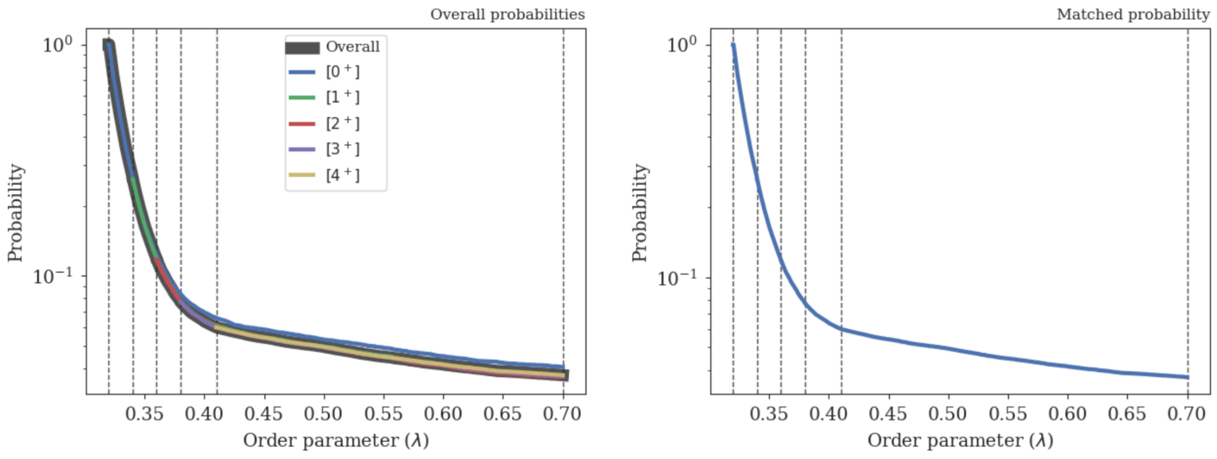


Figure 9: The OP is the ion-ion distance in nanometers. The figure is taken from Wilke [26].

Table 1: Summary of the main results for the NaCl system [26].

Property	Value (%)	Relative error (%)
Crossing probability $\mathcal{P}_A(\lambda_B\|\lambda_A)$	3.7	6.2
Flux f_A (1/gromacs)	17	1.5
Rate constant k_{AB} (1/gromacs)	0.64	6.4

guidance, one can argue that the crossing probabilities seems 'reasonable.' Figure 9 shows a standard PyRETIS output generated by the `pyretisanalyse` application. The main results from the analysis are reported in Table 1. The left panel in Figure 9 shows the individual probabilities, while the right panel shows only the matched overall probability, obtained by matching and aligning the individual probabilities.

Figure 10 shows a simplified illustration of the salt dissociating. We see that the solvent goes through a structural change during the dissociation, which also highlights the problem of finding a proper OP that fully describes the reaction. Note that a phase point is not uniquely defined for a given order parameter $\lambda = r_{\text{ion}}$, as there are infinitely many phase points that give the same order parameter. For example, changing the position or velocity of any water molecule, moving all the water molecules, or translating or rotating the ion-ion pair all reveal the same OP.

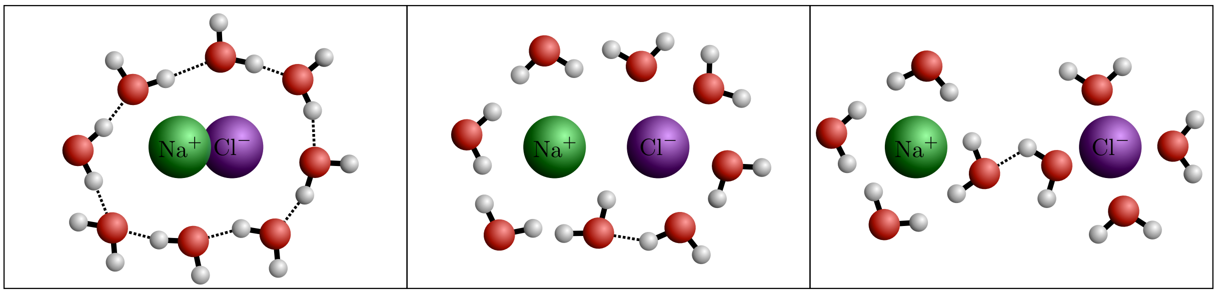


Figure 10: Schematic illustration of dissociation of NaCl in water (dotted lines indicate hydrogen bonds). The left panel shows the associated state ($r_{\text{ion}} < \lambda_A$); the middle panel shows the barrier crossing, which requires the breakage of hydrogen bonds; the right panel shows dissociated state ($r_{\text{ion}} > \lambda_B$). The figure is based on Cabriolu et al. [22].

3 Description of the New Initialization Method

As noted in Section 2.6.5, an initial path for each path ensemble must be generated to perform Markov chain Monte Carlo. Each ensemble does not necessarily need to have a unique path, as the same path, in principle, can be used in several ensembles. Furthermore, as previously noted, the equilibrium properties do not depend on the initial conditions. However, because of computational efficiency and limitations in computational power, it is desirable to use initial conditions that are relatively probable and not unphysical.

The python code I have written as an implementation of the new method builds on ideas and personal notes given by Titus van Erp and Raffaela Cabriolu. The purpose of introducing a new method is to provide a way to generate more likely (i.e., lower energy) paths for the PyRETIS library than, for example, the 'kick' method. We have dubbed the new method 'flick,' as it is a softer way to produce initial paths than by 'kicking' (thanks to Daniel Zhang for suggesting this name).

The output of this method is a set of paths starting in A and ending in A or B. The algorithm will, at minimum, produce one reactive path, connecting state A and B. In principle, only λ_A and λ_B need to be known, but given a set of interfaces $\{\lambda_i\}$, the algorithm will suggest the best possible initial path for the different $[i^+]$ ensembles. This could be a reactive path since that is a valid path for all $[i^+]$ ensembles. However, paths from the set that crosses λ_i but do not reach λ_B might have a higher path probability. The aim is to find the most probable path for each ensemble based on an energy criterion.

This method requires some user inputs, namely a number $\alpha \in [1/2, 1]$, two integers $n_{\text{del}} \geq 1$ and $n_{\text{reac}} \geq 1$, and a list of one or more points in configuration space $\{\mathbf{R}_1, \mathbf{R}_2, ...\}$, with order parameters in the range $\lambda(\mathbf{R}) \in (\lambda_A, \lambda_B)$. Strictly speaking, it is the corresponding phase points $\{\mathbf{x}_1, \mathbf{x}_2, ...\}$ that should have order parameters in the range. However, this will rarely be a problem since the order parameter is often a function of the configuration and not the velocity. The user can opt to input a full path instead of configuration points. The meaning of the user inputs should become apparent in the description of the algorithm. The algorithm is illustrated in Figure 12, and the description is as follows.

Algorithm: The new initialization method ('flick')

1. Generate trajectories from the user-provided configuration points by assigning velocities according to some probability distribution and then integrating backward in time and forward in time using MD. A trajectory's forward and

backward parts are stopped once they cross either λ_A or λ_B. Each path then consists of many phase points in the interval (λ_A, λ_B) and two phase points (the end-points) outside the interval.

2. Categorize the trajectories according to which state they start and end in, i.e., AA, BB, or AB. If a trajectory starts in B and ends in A (that is, it is a BA-trajectory), a time-reversal move (Section 2.5.2) is applied such that it becomes an AB-trajectory. The sole purpose of BB-trajectories is that they can assist in generating AA- and AB-trajectories. The BB category itself, however, is of no interest and will be dismissed after the completion of the algorithm.

3. Choose randomly one of the groups AA, BB, or AB. The default option is that each group has a $1/3$ probability of being selected. If the group is empty, choose again.

4. Each path in the chosen group consists of phase points. Put all the points (except the end-points that are outside the interval (λ_A, λ_B)) into one single list, and sort the list:

   a) For the AA group, sort the points by decreasing $\lambda(\mathbf{x})$.

   b) For the BB group, sort the points by increasing $\lambda(\mathbf{x})$.

   c) For the AB group, sort the points by the number of MD steps away from the shooting point that created the path to which it belongs. Here, it is assumed that a point close to a previously successful shooting point is more likely to create a new AB-trajectory than a point far away.

   Points that are equal regarding their sorting-measure are scrambled in random order each time the list is updated.

5. Let N be the number of points in the chosen group. Associate a unique rank $j = 0, 1, ..., N-1$ to each point in the list corresponding to the sorting order. Calculate the probabilities for each point based on its ranking j and the user-specified parameter α according to

$$p_j = \left(\frac{1-q}{1-q^N}\right) q^j, \tag{16}$$

   where

$$q = \left(\frac{1-\alpha}{\alpha}\right)^{2/N}, \tag{17}$$

and $\alpha \in (1/2, 1]$. If $\alpha = 1/2$, all points get an equal probability, $p_j = 1/N$, by definition. If $\alpha \notin [1/2, 1]$, an error is returned.

6. Choose a random point $\mathbf{x}_j$ according to the corresponding probability p_j, using a biased roulette wheel algorithm, illustrated in Figure 11. The roulette wheel is divided into N segments, each with a size proportional to their probabilities $p_0, p_1, ..., p_{N-1}$. A random angle is drawn, and the point corresponding to that angle is chosen.

7. Do the shooting move from the chosen point and categorize the path correspondingly to AA, BB, or AB.

8. Do the deleting procedure for the group that the path of the previous step belongs. The deleting procedure requires the user-input n_{del}, which is a positive integer. For group AA, the procedure deletes a path if there exist at least n_{del} other AA-paths that have both higher maximum order parameter $\lambda_{\max}$ and lower path energy E. The path energy is here defined as the average total energy of all the phase points in the path (see Equation (27)). For group BB, a path is deleted if there are at least n_{del} other paths in BB that have both lower $\lambda_{\min}$ and lower path energy. For group AB, a path is deleted if there are at least n_{del} other paths in AB that have lower path energy. See the conditions in (26).

9. Repeat steps 3–8 until some stopping criterion is reached, or the user decides to stop the initialization by typing some command (provided that some useful information is printed during the steps). In the current version, the stopping criterion is to acquire n_{reac} AB-paths.

10. The BB-trajectories are deleted after the stopping criterion has been reached, such that one is left with at least one reactive path and possibly some AA-trajectories. All the trajectories can now be stored in one list of trajectories, sorted by increasing path energy. In the RETIS algorithm, the chosen initial path is one with the lowest path energy, which is valid for the path ensemble in question.

11. For the $[0^-]$ ensemble, we can simply take any path of the AA or AB groups and integrate the equations of motion backward in time starting from the first phase point of this path. This is similar to how a new $[0^-]$ path is generated via the replica-exchange move (see Figure 5, right panel). It makes sense

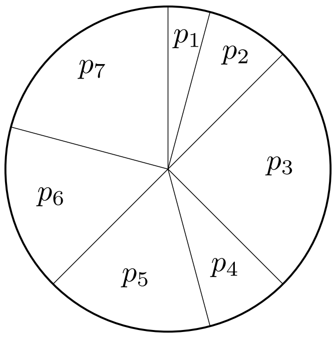


Figure 11: A biased roulette wheel algorithm chooses points according to their probabilities. Here, the roulette wheel is divided into seven segments. Each segment has a size proportional to the probabilities $p_1, p_2, ..., p_7$. There are several ways to implement this algorithm. One way is to draw a random number (corresponding to an angle) from a uniform distribution, say a number between 0 and 2π, and the point corresponding to the angle is chosen. This algorithm is, of course, implemented into popular python libraries such as NumPy (`numpy.random.Generator.choice`) [27].

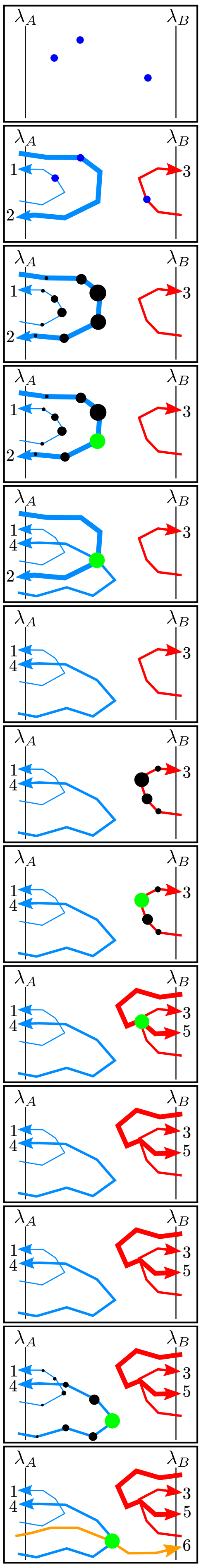
nonetheless to take the most likely path among the AA or AB groups for which we use the same energy-based criterion as in step 10.

The value of q in Equation (17) tells us that the probability of selecting a point with a certain rank is $1/q$ times higher than selecting the point one step lower in the rank [10]. So in principle, we could have opted to let the user provide q directly as an input parameter, where $0 < q \leq 1$. The lower the value of q, the more elitist the selection procedure becomes. $q = 1$ means that all points have the same probability of being selected, with a probability equal to $1/N$. On the other hand, it is not very intuitive to choose a reasonable value of q. Additionally, the fact that the number of points N in the groups is not constant could imply that two similar points suddenly get very different selection probabilities due to additional points getting ranked between these. These are the reasons why we define q indirectly via Equation (17), where α has the following meaning,

$$\alpha \equiv \begin{cases} \text{the probability to select a point from the upper} \\ \text{half } (j = 0, 1, ..., N/2 - 1) \text{ of the ranking list.} \end{cases} \tag{18}$$

Based on the definition of α, $q = q(N, \alpha)$ can be determined as follows. Let

$$s(N) \equiv \sum_{j=0}^{N-1} q^j, \tag{19}$$



User-provided configuration points — User inputs: $n_{\text{del}} = 1$, $n_{\text{reac}} = 1$

Step 1-2 — Category AA; Category BB

Step 3-5 — Category AA chosen

Step 6 — Green point chosen

Step 7 — New path: AA

Step 8 — Deleting procedure for category AA:
$E_1 < E_4 < E_2$
$\lambda_{\max,1} < \lambda_{\max,2} < \lambda_{\max,4}$
Path 2 deleted because 'delete criterion' fulfilled 1 time for path 2:
$\lambda_{\max,2} < \lambda_{\max,4}$ and $E_2 > E_4$

Step 3-5 — Category BB chosen

Step 6 — Green point chosen

Step 7 — New path: BB

Step 8 — Deleting procedure for category BB:
$E_3 < E_5$
$\lambda_{\min,3} > \lambda_{\min,5}$
Nothing deleted because 'delete criterion' never fulfilled

Step 3-5 — Category AA chosen

Step 6 — Green point chosen

Step 7 — New path: AB
Initialization finished because stopping criterion attained

Figure 12: The 'Flick algorithm' steps 1–9 illustrated (with arbitrary α). Larger point size indicate larger probability p_j. Thicker line indicate higher path energy E_k. 'Delete criterion' in (26a) and (26b): progressed further and lower E_k.

such that

$$\alpha = \frac{s(N/2)}{s(N)}. \tag{20}$$

Using standard maths, Equation (19) can be rewritten to

$$s(N) = \frac{1-q^N}{1-q}, \tag{21}$$

such that

$$\alpha = \frac{1-q^{N/2}}{1-q^N} \quad \text{or} \quad \alpha = \frac{1-x}{1-x^2}, \tag{22}$$

where

$$x \equiv q^{N/2} \quad \text{or} \quad q = x^{2/N}, \tag{23}$$

(not to be confused the with position i the x-direction). Since $1 - x^2 = (1 + x)(1 - x)$, we get

$$\alpha = \frac{1}{1+x} \quad \text{or} \quad x = \frac{1-\alpha}{\alpha}, \tag{24}$$

such that

$$q = \left(\frac{1-\alpha}{\alpha}\right)^{2/N},$$

which is identical to Equation (17) in the algorithm. Finally, we normalize the weights such that $p_j = \text{const.} \times q^j$ is the probability to select point $\mathbf{x}_j$. The normalization constant follows from $\sum_j p_j = 1$, and therefore,

$$\text{const.} = \frac{1}{\sum_{j=0}^{N-1} q^j} = \frac{1-q}{1-q^N}, \tag{25}$$

such that

$$p_j = \left(\frac{1-q}{1-q^N}\right) q^j,$$

which is identical to Equation (16).

The parameter n_{del} decides how many times the 'delete criterion' in step 8 has to be fulfilled in order to delete a path from its group. The 'delete criterion' considers a path k and compares it to all the other paths belonging to the same category. It is fulfilled if there exists a path l $(\neq k)$ belonging to the same category as k that satisfies the following:

$$\begin{cases} (\lambda_{\max,k} < \lambda_{\max,l} \ \text{ and } \ E_k > E_l) \ \text{ if } \ k \in AA & \text{(26a)} \\ (\lambda_{\min,k} > \lambda_{\min,l} \ \text{ and } \ E_k > E_l) \ \text{ if } \ k \in BB & \text{(26b)} \\ E_k > E_l \ \text{ if } \ k \in AB, & \text{(26c)} \end{cases}$$

where

$$E_k \equiv \frac{1}{N_k} \sum_{j=1}^{N_k} \left(K(\mathbf{x}_j) + U(\mathbf{x}_j) \right) \tag{27}$$

is the *path energy* of path k, and N_k is the number of phase points in path k. If the number of paths $l_1, l_2, l_3, \ldots$ satisfying the 'delete criterion' in (26) is greater than or equal to n_{del}, path k is deleted.

In other words, for categories AA and BB, the delete criterion for a path k is fulfilled if there exists a path l belonging to the same category that has progressed further and has lower energy. For category AB, the energy criterion is sufficient. The n_{del} parameter can be interpreted as a measure of how 'strict' the algorithm is in deleting paths—the lower the value of n_{del}, the lower the threshold of deleting. Consequently, a higher value of n_{del} will lead to more points to choose from in step 6 of the algorithm and therefore also requires more disk space.

In step 4, we said that points that are equal regarding their sorting-measure are scrambled in random order each time the list updates. However, we could instead have done the following. If points $j = k, k+1, \ldots, k+l$ have the same sorting-measure, we give all these points the same probability using the average of the original probabilities p_j:

$$p_k = p_{k+1} = \ldots = p_{k+l} = \frac{\sum_{j=k}^{k+l} p_j}{l+1}. \tag{28}$$

Scrambling or using the expression in (28) are presumably equally good. Ultimately, we chose to scramble as it was easier to implement.

There is a potential drawback with the path energy definition in Equation (27). A path that stays long in a low energy configuration and then moves past a very high energy barrier will still reveal low path energy, as it takes the average of all the phase points. However, if λ_A and λ_B are chosen sensibly, this will probably not happen.

The energy criterion in (26) might also reveal a weakness in another aspect. We added this criterion to get rid of very high-energy paths. For example, a too fast and aggressive approach for obtaining a transition can lead to overlapping molecules, resulting in high energies—this has been observed in other studies with the 'kick' approach. Based on the Boltzmann weight, $\exp(-\beta E)$, it also seems to make sense to eliminate paths with high energies. On the other hand, we must also account for entropy when generating paths, as low-energy paths might not be as likely as paths with some higher energy. Water, for example, is liquid at room temperature even if a configuration point representing an ice crystal is lower in energy. It is, therefore, an open question whether the energy criterion in the selection approach is not overshooting its goal of producing good paths by selecting paths that are too low in energy compared to the average transition path.

The settings of the 'flick' method are specified in the `Initial-path` section of the input file:

Example initial path section

```
Initial-path
------------
method = flick
reactive_paths = 5
alpha = 0.8
n_del = 2
p_aa = 0.4
p_bb = 0.4
p_ab = 0.2
```

In the current version of the code, only one user-provided configuration point is possible. It should be located the same place as in the 'kick' method: for the Gromacs engine, for instance, in the `input_path` folder specified in the `Engine` section, as `conf.gro` or `conf.g96`. The parameter `alpha` corresponds to α, and is a required input. The parameter `n_del` corresponds to n_{del}, and is set to 1 if not specified. The parameters `p_aa`, `p_bb`, and `p_ab` correspond to the probabilities of choosing a group in step 3 of the algorithm, and are optional. These probabilities, p_{AA}, p_{BB}, and p_{AB}, must sum to one, and are each set to $1/3$ if not specified. In the current version of the code, the initialization is finished (terminated) when a given number of reactive trajectories n_{reac}, connecting A and B, is found, specified by the `reactive_paths` parameter.

As previously noted, the equilibrium properties of a system are independent of the initial conditions. However, it is advantageous to use realistic initial conditions concerning computational efficiency. It is evident, then, that the initialization procedure itself cannot take an extremely long time to run compared to the rest of the RETIS algorithm because then the expenses would absorb the profits.

A technical note regarding the implementation of this algorithm is that significantly more storage is required compared to, e.g., the 'kick' algorithm. This is because the 'flick' method keeps several paths for comparison. One path might require several gigabytes of storage. Moreover, it is not known in advance exactly how much storage will be needed. Therefore, when bookkeeping these paths, the paths can not be stored in the random-access memory (RAM); but need to be stored on the storage, like a solid-state drive (SSD).

4 | Results and Discussion

The implementation of the new initialization method presented in Chapter 3 is virtually finished and has been tested on a 1D single-particle system (Section 4.1) and a 3D many-particle system (Section 4.2).

4.1 1D Double-Well Potential

The single-particle one-dimensional double-well potential system (see Figure 7) has been tested using the internal Langevin engine in PyRETIS. The tests are presented in the examples below. The order parameter is chosen to be the position (in the x direction), specified in the `Orderparameter` section of the input file, and the units are dimensionless (reduced).

4.1.1 Example 1

The 'Example 1' input file below has been used to simulate the paths presented in Figure 13. The `steps=0` keyword means that we are producing only initial paths, i.e., zero RETIS-steps. Note especially the `Initial-path` section. The initial configuration point is given as a separate file ('example1.xyz'), referenced in the `Particles` section.

A notable observation in Figure 13 is that only the second path got deleted by the 'deleting procedure' in step 8 of the algorithm. Only one path gets deleted because with a large α, the algorithm is likely to pick a point close to $\lambda_{\max} = x_{\max}$. This is because $x_{\max}$ always occurs when $\dot{x} = 0$ in the single-particle 1D system, that is, at a point with only potential energy. Hence, when a point with close to zero velocity is given a new velocity (and thus, more energy), it will, in all likelihood, result in a path with higher $x_{\max}$.

4.1.2 Example 2

When a new path that both starts and ends in A (or B) is generated by the new initialization method, it gets added to the group of paths belonging to category AA (or BB). Since the deleting procedure only deletes paths if both the conditions in (26a) (or (26b)) are fulfilled, the path that has the highest $\lambda_{\max}$ in AA (or lowest $\lambda_{\min}$ in BB) can never be deleted. Consequently, this value strictly increases (or decreases) as a function of the *step number*, where the step number is the number of times step 9 in the algorithm has been repeated. This 'super-$\lambda_{\max}$' value, referred to as $\lambda^*_{\max}$, is plotted as function of the step number in Figure 14 (top row). The 'super-$\lambda_{\min}$' value, referred to as $\lambda^*_{\min}$, is plotted in Figure 14 (bottom row). The values of α are chosen somewhat arbitrarily but in a fashion to demonstrate the difference between a low and a high value.

Input file: Example 1

```
Retis 1D example
================

Simulation
----------
task = retis
steps = 0
interfaces = [-0.9, -0.8, -0.7, -0.6,
              -0.5, -0.4, -0.3, 1.0]

System
------
units = reduced
dimensions = 1
temperature = 0.1

Box settings
------------
periodic = [False]

Engine settings
---------------
class = Langevin
timestep = 0.002
gamma = 0.3
high_friction = False
seed = 1

TIS settings
------------
freq = 0.5
maxlength = 20000
aimless = True
allowmaxlength = False
zero_momentum = False
rescale_energy = False
sigma_v = -1
seed = 1

RETIS settings
--------------
swapfreq = 0.5
relative_shoots = None
nullmoves = True
swapsimul = True

Initial-path settings
---------------------
method = flick
n_del = 1
reactive_paths = 1
alpha = 0.9999

Particles
---------
position = {'input_file':
            'example1.xyz'}
velocity = {'generate': 'maxwell',
            'momentum': False,
            'seed': 1}
mass = {'Ar': 1.0}
name = ['Ar']
ptype = [0]

Forcefield settings
-------------------
description = 1D double well

Potential
---------
class = DoubleWell
a = 1.0
b = 2.0
c = 0.0

Orderparameter
--------------
class = Position
dim = x
index = 0
periodic = False

Output settings
---------------
trajectory-file = 1
energy-file = 1
order-file = 1
```

example1.xyz

```
1

Ar -0.75 0 0
```

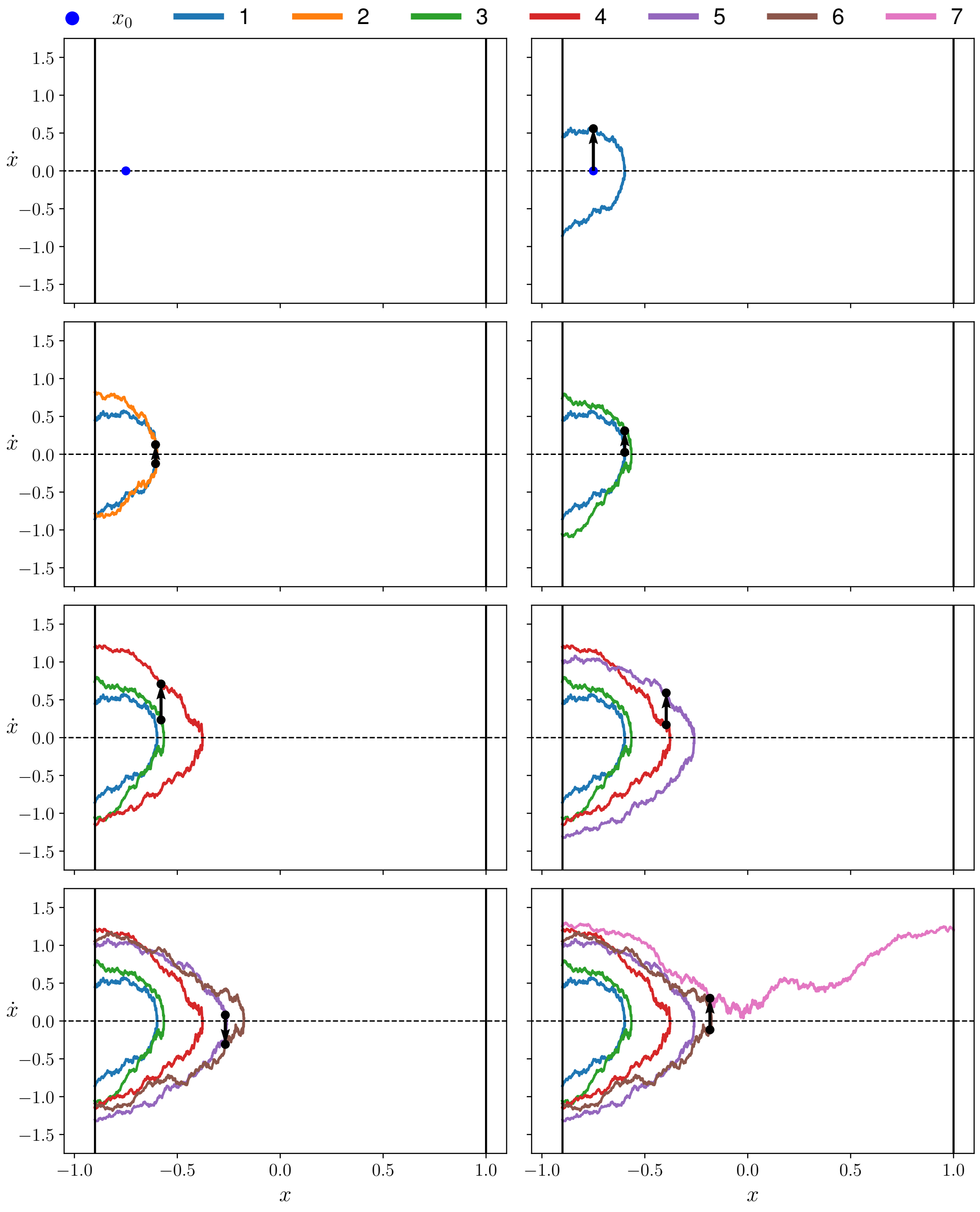


Figure 13: A 1D simulation example shows how the steps achieving a reactive path might look like in phase space. Here, one configuration point is provided by the user: $x_0 = -0.75$. The inputs are the same as in the 'Example 1' input file, but with α chosen extremely close to one (`alpha=0.9999`): this is to achieve a reaction in few simulation steps. The label numbers refer to the path creation order. Observe that the second path gets deleted according to the criteria in (26a). The arrow in each step shows the randomly (Maxwellian) chosen change in the velocity. The arrows point from the randomly chosen phase point, which is chosen using the probabilities in Equation (16). The stable state conditions $\lambda_A = x_A = -0.9$ and $\lambda_B = x_B = 1.0$ are also shown.

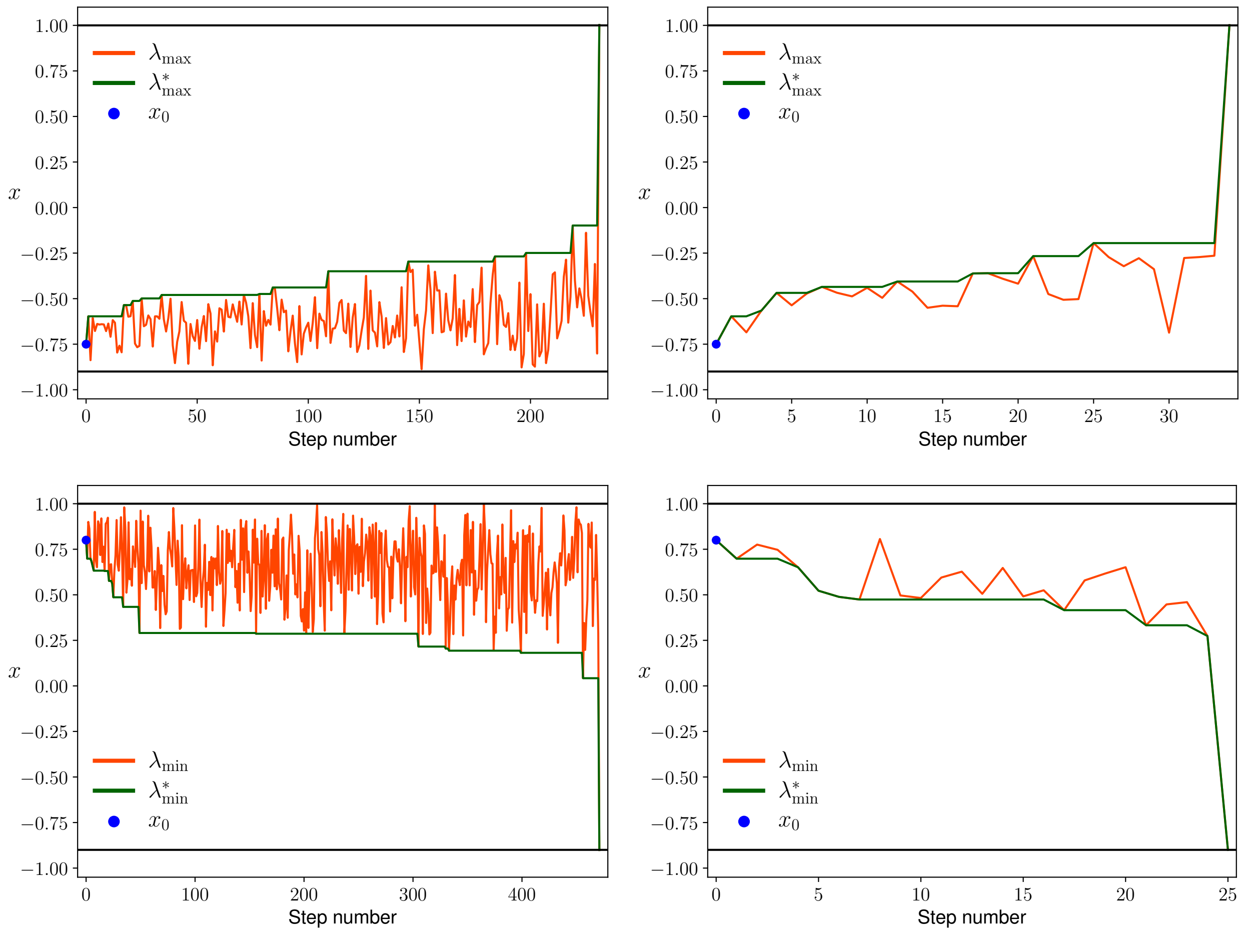


Figure 14: Simulation examples in 1D highlights the difference between a high (close to 1) and a low (close to 1/2) value of α and a high (close to λ_B) and a low (close to λ_A) value of the OP of the user-provided configuration point in the 'flick' method. $\lambda_{\max}$ (or $\lambda_{\min}$) is the OP extremum of the new path created in each step. $\lambda^*_{\max}$ (or $\lambda^*_{\min}$) is the highest $\lambda_{\max}$ (or lowest $\lambda^*_{\min}$) in group AA (or BB) in each step. Here, the parameters are the same as in the 'Example 1' input file, but with different values of α and the user-provided configuration point x_0. The step number is the number of times step 9 in the algorithm have been repeated. *Top row:* $x_0 = -0.75$, *bottom row:* $x_0 = 0.80$, *left column:* $\alpha = 0.70$, *right column:* $\alpha = 0.95$.

We see in Figure 14 that a higher value of α results in a faster convergence (i.e., fewer simulation steps) to a reactive path. As mentioned earlier, a large α makes the new method similar to the 'kick' method, described in Section 2.6.5. In realistic systems, such fast creations of reactive paths (category AB paths) are likely to lead to high path energy, as defined in Equation (27).

4.1.3 Energy Comparison of Initial Paths

Figure 15 shows the path energy for each path ensemble for 15 simulations with 'flick' and 15 simulations with 'kick.' The input files are the same as in the 'Example 1' input file, but with different `Initial-path` sections:

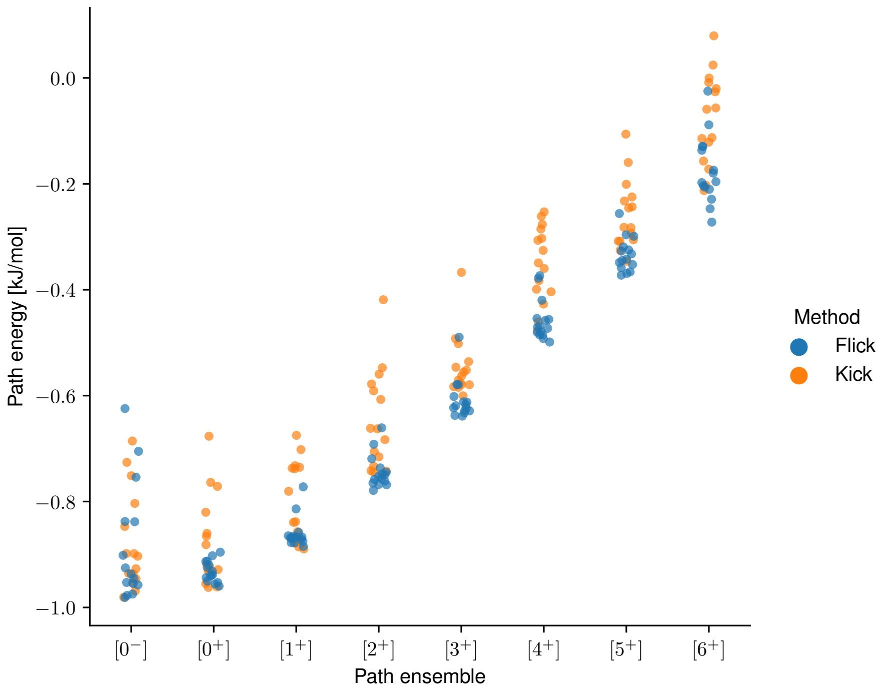
Initial-path section, 'flick'

```
Initial-path
------------
method = flick
reactive_paths = 1
alpha = 0.8
n_del = 1
```

Initial-path section, 'kick'

```
Initial-path
------------
method = kick
kick-from = initial
```

We see that the median energy in all the $[i^{+}]$ path ensembles is lower for 'flick' than for 'kick.' The path energy (average total energy of all the phase points) can, of course, never go below negative one, as is evident by looking at the double-well potential in Figure 7.



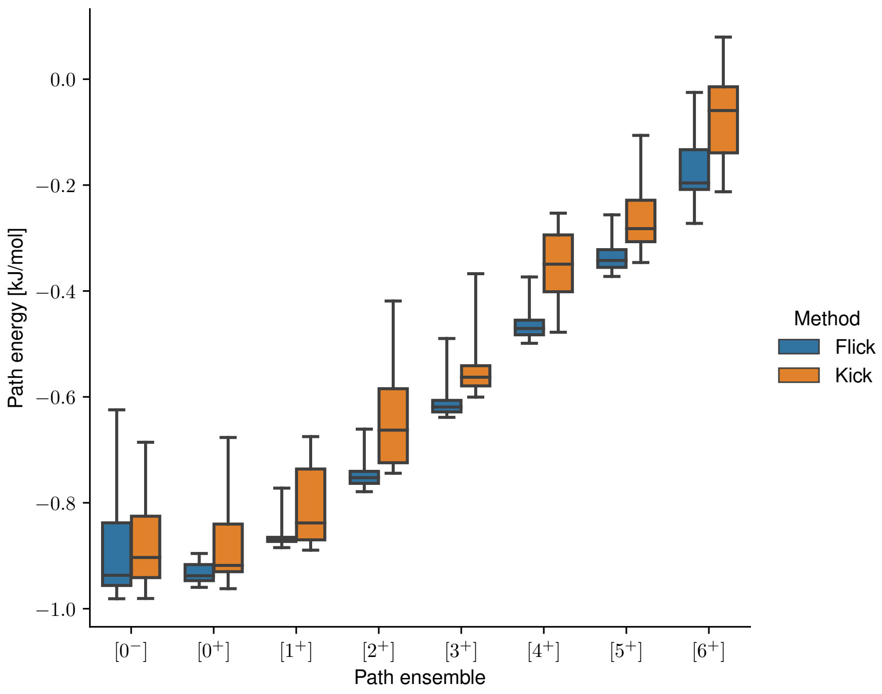


Figure 15: *Top:* Scatterplot of the path energy for 'kick' vs 'flick' on the single-particle 1D double-well potential system. Fifteen data points are used for both method. A random amount of jitter in the categorical axis is applied. The input files are the same as in 'Example 1,' but with different `Initial-path` sections. The parameters in 'flick' are $\alpha = 0.8$, $n_{\text{del}} = 1$, and $n_{\text{reac}} = 1$. The order parameter of the initial configuration point is $\lambda(x_0) = -0.75$. *Bottom:* Box-and-whisker plot of the above data. The line inside a box indicates the median value, whereas the top and bottom lines of a box indicate the median of the upper and lower half of the values, respectively. Here, the whiskers extend from the minimum to the maximum value.

4.2 NaCl in Water

The new initialization method has also been tested on a NaCl system (Section 2.10) consisting of one Na^+ ion and one Cl^- ion immersed in a bath of $N_w = 908$ water molecules using an OPLS-AA force field [28]. Gromacs was used as an engine (Section 2.9) and the ion-ion distance $r_{ion} = |\mathbf{r}_{Na^+} - \mathbf{r}_{Cl^-}|$ in nanometers is used as order parameter. The 'NaCl in Water' PyRETIS input file below and additional Gromacs input files have been used in all the simulations, but with some different values in the `Initial-path` section and in the Gromacs files. All the simulations samples the NVE ensemble with $V = (30\ \text{Å})^3$, except in Section 4.2.1, in which NVT is sampled with $T = 300$ K using velocity rescaling (keyword `v-rescale` in the `.mdp` file).

4.2.1 Obtaining an Initial Point in Configuration Space

There are many ways to acquire a valid and sensible initial configuration point $\mathbf{R}_0$ to use with the 'flick' method to ensure $\lambda_A < \lambda(\mathbf{R}_0) < \lambda_B$. For instance, a straightforward MD run from stable state A using the temperature one wants to study should yield several crossings with λ_A, given that λ_A is placed reasonably.

Figure 16 shows the energy and order parameter of an MD run sampling the NVT ensemble with $T = 300$ K and $V = (30\ \text{Å})^3$ from some low energy configuration. The simulation was executed with Gromacs, with a time step of $\delta t = 2$ fs and integration performed using the velocity Verlet algorithm. After about 85 ps, the order parameter is between the stable state conditions that is going to be used in the 'NaCl in Water' example, i.e. $0.32 < \lambda(\mathbf{R}_0) < 0.70$. The point corresponding to $t = 85$ ps from this trajectory is thus a valid initial configuration point for the 'flick' method. The energy plots show that the energy has converged at this time. Also worth noting is that the salt dissociates (goes from stable state A to stable state B) by itself in this run, suggesting that this event might not be so rare after all.

4.2.2 Errors and Troubleshooting

Not everything has run smoothly while working on this thesis. This section will elaborate on a few of the debugging quest's time-consuming efforts, successes, and workarounds.

One problem emerged while running TIS/RETIS simulations using the `gromacs2` engine on my personal computer. An inexplicable error occurred during simulations that did not occur on a different computer with the exact same configurations and settings. The only difference was the brands of the machines. One was a Linux machine with an Intel chip, while I used a MacBook Pro with an Apple Silicon M1 chip. A hypothesis is that the `gromacs2` engine in PyRETIS was unable to kill the correct processes due to some specifics of the M1 chip's efficiency and performance cores. However, the error did not occur during

Input file: NaCl in Water

```
NaCl in Water
=============

Simulation
----------
task = retis
steps = 0
interfaces = [0.32, 0.34, 0.36,
              0.38, 0.41, 0.70]

System
------
units = gromacs

Box
---
cell = [3, 3, 3]
periodic = [True, True, True]

Engine
------
class = gromacs2
gmx = gmx
mdrun = gmx mdrun
input_path = gromacs_input
timestep = 0.002
subcycles = 1
gmx_format = gro
maxwarn = 15

TIS
---
freq = 0.5
maxlength = 100000
aimless = True
allowmaxlength = False
zero_momentum = False
rescale_energy = False
sigma_v = -1
seed = 0

RETIS
-----
swapfreq = 0.5
relative_shoots = None
nullmoves = True
swapsimul = True

Initial-path
------------
method = flick
n_del = 1
reactive_paths = 4
alpha = 0.8

Orderparameter
--------------
class = Distance
index = (0, 1)
periodic = True

Output
------
backup = 'backup'
order-file = 1
energy-file = 1
trajectory-file = 1
screen = 1
```

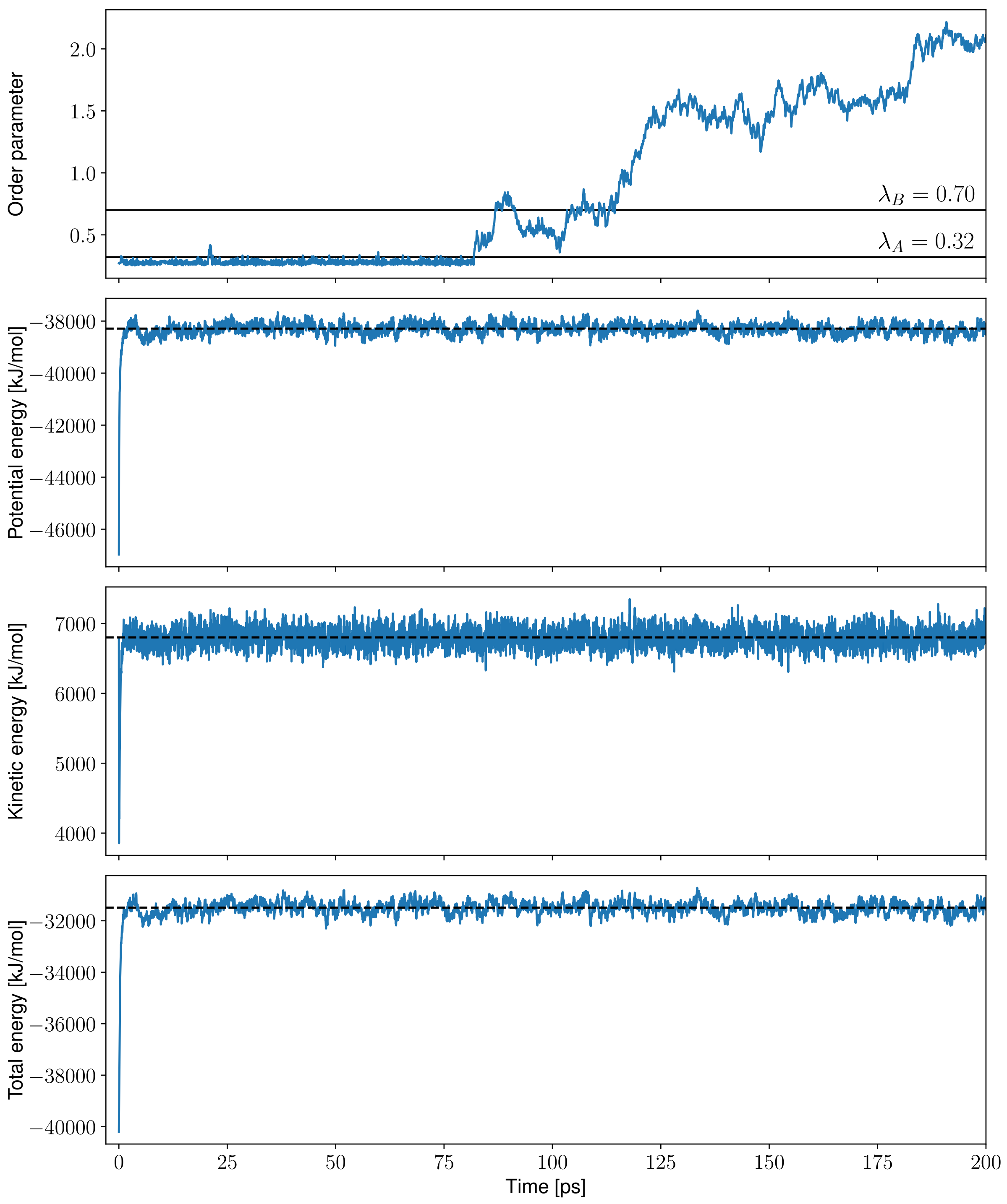


Figure 16: Gromacs MD run for the NaCl system sampling the NVT ensemble with $T =$ 300 K and $V = (30\ \text{Å})^3$. The time step is $\delta t = 2$ fs and the integration is performed using the velocity Verlet algorithm. We see that the system is in equilibrium in the range 25–200 ps. Here, the average total energy is −31 489 kJ/mol, the average kinetic energy −6 800 kJ/mol, and the average potential energy −38 289 kJ/mol (plotted as dashed lines).

the initialization process using 'kick' or 'flick'—only during the TIS/RETIS part. I spent quite some time trying to figure out the bug, but in the end, a workaround was to use one of the Department of Chemistry's computer clusters when performing TIS/RETIS simulations.

Another problem was regarding the integrator chosen in the Gromacs settings (Section 2.9). The problem arose using the leap-frog integration method when performing the shooting move (Section 2.5.2). In Gromacs, leap-frog is the default integrator, selected by the not-so-descriptive keyword `md`, while velocity Verlet requires the keyword `md-vv`. Figures 17 and 18 show a comparison between two paths produced by 'kick': one via leap-frog and the other via velocity Verlet. Otherwise, the initial conditions and settings are identical (except for the random seed).

Figure 17 shows the energy and OP of each step in the kicking part of the 'kick' method (corresponding to the black path in Figure 4). The initial configuration point is the same as the one that was chosen in Section 4.2.1. The OPs are strictly increasing for both integrators, as they should. What is surprising is that the energy is decreasing for the leap-frog integrator. Since the initial point was already in energy equilibrium, and since high-potential moves always will be accepted if they increase the OP—regardless of how unphysical they are—one would expect the energy to increase in the kicking part. Velocity Verlet, on the other hand, seems to increase the energy, as expected.

Figure 18 shows two initial paths produced to the $[3^+]$ ensemble. The last point for each integration scheme in Figure 17 should be identical to the leftmost points in the forward parts in Figure 18 (i.e. equal OPs and energies). Moreover, the second to last point should match the rightmost point in the backward part considering OP and potential energy. The leap-frog path reveals a considerable discontinuity in the total energy in the path connected by the shooting points. Furthermore, it is not expected to see such an extreme change in the energy right by the shooting points. If any, one would expect to see a *decrease* rather than an *increase* in the potential energy near the shooting points, as they are expected to be in a higher potential due to the unphysical nature of the 'kick' method. Figure 18 shows only one simulation for each integration scheme, but one can verify that the jump in energy for the leap-frog integrator is not due to randomness by doing more simulations.

The reason why the leap-frog integrator is not suited for the shooting move is that it uses positions $\mathbf{R}$ at time t and velocities $\mathbf{V}$ at time $t - \frac{1}{2}\delta t$ to update its positions and velocities [25]. On the other hand, the velocity Verlet integrator uses both positions and velocities at time t to integrate the equations of motion. This means that given a starting point $\mathbf{x}(0) = (\mathbf{R}(0), \mathbf{V}(0))$, the leap-frog and the velocity Verlet will *not* produce identical trajectories, as the leap-frog will interpret the velocities as corresponding to $t = -\frac{1}{2}\delta t$, while the velocity Verlet will interpret them as corresponding to $t = 0$. While performing the shooting move, the velocities at $\pm\frac{1}{2}\delta t$ shifted time steps are not defined; only the velocities at $\pm\delta t$ shifted time steps. I do not know what Gromacs does when it finds out that the velocities are not defined—it is presumably drawing new velocities from some distribution. In any case, it is evident that the velocity Verlet algorithm is a more suitable integration scheme than the leap-frog algorithm in TIS/RETIS.

In Sections 4.2.3 and 4.2.4, we compare the path energies of 'kick' and 'flick' using initial configuration points just to the right of the $\lambda_A = 0.32$ nm interface. We also wanted to compare the energies using a configuration point close to the free energy barrier at $\lambda \approx 0.36$ nm (Figure 8). However, in all the simulations, 'kick' was unable to produce initial paths to the $[2^+]$ ensemble ($\lambda_2 = 0.36$ nm). What happened was that the forward MD trajectory went to B in the shooting move and was accepted. The backward trajectory also went to B and, hence, got rejected. In fact, the backward path went to B in all the consecutive trials, eventually leading to a disk overflow error since they did not get deleted. Some trivial bug assumably causes this error in the 'kick' method. Furthermore, there is an improvement of the 'kick' method described in Section 2.6.5 that ensures consecutive trials to progress toward λ_A—maybe this improvement has not yet been implemented? The new method, on the other hand, was indeed able to produce initial paths from a configuration point close to the energy barrier.

4.2.3 Energy Comparison of Initial Paths

Figure 19 shows the OPs and energies of two example initial paths made to the $[3^+]$ ensemble: one made by 'kick' and the other by 'flick.' The box plot in Figure 20 was made using 15 simulations for both methods. The plot clearly shows that the initial paths created by 'flick' are lower than those created by 'kick.' The reason why 'flick' has identical boxes in path ensembles $[2^+]$, $[3^+]$, and $[4^+]$ is that the algorithm found a low-energy path in $[4^+]$ that it reused in the lower-order ensembles—for all the individual simulations—which we can verify by studying the scatterplot.

Figure 21 conveys the same message as Figure 20, but in a different representation: a straightforward MD run of the NaCl system in equilibrium shows the energy fluctuations and mean, alongside the $[3^+]$-ensemble initial paths from 'kick' and 'flick.' A comparison of different values of input parameters for the 'flick' method is shown in Figure 22. The parameter values are chosen somewhat arbitrarily.

For *NVE* dynamics, the energy in a path is constant, whereas this is not true for *NVT*. The energy fluctuations can be analyzed in a system with many particles via the relation between heat capacity and the standard deviation in the total energy. The heat capacity

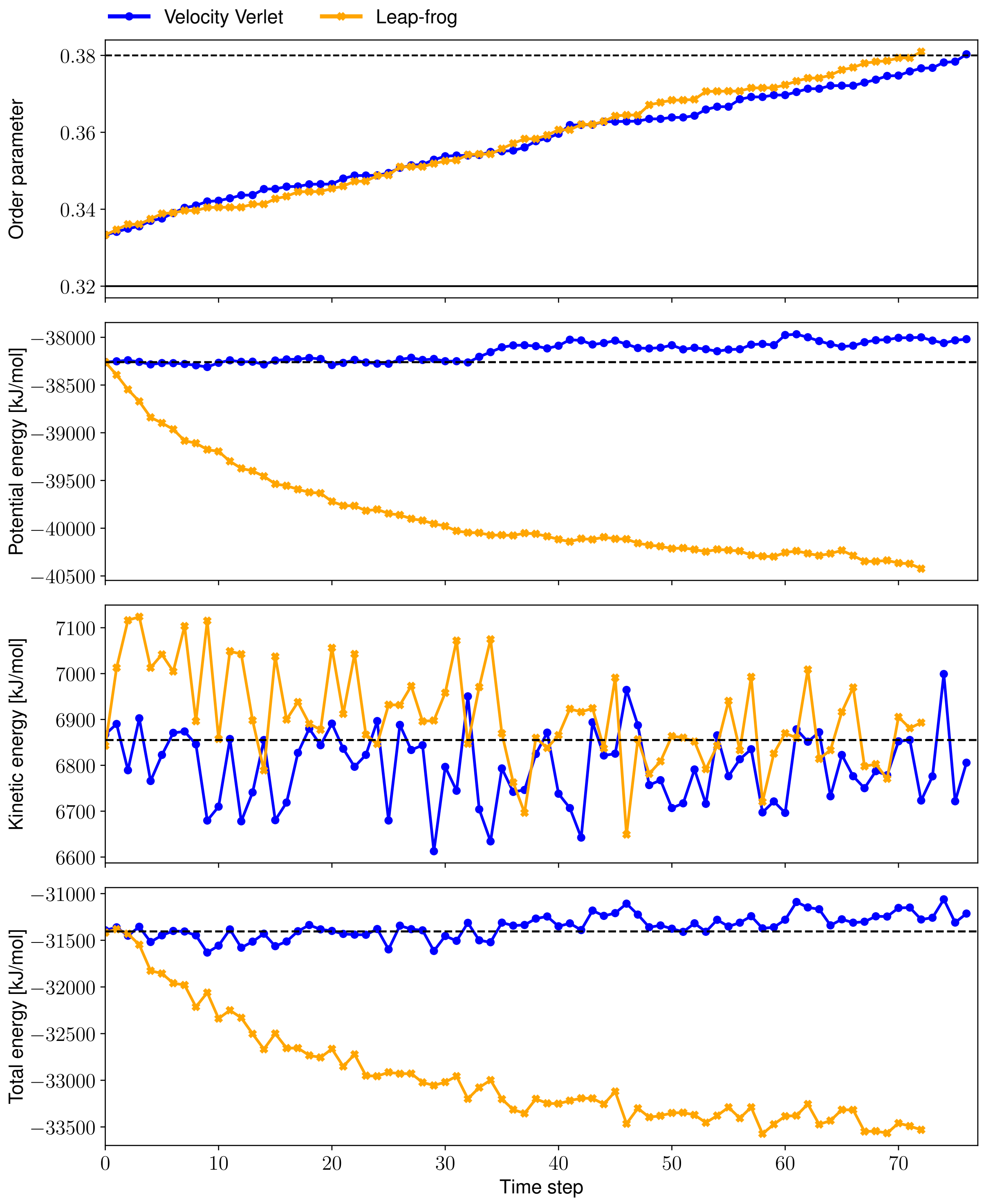


Figure 17: Kicking (corresponding to the black path in Figure 4) in the $[3^+]$ path ensemble ($\lambda_3 = 0.38$) with velocity Verlet and leap-frog. The NVE ensemble is sampled using the 'NaCl in Water' input file, but with `method=kick`. The dashed lines in the energy panels are references to the energy in step 0.

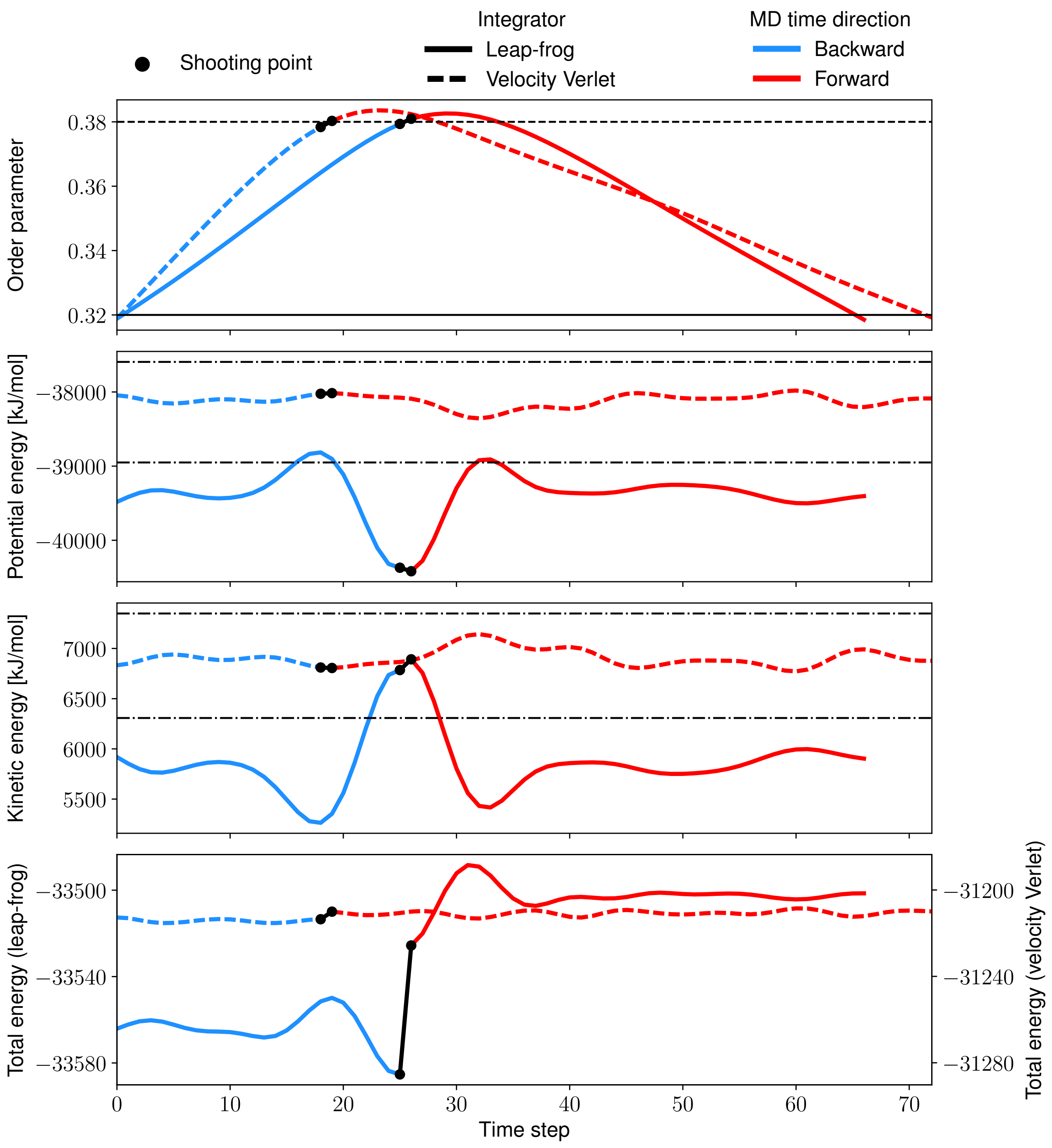


Figure 18: Comparison of an initial path to the $[3^+]$ path ensemble ($\lambda_3 = 0.38$) produced by the 'kick' method using the leap-frog algorithm (solid curve) and the velocity Verlet algorithm (dashed curve). The dynamics is run in the *NVE* ensemble, the order parameter is the ion-ion distance in nanometers, and the input parameters are the same as the 'NaCl in Water' input file but with `method=kick`. The two black dots on each curve are the two shooting points produced in the kicking part of the 'kick' method (Figure 4): they correspond to the last two steps in Figure 17. The dash-dotted lines in the middle panels indicate the minimum and maximum energy observed for the system in equilibrium (see Figure 21).

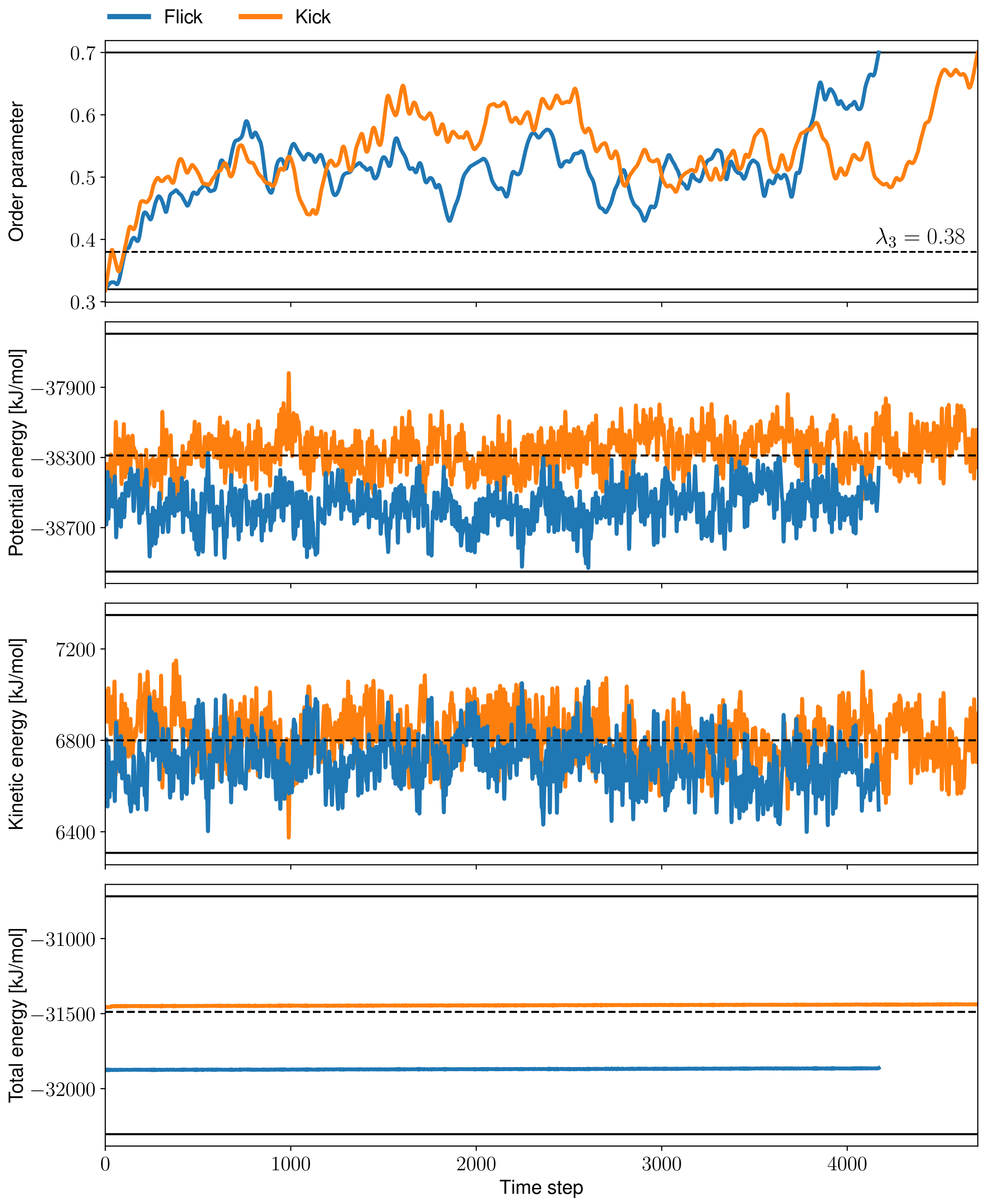
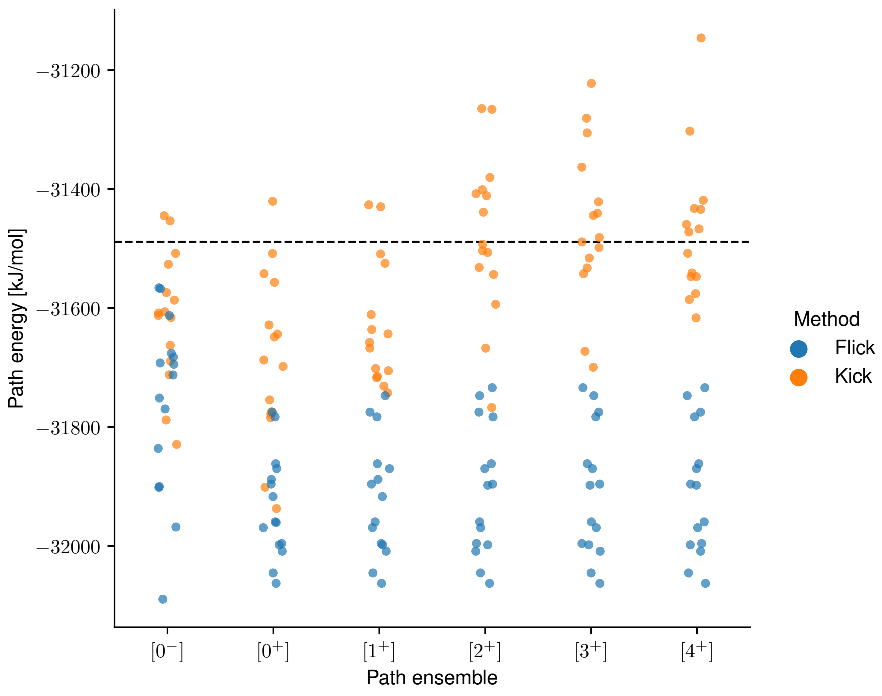


Figure 19: A reactive path was made to the $[3^+]$ path ensemble ($\lambda_3 = 0.38$) by 'flick' and 'kick' during an initialization. The *NVE* ensemble is sampled. The order parameter of the initial configuration point is $\lambda(\mathbf{R}_0) \approx 0.327$. Here, the 'flick' input parameters are $\alpha = 0.8$, $n_{\text{del}} = 1$, and $n_{\text{reac}} = 4$. The solid black lines in the energy panels indicate the maximum and minimum observed energy for the system in equilibrium; the black dashed lines indicate the average energy (see Figure 21).



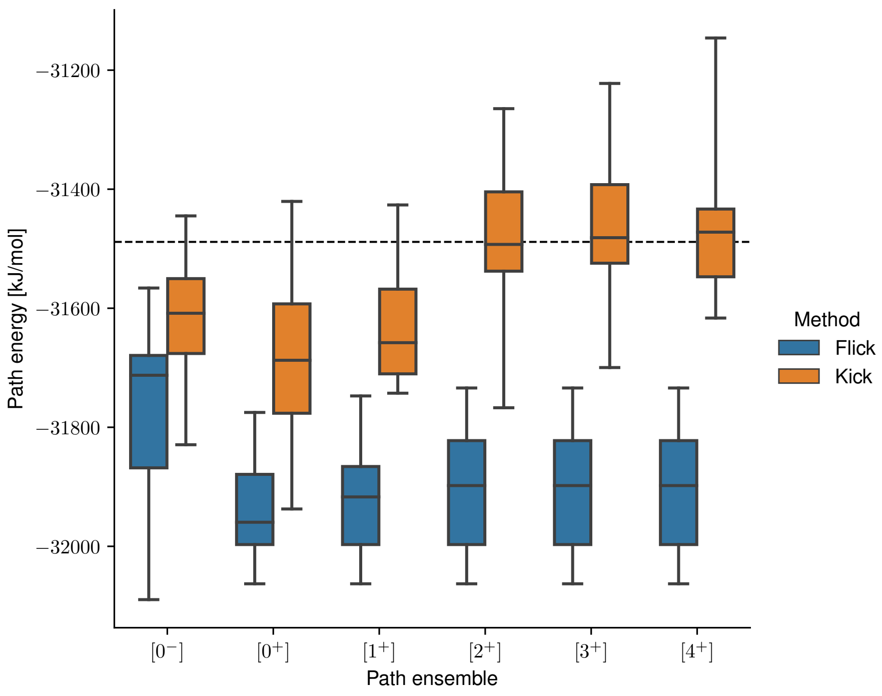


Figure 20: *Top:* Scatterplot of the path energy of 'kick' vs. 'flick' for the NaCl-in-water system in the NVE ensemble. Fifteen data points are used for both methods. A random amount of jitter in the categorical axis is applied. The inputs are the same as in the 'NaCl in Water' input file. The order parameter of the initial configuration point is $\lambda(\mathbf{R}_0) \approx 0.327$. Here, the 'flick' input parameters are $\alpha = 0.8$, $n_{\mathrm{del}} = 1$, and $n_{\mathrm{reac}} = 4$. The dashed line indicate the average total energy of the system in equilibrium (Figure 16). *Bottom:* Box-and-whisker plot of the above data point. The line inside a box indicates the median value, whereas the top and bottom lines of a box indicate the median of the upper and lower half of the values, respectively. Here, the whiskers extend from the minimum to the maximum value.

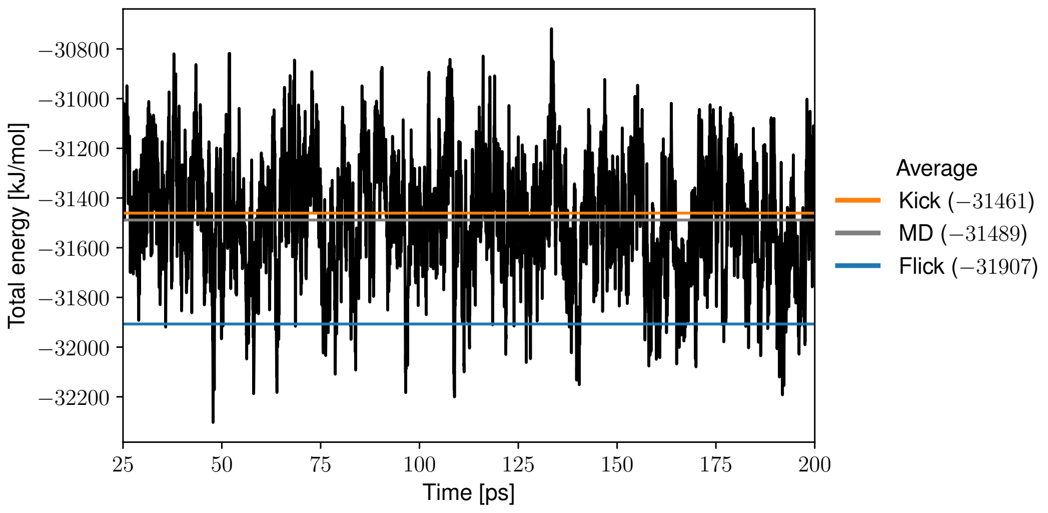


Figure 21: Equilibrium part of the MD run in Figure 16. The averages of 'kick' and 'flick' from the $[3^+]$ ensemble in the scatterplot in Figure 20 are also shown. The minimum and maximum value the energy takes within this 175 ps window are $-32\,303$ kJ/mol and $-30\,718$ kJ/mol.

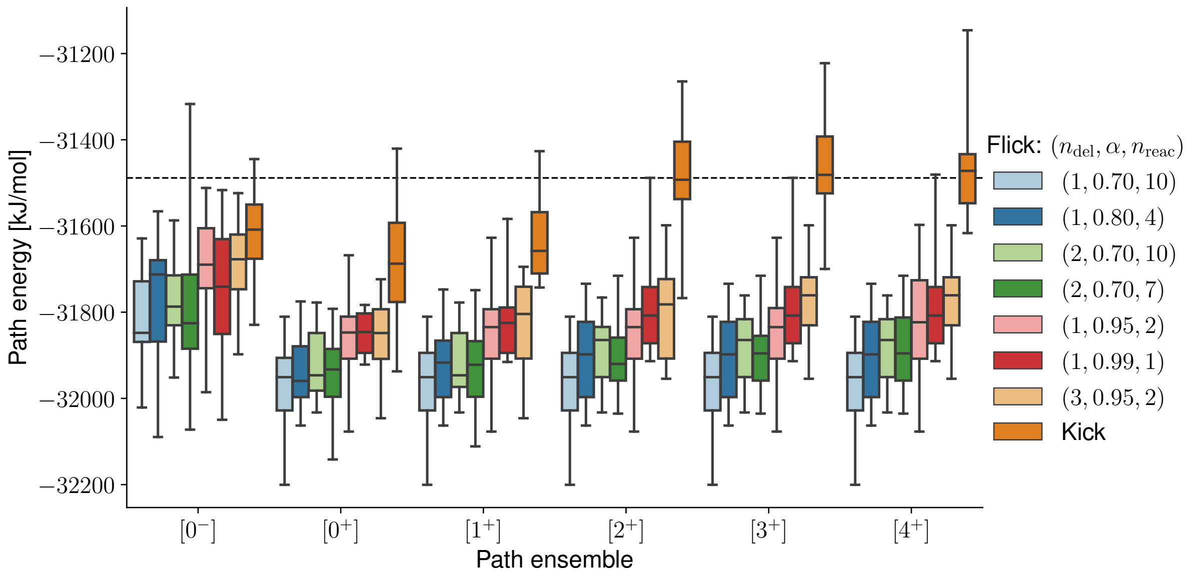


Figure 22: Box-and-whisker plot of the path energy for 'flick' with different values of α, n_{del}, and n_{reac}. Otherwise, the input files are equal to the 'NaCl in Water' input file. 'Kick' from Figure 20 is shown as a reference. Fifteen simulations are used for each input file. The horizontal dashed line indicates the average total energy of the system in equilibrium (Figure 21). The line inside a box indicates the median value, whereas the top and bottom lines of a box indicate the median of the upper and lower half of the values, respectively. Here, the whiskers extend from the minimum to the maximum value.

$C_V = \langle \partial E / \partial T \rangle$ can be written as [16]

$$C_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{k_{\mathrm{B}} T^2} = \frac{\sigma_E^2}{k_{\mathrm{B}} T^2}, \tag{29}$$

where σ_E^2 is the variance. The standard deviation can thus be written as

$$\sigma_E = T\sqrt{k_{\mathrm{B}} C_V} = T\sqrt{k_{\mathrm{B}} C_{V,m} n} = T\sqrt{k_{\mathrm{B}} C_{V,m} N / N_{\mathrm{A}}}, \tag{30}$$

where $n = N/N_{\mathrm{A}}$ is the number of moles, N the number of molecules, and $C_{V,m}$ the molar heat capacity. To express the standard deviation $\sigma_E = \sqrt{\langle E^2 \rangle - \langle E \rangle^2}$ in J/mol, we multiply by Avogadro's number, yielding

$$\sigma_E^* = \sigma_E N_{\mathrm{A}} = T\sqrt{k_{\mathrm{B}} C_{V,m} N_{\mathrm{A}} N} = T\sqrt{R N C_{V,m}}, \tag{31}$$

where $R = k_{\mathrm{B}} N_{\mathrm{A}}$ is the gas constant. There are $N_w = 908$ water molecules, the heat capacity of water is 75.38 J/(K mol) $\approx 9R$, and $T = 300$ K. Therefore,

$$\sigma_E^* \approx 3RT\sqrt{908} \approx 225 \text{ kJ/mol}. \tag{32}$$

This shows that energy fluctuations up to 225 kJ/mol happen frequently. The solvent is the main cause of the high energy states. Since there are so many particles, there will often be some particles that overlap, thereby causing the energy to fluctuate.

4.2.4 Energy Comparison of TIS Simulations

In Figure 23, we see two TIS simulations executing 1000 cycles in the $[3^+]$ path ensemble, comparing 'flick' and 'kick.' Like in a Markov chain, only the accepted cycles are shown. In the top panel, the path energy is shown for each cycle; in the bottom panel, the full trajectories of all the cycles are pasted next to each other. We see that the energy difference between the two methods is large in the beginning (as in Figure 20) and that both methods eventually start fluctuating around the system-equilibrium energy.

Figure 24 shows the average of several TIS simulations executing 300 cycles in the $[3^+]$ path ensemble. The figure compares how the path energy of the accepted paths propagates from the initial path. We see that the energy at step 0 is like in Figures 21: 'kick' is in the system-equilibrium zone, whereas 'flick' is substantially lower. The propagation of the new method suggests that after 100 cycles, the path energy of new accepted paths has regressed to a fluctuation about the mean, keeping in mind the standard deviation in Equation (32).

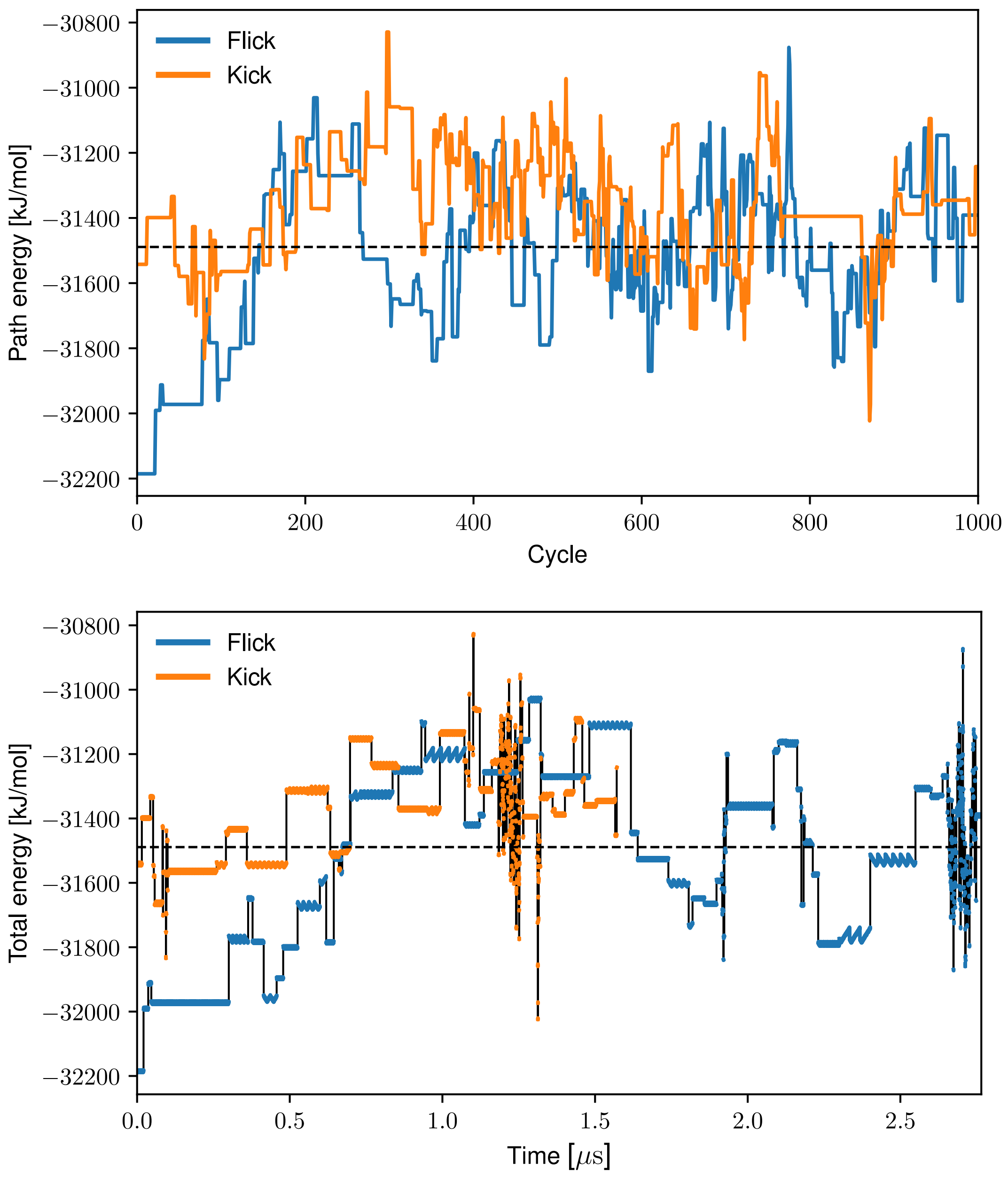


Figure 23: Two TIS simulations in the $[3^+]$ path ensemble ($\lambda_3 = 0.38$) with 1000 cycles resulted in 245 accepted cycles for the simulation that used 'kick' and 256 for the one that used 'flick.' Here, the initial configuration point had an order parameter $\lambda(\mathbf{R}_0) \approx 0.327$, and the 'flick' parameters were $\alpha = 0.7$, $n_{\text{del}} = 2$, and $n_{\text{reac}} = 7$. The dashed horizontal lines indicate the average total energy of the system in equilibrium (Figure 21). *Top:* The path energy (Equation (27)) of each cycle in the Markov chain (i.e., if a cycle is rejected, the previous cycle is kept and recounted, thereby causing the horizontal segments in the plots). *Bottom:* The full trajectories of all the cycles are pasted next to each other. The vertical black lines show the link between the accepted cycles.

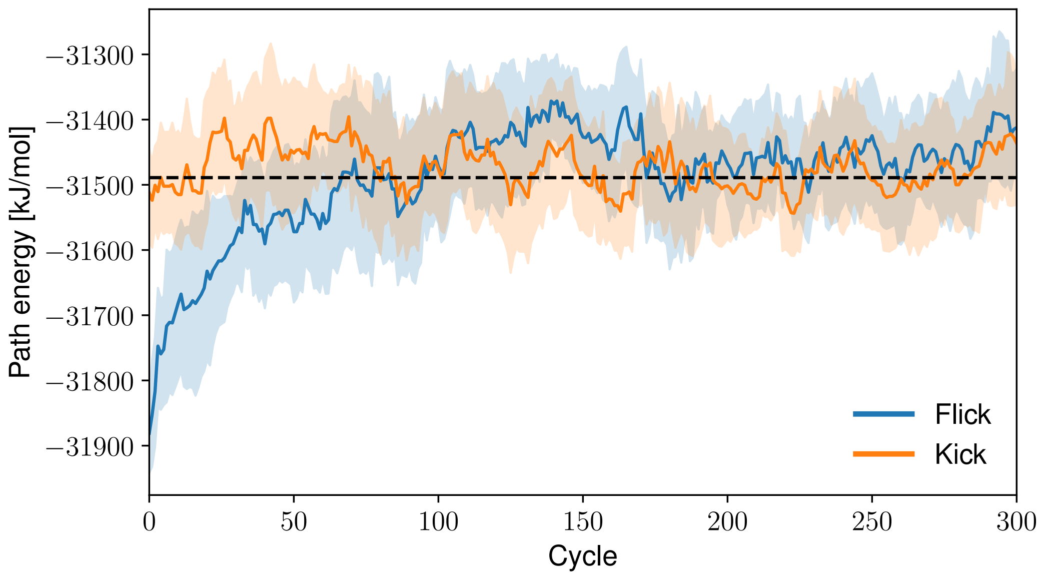


Figure 24: Fifteen TIS simulations with 300 cycles in the $[3^+]$ path ensemble ($\lambda_3 = 0.38$) with initial paths made by 'flick' and fifteen made by 'kick.' The plot shows the average path energy for each cycle in the Markov chains (i.e., the average of 15 plots similar to the top panel in Figure 23). The initial configuration point had an order parameter $\lambda(\mathbf{R}_0) \approx 0.327$ and the 'flick' input parameters were $\alpha = 0.7$, $n_{\mathrm{del}} = 2$, and $n_{\mathrm{reac}} = 7$. The solid lines show the mean and the bands show the 95 % confidence interval (automatically generated by the plotting software [29, 30]). The dashed horizontal line indicates the average total energy of the system in equilibrium (Figure 21).

5 Conclusion and Further Work

As stated in Section 2.3, the first, say 5–10 %, of the steps of a molecular dynamics or Monte Carlo method, is usually ignored when averaging because they are generally not representative of the desired equilibrium properties. In path sampling, this issue can be even more delicate. Since generating new paths is computationally costly, minimizing the number of paths that will later be ignored is essential. Besides, finding a suitable initial trajectory for each path ensemble is far more complicated than finding an initial configuration in MD and MC.

Since integration is done forward and backward in time in the shooting move, it is important to use a time-symmetric integration scheme. Specifically, we have studied the leap-frog (time-asymmetric) and the velocity Verlet (time-symmetric) and concluded that the former leads to strange behavior and discontinuities, whereas the latter produces sensible trajectories.

The 'kick' procedure makes physical trajectories (since they obey Newton's equations of motion) and finds the initial trajectories very quickly. However, it generally leads to highly unlikely trajectories and can even result in a permanent sampling trap where it is impossible to relax to the more relevant region in path space in consecutive MC moves. The 'kick' initial paths did not show the expected rapid energy increase due to forcefully increasing the OP while kicking—even as it would potentially lead to overlapping molecules. The test system (NaCl in water) was perhaps not the best system to illustrate the shortcomings of the 'kick' method.

We see that the new method (dubbed the 'flick' method) produces paths with far lower energy than the 'kick' method—it even produces paths with far lower energy than the NaCl system exhibits in equilibrium. In a way, 'flick' finds trajectories that almost freeze the system. However, this might be different for other systems. It means, nevertheless, that the energy criterion in (26) might be extravagant. The 'kick' approach is known to produce high-energy paths, and we added the energy criterion to the new method to eliminate such paths. Eliminating paths with high energy also seems to make sense based on the Boltzmann weight. On the other hand, entropy also plays a role, which is why water is liquid at room temperature even if a configuration point representing an ice crystal is lower in energy. The removal of the energy criterion will still presumably produce better paths than 'kick,' as no unphysical kicking is involved.

When the user provides the interfaces $\{\lambda_1, \lambda_2, ..., \lambda_{n-1}\}$, the path chosen as the initial path by the 'flick' method to each ensemble is the one with the lowest path energy that is a valid path for the ensemble in question. A task for further work is: Can the information of the sampled paths in the initialization procedure be used to suggest the 'best' interfaces?

The algorithm presented in Chapter 3 starts by assuming that λ_A and λ_B are given, and so do most theoretical 'rules of thumb.' However, it is known that the placement of λ_A greatly influences the efficiency since shifting λ_A to the right decreases the average path length in all the ensembles except for the $[0^-]$ ensemble, which is increased. The next phase of the initialization should therefore focus on optimizing the number of interfaces and their placements (including λ_A and λ_B) during a preliminary RETIS run in which all MC moves are executed in addition to on-the-fly adjustments of the interfaces.

The implementation of the new initialization method is virtually finished. Further work will involve the above-mentioned prospects, finishing a small remainder of the implementation of the algorithm, and testing it on additional systems. As mentioned earlier, the dissociation of salt in water is not particularly rare—it is desirable to test the new method on a rarer event. The remaining parts of the algorithm are making it support more than one initial configuration point, making it possible to terminate before the acquired number of reactive paths are found, and handling a potential disk overflow. Moreover, we would like to see running time, error, and efficiency analyses compared to other initialization methods. Also, analysis of how the user-specified parameters (α, n_{del}, and n_{reac}) affects efficiency and performance is of interest. Finally, the declaration of *docstrings* in the source code, proper output to the log file, and so on, in alignment with the PyRETIS project, is required before a possible adoption into the next version of PyRETIS. On a final note, a parallelizable RETIS method was recently developed [23]. The 'flick' algorithm will probably need some adaptions to be compatible with this method.

References

[1] Jan G. Frydenlund. “New Initialization Procedure in RETIS”. TFY4510 – Physics, Specialization Project. Norwegian University of Science and Technology, Jan. 2022.

[2] David E. Shaw et al. “Anton 3: Twenty Microseconds of Molecular Dynamics Simulation Before Lunch”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. St. Louis Missouri: ACM, Nov. 2021, pp. 1–11. ISBN: 978-1-4503-8442-1. DOI: 10.1145/3458817.3487397.

[3] Josep Gelpi et al. “Molecular Dynamics Simulations: Advances and Applications”. In: *Advances and Applications in Bioinformatics and Chemistry* (Nov. 2015), p. 37. ISSN: 1178-6949. DOI: 10.2147/AABC.S70333.

[4] 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”. In: *Advanced Theory and Simulations* 4.4 (Apr. 2021), p. 2000237. ISSN: 2513-0390, 2513-0390. DOI: 10.1002/adts.202000237.

[5] Christoph Dellago, Peter G. Bolhuis, and David Chandler. “Efficient transition path sampling: Application to Lennard-Jones cluster rearrangements”. In: *The Journal of Chemical Physics* 108.22 (June 1998), pp. 9236–9245. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.476378.

[6] Christoph Dellago, Peter G. Bolhuis, and David Chandler. “On the Calculation of Reaction Rate Constants in the Transition Path Ensemble”. In: *The Journal of Chemical Physics* 110.14 (Apr. 1999), pp. 6617–6625. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.478569.

[7] Titus S. van Erp. “Efficiency Analysis of Reaction Rate Calculation Methods Using Analytical Models I: The Two-Dimensional Sharp Barrier”. In: *The Journal of Chemical Physics* 125.17 (Nov. 2006), p. 174106. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.2363996.

[8] Titus S. Van Erp. “Dynamical Rare Event Simulation Techniques for Equilibrium and Non-Equilibrium Systems”. In: *Advances in Chemical Physics*. Ed. by Gregoire Nicolis and Dominique Maes. Hoboken, NJ, USA: John Wiley & Sons, Inc., Apr. 2012, pp. 27–60. ISBN: 978-1-118-30951-3. DOI: 10.1002/9781118309513.ch2.

[9] Titus S. van Erp. “Reaction Rate Calculation by Parallel Path Swapping”. In: *Physical Review Letters* 98.26 (June 2007). ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.98.268301.

[10] *Personal Communication With Titus S. van Erp.*

[11] Anders Lervik, Enrico Riccardi, and Titus S. van Erp. "PyRETIS: A Well-Done, Medium-Sized Python Library for Rare Events". In: *Journal of Computational Chemistry* 38.28 (July 2017), pp. 2439–2451. ISSN: 01928651. DOI: 10.1002/jcc.24900.

[12] Enrico Riccardi et al. "PyRETIS 2: An Improbability Drive for Rare Events". In: *Journal of Computational Chemistry* 41.4 (Nov. 2019), pp. 370–377. ISSN: 0192-8651, 1096-987X. DOI: 10.1002/jcc.26112.

[13] Jan G. Frydenlund. *Git-fork: PyRETIS 'flick' method.* July 2022. URL: https://gitlab.com/jangfr/pyretis/-/tree/MSc_JGF_flick.

[14] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications.* 2nd ed. Computational science series 1. San Diego: Academic Press, 2002. ISBN: 978-0-12-267351-1.

[15] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics.* 3. ed., new internat. ed. Harlow: Pearson, 2014. ISBN: 978-1-292-02655-8.

[16] Andrew R. Leach. *Molecular Modelling: Principles and Applications.* 2nd ed. Harlow, England ; New York: Prentice Hall, 2001. ISBN: 978-0-582-38210-7.

[17] Dominik Marx and Jürg Hutter. *Ab Initio Molecular Dynamics: Basic Theory and Advanced Methods.* Cambridge University Press, Apr. 2009. ISBN: 978-0-511-60963-3. DOI: 10.1017/CBO9780511609633.

[18] Titus S. van Erp, Daniele Moroni, and Peter G. Bolhuis. "A Novel Path Sampling Method for the Calculation of Rate Constants". In: *The Journal of Chemical Physics* 118.17 (May 2003), pp. 7762–7774. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.1562614.

[19] Andrew J. Ballard and Christoph Dellago. "Toward the Mechanism of Ionic Dissociation in Water". In: *The Journal of Physical Chemistry B* 116.45 (Nov. 2012), pp. 13490–13497. ISSN: 1520-6106, 1520-5207. DOI: 10.1021/jp309300b. URL: https://pubs.acs.org/doi/10.1021/jp309300b (visited on 05/06/2022).

[20] Titus S. van Erp. "Solvent Effects on Chemistry with Alcohols. An Ab Initio Study". Ph.D. thesis. Apr. 2003.

[21] Titus S. Van Erp et al. "Prospects of Transition Interface Sampling Simulations for the Theoretical Study of Zeolite Synthesis". In: *Physical Chemistry Chemical Physics* 9.9 (2007), p. 1044. ISSN: 1463-9076, 1463-9084. DOI: 10.1039/b614980d.

[22] Raffaela Cabriolu et al. "Foundations and Latest Advances in Replica Exchange Transition Interface Sampling". In: *The Journal of Chemical Physics* 147.15 (Oct. 2017), p. 152722. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4989844.

[23] Sander Roet, Daniel T. Zhang, and Titus S. van Erp. *Exchanging Replicas With Unequal Cost, Infinitely and Permanently.* arXiv:2205.12663 [cond-mat, physics:physics]. May 2022. DOI: 10.48550/arXiv.2205.12663.

[24] Titus S. van Erp et al. *PyRETIS: Rare events in Python.* 2021. URL: www.pyretis.org (visited on 04/27/2022).

[25] Paul Bauer, Berk Hess, and Erik Lindahl. *GROMACS 2022 Manual.* Feb. 2022. DOI: 10.5281/zenodo.6103568.

[26] Konrad Wilke. "Investigation of the Molecular Mechanism of Sodium Chloride Dissociation in Water With Rare Event Simulations". MA thesis. Technical University Dresden, May 2022.

[27] Charles R. Harris et al. "Array Programming With NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-020-2649-2.

[28] William L. Jorgensen, David S. Maxwell, and Julian Tirado-Rives. "Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids". In: *Journal of the American Chemical Society* 118.45 (Nov. 1996), pp. 11225–11236. ISSN: 0002-7863, 1520-5126. DOI: 10.1021/ja9621760.

[29] Thomas A Caswell et al. *matplotlib/matplotlib: REL: v3.5.1.* Dec. 2021. DOI: 10.5281/zenodo.5773480.

[30] Michael Waskom. "Seaborn: Statistical Data Visualization". In: *Journal of Open Source Software* 6.60 (Apr. 2021), p. 3021. ISSN: 2475-9066. DOI: 10.21105/joss.03021.