Biasing sampling on Networks

Feifan Cai

First Supervisor: Professor Edina Rosta
Second Supervisor: Dr Balint Dudas

A Dissertation Submitted for the Degree of
MSc Scientific and Data Intensive Computing
of
University College London
Department of Physics & Astronomy

August 2024

Declaration

I, Feifan Cai, confirm that the work presented in this dissertation is my own. Where information has been derived from other sources, I confirm that this has been indicated in the dissertation.

Abstract

This dissertation explores advanced methodologies for biasing sampling on networks, with a particular focus on the application of molecular dynamics simulations. The primary challenge addressed is the inherent limitation of traditional MD simulations getting trapped in local potential minima. To tackle this, the study employs enhanced sampling techniques with mean first passage time developed by Professor Edina Rosta's group, which gives better performance compared with other enhanced sampling techniques such as MetaDynamics in 1D, 2D and NaCl cases [1]. Through the implementation of principal component analysis methods, the research projects a 6D collective variables space into a 2D principal component space and successfully push the system into our desired target state.

Keywords— Molecular Dynamics - Mean First Passage Time - Markov State Models

Acknowledgements

I would like to express my deepest gratitude to my primary supervisor, Professor Edina Rosta, for her invaluable guidance and insightful suggestions throughout this project. Her support has been instrumental in shaping the direction of my research. I would also like to extend my sincere thanks to my second supervisor, Dr. Balint Dudas, for his constructive feedback and encouragement. Lastly, I am grateful to Dr. Tiejun Wei for his patient assistance and warm encouragement.

Link to GitHub repository

https://github.com/benzi604/PHAS0077-biasing-sampling-on-networks

Table of Contents

1 Introduction

1.1 Background

Molecular Dynamics (MD) is a powerful computational technique used to simulate the physical movements of atoms and molecules over time. By solving Newton's equations of motion, MD provides detailed insights into the structural and dynamical properties of molecular systems. It has found widespread application in various fields, including biophysics, materials science, and chemistry, where it is used to study phenomena such as protein folding, ligand binding, and phase transitions [2]. The primary advantage of MD lies in its ability to offer atomistic-level detail and temporal evolution of complex systems [3]. However, it is also computationally intensive and often limited by the timescales accessible, which can hinder its ability to capture slow processes or rare events in large systems [4, 5].

To address these limitations, enhanced sampling methods have been developed to accelerate the exploration of free energy landscapes and facilitate the observation of rare events. One of the classic enhanced sampling techniques is Umbrella Sampling, which biases the system to overcome energy barriers by applying a potential that encourages sampling of less accessible states [6]. Despite its effectiveness, Umbrella Sampling requires prior knowledge of the reaction coordinate and can be challenging to implement in high-dimensional systems [7]. Another widely used enhanced sampling method is Metadynamics, which adaptively biases the system by adding history-dependent potentials to the collective variables [8]. This approach allows the system to escape local minima and explore new regions of the free energy landscape without requiring a predefined reaction coordinate. Metadynamics has become a popular choice for studying complex molecular systems due to its flexibility and ability to efficiently explore multiple pathways [9].

Markov State Models (MSMs) offer a complementary approach by providing a framework to describe the underlying molecular kinetics of a system [10]. MSMs divide the conformational space into discrete states and model the transitions between these states as a Markov process. This allows for the prediction of long-timescale behaviors from short MD simulations [11]. While MSMs are powerful tools for capturing the dynamic behavior of molecular systems, they can be limited by the quality of the state decomposition and the assumption that the

dynamics between states are Markovian [12]. Additionally, recent advancements in machine learning have been incorporated into MSMs to enhance state decomposition and improve kinetic modeling [13].

Designing optimal kinetic-based biases is a crucial aspect of effectively sampling rare events. The Weighted Histogram Analysis Method (WHAM) [7] and the Dynamic Histogram Analysis Method (DHAM) [14] are two important techniques used to calculate free energy surfaces from biased simulations. WHAM combines histograms from different biased simulations to estimate the free energy surface, but it assumes equilibrium distributions. On the other hand, DHAM extends this approach by considering non-equilibrium dynamics, allowing for the analysis of systems where equilibrium may not be easily achieved. While WHAM is well-established and widely used, DHAM offers a more flexible approach but can be more complex to implement.

A recent advancement by Professor Edina Rosta's group has introduced a novel algorithm that utilizes the Mean First Passage Time (MFPT) as a measure to optimize the biasing potential [1]. This method leverages the MFPT to identify the most kinetically relevant pathways and enhance sampling along these directions. By focusing on optimizing the bias with respect to MFPT, this approach aims to improve the efficiency of sampling rare events and provide more accurate insights into the kinetics of molecular systems.

1.2 Structure

In Chapter 2, the fundamental concepts of the MSM, MFPT, Kemeny Constant, and Principal Component Analysis (PCA) are introduced. Chapter 3 details the algorithms for the DHAM, MFPT calculation, and PCA transformation. Chapter 4 presents the results of 1D and 2D Langevin simulations using the MFPT-based enhanced sampling method, as well as the projection of a 6D Collective Variable (CV) space onto a 2D Principal Component (PC) space through PCA transformation. Finally, Chapter 5 offers a conclusion of this work, along with a discussion of potential future research directions.

2 Literature Review

2.1 Markov State Model

MSMs are mathematical models used to describe systems that undergo transitions from one state to another. System dynamics can be analysed by it as a Markov chain whose transitions depend only on the current state and not on the prior chain of events, which is known as Markov property. Unlike traditional molecular dynamics simulations, which track the continuous evolution of a system over time, MSM decomposes complex processes into discrete states and transitions, allowing for the efficient analysis of long-timescale phenomena [15].

MSM model has been widely used to model the folding pathways of proteins, offering insights into the molecular kinetics and thermodynamics of these processes [5]. Protein folding is inherently complex, with multiple intermediate states and pathways. MSM allow these states to be discretized, with transition probabilities representing the likelihood of moving from one state to another. Beyond protein folding, MSM have also been applied to study ligand binding, conformational changes in large biomolecules, and enzyme catalysis. These models have provided a deeper understanding of the timescales and mechanisms involved in these processes, which are often difficult to observe directly in experiments [4].

Consider a kinetic network consisting of n states. The evolution of the probability distribution from the initial time $t = 0$ to a new distribution after a lag time t can be described as follows [16]:

$$\mathbf{p}(t) = e^{\mathbf{K}t}\mathbf{p}(0)$$

where K is the rate matrix containing the transition probabilities per unit time and the corresponding Markov matrix is defined as $\mathbf{M} = e^{\mathbf{K}t}$.

2.2 Mean First Passage Time

The MFPT is a key concept in the study of stochastic processes, particularly in the context of molecular dynamics and chemical kinetics. MFPT is defined as the expected mean average time it takes for a system to reach a specified state for the first time. This metric is particularly useful in understanding the dynamics of systems where rare events or transitions between states play a critical role, such as in protein folding, chemical reactions, or diffusion processes [17]. The concept of MFPT is extensively applied in molecular dynamics and computational chemistry to quantify the kinetics of transitions between molecular states. For example, in protein folding studies, the MFPT can be used to determine the expected time for a protein to fold from an unfolded state to its native conformation. This provides crucial insights into the folding kinetics and stability of proteins, which are essential for understanding their biological function [18]. MFPT is also employed in studies of ligand binding, where it can describe the expected time for a ligand molecule to find and bind to its target site on a protein. Such calculations are vital for drug design, as they help in assessing the efficacy and binding kinetics of potential drug candidates [19].

The MFPT provides a clear and interpretable measure of the kinetics of stochastic processes. It is particularly advantageous in systems where direct observation of transitions is difficult, either due to the rarity of the event or the complexity of the system. By focusing on the expected time to reach a specific state, MFPT simplifies the analysis of kinetic pathways and allows for the quantification of transition timescales without requiring exhaustive sampling of all possible pathways [20]. Despite its utility, the calculation of MFPT can be computationally demanding, especially in systems with a large number of states or when dealing with continuous-time processes and dimensionality reduction technique can help to mitigate the impact of this issue which will be introduced later. Additionally, for highly complex systems, the linear equation systems required to calculate MFPT can become challenging to solve [21].

Below shows the formula for calculating the mean average time that a system from state i to state j for the first time [1]:

$$t_{ji} = \frac{1}{\mathbf{p}_j^{eq}}[(\mathbf{p}^{eq}\mathbf{1}_n^T - \mathbf{K})_{jj}^{-1} - (\mathbf{p}^{eq}\mathbf{1}_n^T - \mathbf{K})_{ji}^{-1}]$$

where $\mathbf{p}^{eq}$ is the equilibrium probability or stationary distribution and $\mathbf{1}_n^T$ is simply a ones vector of length n.

2.3 Kemeny Constant

The Kemeny Constant is an important metric in the analysis of Markov chains, particularly in understanding the long-term behavior of stochastic processes. Named after the mathematician

John G. Kemeny [22], this constant represents the expected number of steps required to reach a randomly chosen state from a given initial state, averaged over all possible target states with the stationary distribution as the weighting factor. The Kemeny Constant has found significant application in various fields, including network analysis, search algorithms, and ergodic theory. In network analysis, the Kemeny Constant is used to measure the efficiency of random walks on networks. A lower Kemeny Constant indicates that, on average, it takes fewer steps to reach a randomly chosen node, which is often desirable in the design of efficient communication or transportation networks [23]. Additionally, the Kemeny Constant is particularly useful because it is independent of the starting state i, connecting a weighted sum of MFPTs starting from the specific state i to a sum over relaxation timescales [16].

$$t_{ji} = \frac{1}{p_j^{eq}} \sum_{\ell>1} \frac{1}{|\lambda_\ell|} \psi_j^{(\ell)} (\phi_j^{(\ell)} - \phi_i^{(\ell)}),$$

$$\sum_j p_j^{eq} t_{ji} = \sum_j \sum_{\ell>1} \frac{1}{|\lambda_\ell|} \psi_j^{(\ell)} (\phi_j^{(\ell)} - \phi_i^{(\ell)}) = \sum_{\ell>1} \frac{1}{|\lambda_\ell|} = \sum_{\ell>1} \tau_\ell \equiv \zeta.$$

$$t'_{ji} = \frac{1}{p_j^{eq}} [1 + \sum_{\ell>1} \frac{\lambda'_\ell}{1-\lambda'_\ell} \psi_j^{(\ell)} (\phi_j^{(\ell)} - \phi_i^{(\ell)})],$$

$$\sum_j p_j^{eq} t'_{ji} - N = \sum_{\ell>1} \frac{\lambda'_\ell}{1-\lambda'_\ell} (1 - \delta_{\ell,1}) = \sum_{\ell>1} (\frac{1}{1-\lambda'_\ell} - 1),$$

$$\sum_j p_j^{eq} t'_{ji} = 1 + \sum_{\ell>1} \frac{1}{1-\lambda'_\ell}.$$

The formulas for calculating the Kemeny Constant, as outlined in [16], are shown above and λ_ℓ is the eigenvalues of $\mathbf{K}$ in descending order and ψ^ℓ and ϕ^ℓ are the left and right eigenvectors of $\mathbf{K}$ respectively. In this study, the Kemeny Constant is verified by setting a threshold ε, such as 1×10^{-6} and comparing it to the difference between the maximum and minimum values of the Kemeny Constant array after the MFPT is computed.

2.4 PCA Transformation

PCA is a widely used dimensionality reduction technique in data analysis and machine learning. This transformation simplifies the complexity of high-dimensional data while preserving as much variability as possible, making PCA a powerful tool for data visualization, noise reduction, and feature extraction [24]. PCA is extensively applied in various fields, including finance, biology, and image processing, due to its ability to reduce data dimensionality while

retaining the most critical information. In computational biology, PCA is often used in the analysis of gene expression data. For example, PCA can be applied to high-dimensional gene expression datasets to identify the main components that explain variations in gene expression across different conditions or time points. This allows researchers to focus on a reduced set of components that capture the most significant biological signals [25].

The main benefit of PCA lies in its capability to reduce the dimensionality of large datasets while preserving most of the essential information. By focusing on the directions of maximum variance, PCA efficiently compresses data, making it easier to analyze and visualize. Additionally, PCA is also relatively simple to implement and computationally efficient, especially when compared to more complex dimensionality reduction techniques like kernel PCA. This makes it a preferred choice for exploratory data analysis and as a preprocessing step in machine learning pipelines [26]. Despite its widespread use, PCA has several limitations. One major limitation is that PCA is a linear method, meaning that it can only capture linear relationships between variables. it can fall short to capture important patterns when the input datasets is too complex and have non-linear structures [27]. Moreover, the difficulty of interpreting the principal components and sensitivity to variable scaling can lead to misleading results [24].

3 Methodology

In this chapter, several important algorithms will be introduced.

3.1 Dynamic Histogram Analysis Method

The DHAM is an innovative algorithm used to calculate the free energy surfaces of complex systems from molecular dynamics simulations. With the use of multiple histograms that are dynamically updated, DHAM allows it to accurately capture the free energy landscape over a range of conditions and configurations. Additionally, the DHAM algorithm addresses issues that arise in the WHAM where partial convergence in certain umbrella sampling windows leads to substantial errors on the free energy surface. [14].

In this study, we employed the DHAM algorithm to estimate the unbiased Markov matrix from biased simulation trajectories, denoted as $X^{\alpha}(t)$, under previously applied biasing potentials $U^{\alpha}_{\text{bias}}(\xi)$. Before the DHAM operation, we need to firstly define a space of collective variables, $\xi(x)$, which is subsequently discretized and ravelled into a 1D array, consisting of $n = b^d$ states, where b is the number of bins and d is the dimensionality of the collective variable space.

Initially, an $n \times n$ transition count matrix, $M_{unbiased}$, is set to zero. For each trajectory transition observed from state s_i to s_j, the matrix $M_{unbiased}$ is incremented accordingly. Following this, each row in $M_{unbiased}$ is normalized to ensure that the total transition probability remains unity, thereby maintaining the Markov property. Subsequently, for each pair of states (i, j), the method recalculates the unbiased transition probabilities considering the total effect of the previously applied biases over a given lag time $\tau = 1$. This recalibration involves adjusting the transitions in $M_{unbiased}$ based on the comprehensive bias previously applied to the system.

An optional verification step involves assessing the consistency of the Kemeny constant by analyzing the MFPT and calculating the equilibrium probabilities p^{eq}. The final output of DHAM, the matrix $M_{unbiased}$, provides an estimate of the unbiased transitions between states, essential for understanding the thermodynamic and kinetic properties of the system under study. This methodological approach ensures that significant errors in the free energy surface, potentially caused by insufficient convergence in parts of the umbrella sampling windows, are effectively addressed, thereby enhancing the accuracy of the simulation results.

Algorithm 1 Dynamic Histogram Analysis Method

Require: CV space $\xi(x)$ and gaussian parameters $G(A,\mu,\sigma)$ or $G(A,\mu_{cartesian},\Xi_{cartesian})$

procedure DHAM
    $\xi(x)_{digitised} \leftarrow Digitize(\xi(x),X(b))$ ▷ Digitize the space with boundary X and number of bins b
    $\xi(x)_{ravelled} \leftarrow Ravel(\xi(x)_{digitised},(b,b))$ ▷ Ravel the digitised CV space
    Define a $n \times n$ zero matrix $M_{unbiased}$ where $n = b^d$
    for $i = 0$ to $Numiter - 1$ **do**
        for each trajectory **do**
            Increment $M_{unbiased}$ by 1
            Calculate the bias
        end for
    end for
    for $k = 0$ to $n - 1$ **do**
        if $Rowsum(M_{unbiased}) > 0$ **then**
            $M_{unbiased}[k,:] \leftarrow \frac{M_{unbiased}[k,:]}{Rowsum(M_{unbiased})}$
        else
            $M_{unbiased}[k,:] = 0$
        end if ▷ Normalize the $M_{unbiased}$
    end for
    Compute p^{eq} and $free\_energy$
    return $M_{unbiased}$, $free\_energy$
end procedure

3.2 MFPT Calculation

3.2.1 Diagonal Method

In the implementation of the MFPT calculation, the algorithm commences by determining the number of states, $N = b^d$, within the transition matrix M, where b is the number of bins and d is the dimensionality. An identity matrix of size $N \times N$, I, alongside a vector of ones of length N, *ones*, are prepared to facilitate subsequent matrix operations. Matrix A is constructed via the outer product of the stationary distribution p^{eq} and the transpose of *ones*, followed by its transposition to align with M's dimensions. This setup enables the adjustment of M by adding A and subtracting I from it, with the subsequent step involving the calculation of the inverse of this adjusted matrix, denoted as Q_{inv}.

The MFPT matrix, initialized as a zero matrix of dimensions $N \times N$, is populated through a nested iteration over each state pair (i, j). Here, a term is computed using elements from Q_{inv} and I, which reflects the modified impact of transitions from state i to state j. If the product of $p^{eq}[j]$ and the computed term is equal to zero, suggesting of an impracticable transition under equilibrium conditions, a substantially large value, for example 1×10^{12}, should be assigned to MFPT, representing an infinite MFPT and denoting an unreachable state. Conversely, if feasible, the MFPT for the transition is quantified as the inverse of the product of $p^{\mathrm{eq}}[j]$ and the term, thereby measuring the expected time to first reach state j from state i under unbiased conditions. This algorithm furnishes a comprehensive framework for the evaluation of MFPT across a network of states, addressing the intricacies of transitional probabilities and stationary distributions. Although it offers critical insights into the dynamics and stability of the system , the Algorithm 2 has some limitations which will be talked in next section.

Algorithm 2 MFPT Diagonal method

Require: The equilibrium probability distribution p^{eq} and the transition matrix M

procedure MFPT DIAG METHOD
    Define $N \leftarrow b^d$ where b is the number of bins and d is the dimensionality
    Define a ones vector of size N, $ones \leftarrow Ones(N)$
    Define an Identity matrix of size $N \times N$, I
    $A \leftarrow Transpose(p^{eq} \otimes Transpose(ones))$ ▷ The A is the transpose of the matrix product of p^{eq} and transpose of $ones$
    $Q_{inv} \leftarrow Inv(A + I - M)$
    Define a zero matrix of size $N \times N$ for $MFPT$
    for $i = 0$ to $N - 1$ **do**
        for $j = 0$ to $N - 1$ **do** $term1 \leftarrow Q_{inv}[i,i] - Q_{inv}[j,i] + I[j,i]$
            if $p^{eq} \times term1 == 0$ **then**
                $MFPT[j,i] \leftarrow LargeNumber$
            else
                $MFPT[j,i] \leftarrow \frac{1}{term1 \times p^{eq}[i]}$
            end if
        end for
    end for
    $MFPT \leftarrow \frac{MFPT}{time\_step}$
    return $MFPT$
end procedure

3.2.2 JJ-Hunter Method

In MFPT diagonal method, eigenvalue and eigenvector decomposition is used to calculate the stationary distribution p^{eq} which can lead to obtaining negative MFPT in some cases. JJ-Hunter Method can give more stable results compared to the diagonal method but the computational cost will be inevitably higher [28].

The JJ-Hunter MFPT Calculation Method commences by preparing a square matrix of ones, E, and an exact copy of the transition matrix M, referred to as PDD. The algorithm then initializes two zero vectors, rD and SD, corresponding to the residual and sum of differences, respectively. Iteratively, for each state from $m-1$ to 1, the algorithm updates the sum of the modified matrix's current row in SD. This sum then aids in the recalculation of transition probabilities within M_{PDD} for states below the current index by adjusting the probabilities based on the ratio of product terms from the modified matrix and the sum for the current state.

Following this, the algorithm sets the initial value of the residual differences vector, $rD[0]$, to 1 and computes the residual differences for subsequent states. Each entry of rD is calculated as the accumulation of previously computed values scaled by the corresponding modified transition probabilities, normalized by the sum of differences. After computing the total of residual differences, $TOTD$, a normalized transition distribution, $norm$, is derived as the ratio of rD to $TOTD$. Subsequently, a projection matrix PiD is formed by the outer product of the unit vector and the transpose of $norm$.

In the final steps, the matrix ZD is established as the difference between the identity matrix and M adjusted by PiD. The diagonal difference matrix DD is then calculated from the inverse of the diagonal elements of PiD. Ultimately, the MFPT matrix is determined as the product of $(I - ZD + E \times \text{diag}(ZD))$ and DD, providing a comprehensive matrix of mean first passage times for all state transitions. This detailed algorithm safely encapsulates the dynamic processes involved in calculating the MFPT, ensuring stable and consistent results.

Algorithm 3 MFPT JJ-Hunter Method

Require: Transition Matrix M of size $N \times N$

procedure JJ-HUNTER METHOD
  Define $P \leftarrow Array(M)$
  Let $m \leftarrow Len(P) = N$
  Define an identity matrix I of size $m \times m$
  Define a vector of ones $e1$ of length m
  Define a matrix of ones E of size $m \times m$
  Create a copy of P, $PPD \leftarrow Copy(P)$
  Define a vector of zeros rD of length m
  for $n = m-1$ to 1 **do**
    $SD[n] \leftarrow Sum(PPD[n,:n])$
    for $i = 0$ to $n-1$ **do**
      for $j = 0$ to $n-1$ **do**
        $PPD[i,j] \leftarrow PPD[i,j] + PPD[i,n] \times \frac{PPD[n,j]}{SD[n]}$
      end for
    end for
  end for
  $rD[0] \leftarrow 1$
  for $n = 1$ to $m-1$ **do**
    for $i = 0$ to $n-1$ **do**
      $rD[n] \leftarrow rD[n] + rD[i] \times \frac{PPD[i,n]}{SD[n]}$
    end for
  end for
  $TOTD \leftarrow Sum(rD)$
  $norm \leftarrow \frac{rD}{TOTD}$
  $PiD \leftarrow e1 \otimes norm$
  $ZD \leftarrow Inv(I - P + PiD)$
  $DD \leftarrow Diag(\frac{1}{Diag(PiD)})$
  $MFPT \leftarrow (I - ZD + E \times Diag(ZD) \otimes DD)$
  $MFPT \leftarrow \frac{MFPT}{time\_step}$
  return $MFPT$
end procedure

3.3 Bias Optimisation

The bias optimisation procedure begins with the consideration of a transition matrix M of size $m \times m$. The aim is to optimize a set of parameters for Gaussian functions to minimize the mean first passage time.

Initially, for each iteration i from 1 to 1000, a set of parameters G^i is generated. This set includes A^i, the amplitude, μ^i, the mean values and σ^i, the standard deviations. Using these parameters, a corresponding bias u^i_{bias} is computed through a Gaussian function $\mathscr{G}(G^i)$.

This bias is then applied to the original Markov transition matrix M to produce a biased matrix M_{biased}. The algorithm then calculates the mean first passage time for this biased matrix, denoted as MFPT, using a calculating MFPT Algorithm 2, which determines the average time required for the Markov chain to reach a specific end state from a start state. If the calculated MFPT is smaller than the current best-known MFPT (denoted as $\text{MFPT}_{\text{best}}$), then $\text{MFPT}_{\text{best}}$ is updated to this new value, and the parameters G^i corresponding to this optimal MFPT are stored as G_{best}.

After all iterations are complete, the algorithm proceeds to optimize the best set of parameters G_{best} using L-BFGS-B method, yielding $G_{\text{optimised}}$. The optimised parameters include the amplitude A, which corresponds to the first n elements of $G_{\text{optimised}}$, the mean values μ_k and μ_y, and the standard deviations σ_x and σ_y, extracted from the respective segments of $G_{\text{optimised}}$.

Next we can obtain the corresponding mean vector and covariance matrix from $G_{\text{optimised}}$: the mean vector μ is in the form of $(\mu_x\ ,\ \mu_y)$, and the covariance matrix $M_{\text{covariance}}$ is calculated as a diagonal matrix with elements σ_x^2 and σ_y^2. Finally, the procedure returns the optimised parameters $G_{\text{optimised}}$, the mean vector μ, and the covariance matrix $M_{\text{covariance}}$, which are ready for use in further analyses or applications.

Algorithm 4 Bias Optimisation

Require: Transition Matrix M with size $m \times m$

procedure BIAS OPTIMISATION
  for $i = 1$ to 1000 **do**
    $G^i \leftarrow (A^i, \mu_x^i, \mu_y^i, \sigma_x^i, \sigma_y^i)$ ▷ Generate random parameters for Gaussian functions
    $u_{bias}^i \leftarrow \mathscr{G}(G^i)$ ▷ Compute corresponding bias using these parameters
    $M_{biased} \leftarrow BiasM(M, u_{bias}^i)$ ▷ Apply the bias to Markov matrix
    $MFPT \leftarrow CalMFPT(M_{biased}, start, end)$ ▷ Calculate the mean first passage time
    if $MFPT_{best} > MFPT$ **then**
      $MFPT_{best} \leftarrow MFPT$
      $G_{best} \leftarrow G^i$
    end if ▷ Save the shortest MFPT and corresponding parameters
  end for
  $G_{optimised} \leftarrow Optimizer(G_{best})$
  $A \leftarrow G_{optimised}[:n]$
  $\mu_x \leftarrow G_{optimised}[n:2n]$
  $\mu_y \leftarrow G_{optimised}[2n:3n]$
  $\sigma_x \leftarrow G_{optimised}[3n:4n]$
  $\sigma_y \leftarrow G_{optimised}[4n:5n]$
  for $j = 0$ to $n-1$ **do**
    $\mu \leftarrow (\mu_x[j], \mu_y[j])$
    $M_{covariance} \leftarrow [(\sigma_x[j]^2, 0), (0, \sigma_y[j]^2)]$
  end for
  return $G_{optimised}, \mu, M_{covariance}$
end procedure

3.4 Langevin Simulation

Langevin dynamics is a widely utilized approach in molecular simulations, particularly for sampling the phase space of systems in thermodynamic equilibrium. It mathematically models the dynamics of molecular system with the langevin equation [29]:

$$m\frac{dv}{dt} = \lambda v + \eta(t)$$

where m is the mass of the of the particle, v is its velocity, λ is the damping constant and $\eta(t)$ is the noise term.

For a system with N particles with masses M and a time dependent RV (random variable) $X(t)$, the langevin equation [30]:

$$M\ddot{X} = -\nabla U(X) - \gamma M\dot{X} + \sqrt{2M\gamma k_B T}R(t)$$

where γ is damping coefficient, T is the temperature and k_B is the Boltzmann constant and $R(t)$ is a delta-correlated stationary Gaussian process with zero-mean.

This incorporation of stochastic elements allows the simulation to escape local minima more effectively, facilitating a more comprehensive exploration of the potential energy landscape. Langevin dynamics is especially useful for systems where temperature control and integration over complex potential energy surfaces are critical [31].

In this project we use the langevin integrator to create the OpenMM simulation object. The MFPT CV Space Exploration Algorithm 5 begins by defining the base Hamiltonian potential for the system as $H = U(x)$. It then establishes the initial state $x^0(0)$ and the target state x_{target}, initializing the bias as $U^0_{bias}(\xi)$ using a Gaussian distribution parameterized by μ^0, σ^0, ξ, and $N_{gaussian}$. A convergence threshold δ is set to determine proximity to the target state. The procedure iterates until the target is reached, employing MD simulations with the perturbed potential $H = U(x) + U^i_{bias}(\xi)$. Trajectories $x^i(t)$ are collected with a preset maximum length, checking if any state x within the trajectory comes within δ of x_{target}. If the target is reached, the loop exits. Otherwise, DHAM is applied to compute an unbiased transition matrix $M_{unbiased}$, which is cleaned of any rows and columns full of zeroes to yield $M_{working}$. The algorithm identifies the first point of the current trajectory as the new local start point x_{start}, and the closest point to x_{target} as the local end point x_{end}. A bias optimization algorithm then adjusts $U^i_{bias}(\xi)$ for the next iteration, seeking to improve the approach towards the target in successive cycles.

Algorithm 5 MFPT CV Space Exploration

procedure MFPT ENHANCED SAMPLING
    Define system with Hamiltonian $H = U(x)$
    Define the start state $x^0(0)$ and target state x_{target}
    Initialize the bias $U^0_{bias}(\xi) \leftarrow \mathscr{G}(\mu^0, \sigma^0, \xi, N_{gaussian})$
    Define the convergence threshold δ
    while *NotReach* **do**
        Run MD with perturbed potential $H = U(x) + U^i_{bias}(\xi)$
        Record trajectory $x^i(t)$ with maximum length T
        for Each x in $x^i(t)$ **do**
            if $|x - x_{target}| < \delta$ where x **then**
                Reach Target
                $NotReach \leftarrow False$
            end if
        end for
        Use DHAM algorithm to obtain $M_{unbiased}$
        Clean full-zero rows and columns to get the working matrix $M_{working}$
        Choose first point as the local start point, $x_{start} \leftarrow x^i(0)$
        Calculate the closest point to the target as local end point, x_{end}
        Use bias optimisation algorithm to find the the optimal bias $U^{i+1}_{bias}(\xi)$
        Increment i by 1
    end while
end procedure

3.5 PCA Simplification

The computational time significantly increases with higher dimensions of collective variables. In this project, we aim to project a six-dimensional CV space into a two-dimensional principal component space. This approach differs from our original MFPT enhanced sampling algorithm, as referenced in 5, by incorporating several crucial PCA transformations. Initially, we will project the original trajectory into the PC space. Subsequent steps will involve the forward and backward transformations of the Gaussian parameters.

The PCA Transformation Algorithm 6 begins by initializing a PCA [32] object configured to reduce the data into two principal components. This object, denoted as `pca`, is then employed to fit the data, encapsulated by the variable x, using the `pca.fit(x)` method. Upon successful fitting, the model is saved for subsequent transformations, ensuring that the original fit can be reused without refitting the data. For further operations, the model is reloaded as `pca_reload`. Depending on the transformation requirements, the data may either be projected into the PC space using the `transform(x)` method or reverted to its original space through `inverse_transform(x)`. The outcome, $x_{transformed}$, is either the data in PC space or its original form, ready for further analysis or processing.

Algorithm 6 PCA Transformation

Require: Data x to be fitted

procedure PCA TRANSFORMATION
    Create a PCA object $pca \leftarrow PCA(n = 2)$
    Fit data and get the model $pca.fit(x)$
    Save the model for next transformation
    Reload the model as pca_{reload}
    if Transform to PC space **then**
        $x_{transformed} \leftarrow transform(x)$
    end if
    if Transform from PC space **then**
        $x_{transformed} \leftarrow inverse\_transform(x)$
    end if
    return $x_{transformed}$
end procedure

However, the gaussian parameters for calculating the bias are saved in the form of $G(A, \mu, \sigma)$. The transformation Algorithm 6 cannot directly obtain the σ_{pca}. Hence, steps of processing the original gaussian parameters to the form of $G(A, \mu, \Xi)$, where Ξ is the covariance matrix calculated with σ, are needed. The Gaussian parameters back transformation Algorithm 7 is designed to convert Gaussian parameters from the PC space back to their original Cartesian form. Initially, the mean vector μ undergoes a back transformation through the function `Backtransform(`μ`)`, which adjusts μ from PC space to its equivalent in Cartesian coordinates, denoted as $\mu_{cartesian}$. This process is systematically applied to each element of the covariance matrix Ξ , where each $\Xi[i]$ is individually transformed back to $\Xi_{cartesian}[i]$ using the same back transformation method. Upon completion of these transformations for all elements, the algorithm constructs the Gaussian parameters in their original form as $G_{cartesian}$, which comprises the amplitude A , the back-transformed mean $\mu_{cartesian}$, and the back-transformed covariance matrix $\Xi_{cartesian}$. Then we can use the $G_{cartesian}$ to update our OpenMM simulation and collect the new trajectory.

Algorithm 7 Gaussian parameters back transformation

Require: Gaussian parameters in PC space, Gaussian parameters in (A, μ, Ξ) form

procedure BACK TRANSFORMATION
    $\mu_{cartesian} \leftarrow Backtransform(\mu)$
    for $i = 0$ to $n - 1$ **do**
        $\Xi_{cartesian}[i] \leftarrow Backtransform(\Xi[i])$
    end for
    return $G_{cartesian} \leftarrow (A, \mu_{cartesian}, \Xi_{cartesian})$
end procedure

The MFPT CV Space Exploration Algorithm 8 facilitates the transition from a 6D CV space to a 2D PC space. It begins by defining the system's Hamiltonian $H = U(x)$, alongside initializing the start state $x^0(0)$ and target state x_{target}. An initial bias $U^0_{bias}(\xi)$ is set up using Gaussian parameters, and a convergence threshold δ is defined. Then MD simulations are conducted under a perturbed potential $H = U(x) + U^i_{bias}(\xi)$, and the trajectories $x^i(t)$ are recorded. If any state within these trajectories approaches the target state within the threshold δ, the target is considered reached and we break the loop and finish our simulation.

Continuously, the algorithm employs a PCA transformation to map the coordinates ξ into the PCA space, obtaining ξ_{PCA} and x^i_{PCA}. Depending on the iteration, either a transformation of the initial bias is applied or the bias from the previous iteration is reloaded in the PCA space. The DHAM 1 is then used with the biased potentials and PCA-transformed states to compute an unbiased transition matrix $M_{unbiased}$, which is subsequently pruned of zero entries to form $M_{working}$.

The algorithm sets the first point of x^i_{PCA} as the new local start point and identifies the closest point to the target as the local end point. A bias optimization algorithm 4 refines the next iteration's bias, which is then saved in the PC space. This updated bias is finally back-transformed to the original space using the Gaussian parameters back transformation method 7, preparing it for the next MD simulation cycle. This iterative process continues until the target is consistently reached, indicating successful exploration and sampling of the CV space from 6D CV space down to 2D PC space using PCA.

Algorithm 8 MFPT CV Space Exploration(6D to 2D PC space)

procedure MFPT ENHANCED SAMPLING
    Define system with Hamiltonian $H = U(x)$
    Define the start state $x^0(0)$ and target state x_{target}
    Initialize the bias $U^0_{bias}(\xi) \leftarrow \mathscr{G}(\mu^0, \sigma^0, \xi, N_{gaussian})$
    Define the convergence threshold δ
    while $NotReach$ **do**
        Run MD with perturbed potential $H = U(x) + U^i_{bias}(\xi)$
        Record trajectory $x^i(t)$ with maximum length T
        for Each x in $x^i(t)$ **do**
            if $|x - x_{target}| < \delta$ where x **then**
                Reach Target
                $NotReach \leftarrow False$
            end if
        end for
        Use PCA transformation algorithm to obtain ξ_{PCA} and x^i_{PCA}
        if $i = 0$ **then**
            $U^0_{bias}(\xi_{PCA}) \leftarrow Transform(U^0_{bias}(\xi))$
        else
            Reload $U^i_{bias}(\xi_{PCA})$
        end if
        Use DHAM algorithm with $U^i_{bias}(\xi_{PCA})$ and x^i_{PCA} to obtain $M_{unbiased}$
        Clean full-zero rows and columns to get the working matrix $M_{working}$
        Choose first point as the local start point, $x_{start} \leftarrow x^i_{PCA}(0)$
        Calculate the closest point to the target as local end point, x_{end}
        Use bias optimisation algorithm to find the the optimal bias $U^{i+1}_{bias}(\xi_{PCA})$
        Save the bias in PC space $U^{i+1}_{bias}(\xi_{PCA})$
        Use Gaussian parameters back transformation to get $U^{i+1}_{bias}(\xi)$
        Increment i by 1
    end while
end procedure

4 Results

This chapter will show the outcomes of 1D, 2D langevin dynamics simulation and 6D langevin simulation with the assistance of PCA method.

4.1 1D Langevin Simulation

In this 1D langevin simulation, we only consider a single hydrogen atom where a multi-well free energy surface is added along the x axis and very large constant force is applied to the rest of two axes, leading to the whole simulation in 1D case. After setting the real space boundary as $(0, 2\pi)$, we initialise a set of start gaussian parameters and collect the trajectory from the OpenMM simulation with maximum length T. Then the corresponding partially observed Markov matrix will be obtained with the help of DHAM Algorithm. We can get the optimised bias from Algorithm 4 and apply it to update the simulation. New trajectories will be collected and new unbiased Markov matrix will be reconstructed. The iteration will end till the final target state is reached. This is a brief overview of the 1D langevin simulation and the key steps are summarised in Algorithm 5.

As shown in Figure 4.1, the total bias applied for each iteration is correctly set, since the corresponding mean value is always on the left of our local start state and the standard deviation is suitable to push it towards right as well. In the beginning, our start point is at 2 for x coordinate. After four continuous propagation, it is pushed to around 5 which is our pre-set target state.

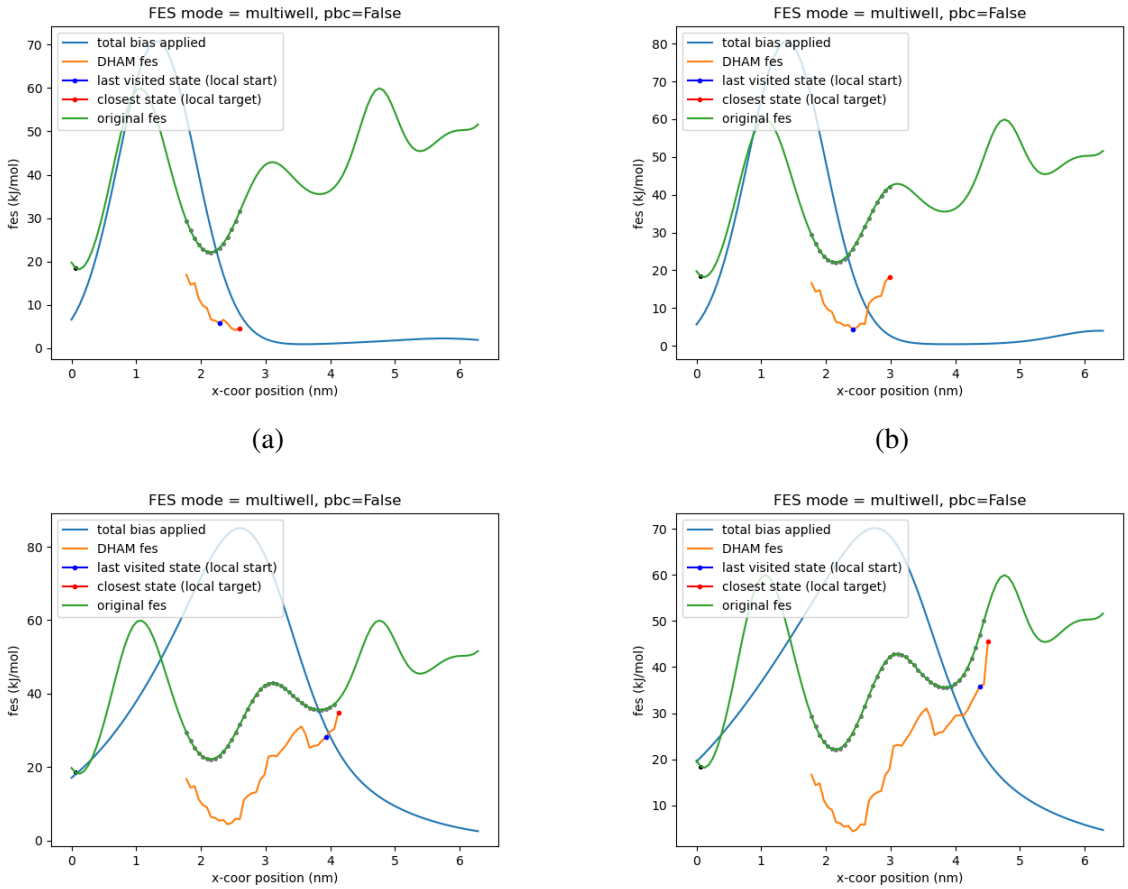


Figure 4.1: This figure depicts four consecutive propagations in a 1D Langevin simulation involving a single atom, which can move along only one degree of freedom. (a), (b), (c) and (d) show the results of propagation 1 to 4 respectively. As shown in the legend, the blue dot indicate the local start and the red dot is the end points where the end points are the closest points towards the global target for each iteration step. The green line represents the original free energy surface, while the blue line shows the perturbing bias potential. The orange line illustrates the free energy surface calculated from reconstructed unbiased markov matrix returned by DHAM algorithm.

4.2 2D Langevin Simulation

In 2D langevin simulation, similarly as 1D case, we consider a single atom. Then we apply the multi-well free energy surface on both x and y axis and add a great harmonic potential, for example $1000 \times z^2$, along the z axis. The rest of the process is similar to 1D langevin simulation but some variables such as boundary and gaussian parameters should be in 2D version.

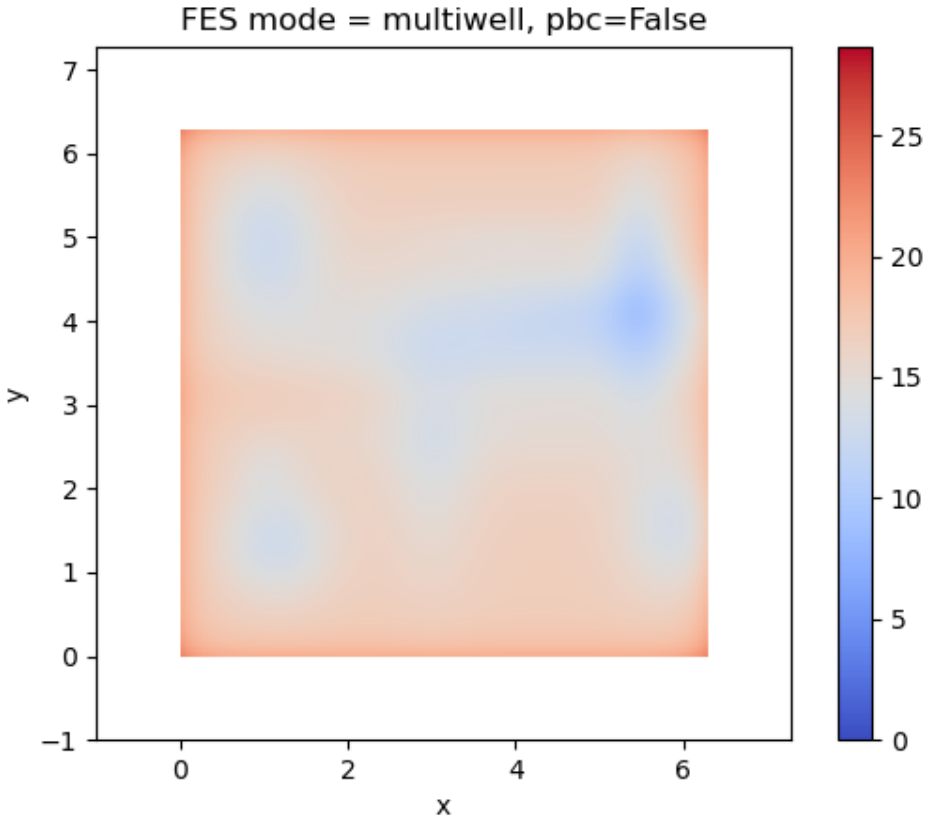
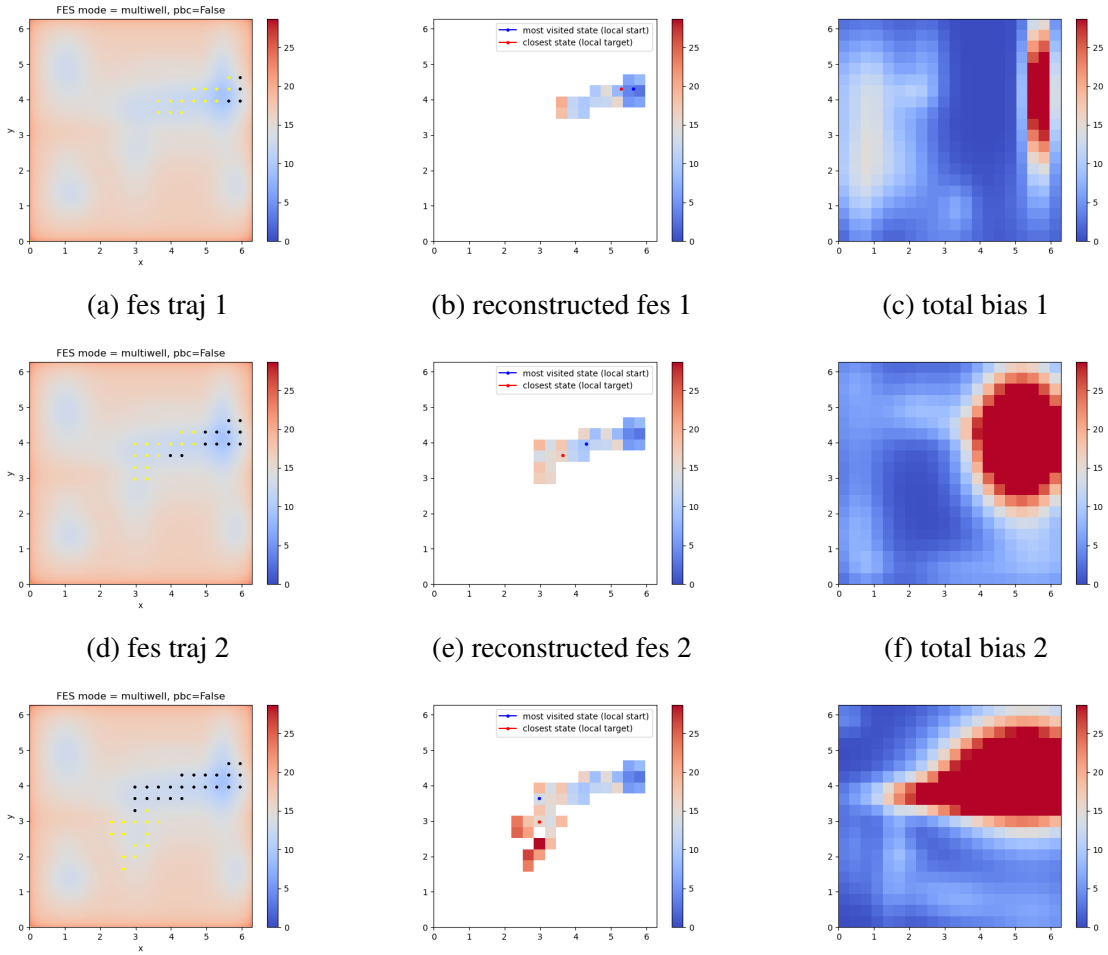


Figure 4.2: This figure illustrates the base potential (Hamiltonian) of the 2D langevin dynamics simulation system. The x and y boundaries are both $(0, \pi)$ similarly as 1D case. For the mode of free energy surface, multi-well with 9 wells and 1 barrier is used. Additionally, all enery is in Kj/mol unit.

Figure 4.2 gives the base Hamiltonian of the 2D langevin simulation system. Figure 4.3 shows that the total bias applied to each propagation is on the top right corner and tends to push the trajectory to the bottom left corner. The 2D langevin simulation is correctly biasing the system towards the target state.



(a) fes traj 1 (b) reconstructed fes 1 (c) total bias 1

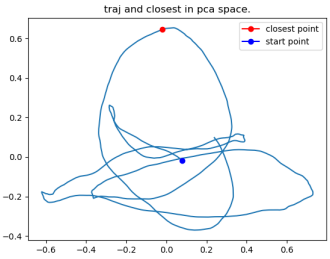
(d) fes traj 2 (e) reconstructed fes 2 (f) total bias 2

(g) fes traj 3 (h) reconstructed fes 3 (i) total bias 3

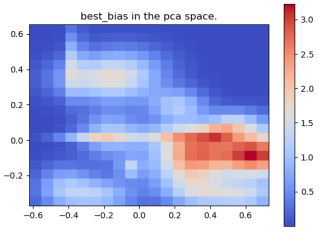
Figure 4.3: These figures shows the basic result of 2D langevin simulation. Each row corresponds to a propagation and the columns on the left, middle and right illustrate the free energy surface with trajectory, reconstructed free energy surface from DHAM algorithm and bias applied to each propagation respectively.

4.3 6D Langevin Simulation with PCA

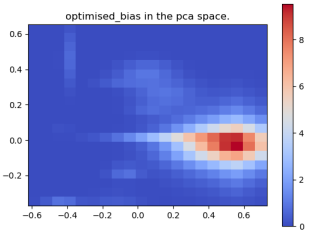
In this 6D CV space to 2D PC space langevin simulation, two atoms labeled with atom 1 and atom 2 and have coordinate (x_1, y_1, z_1) and (x_2, y_2, z_2) respectively. Multi-well free energy surface is applied to x_1 and y_1 together with very large constant forces added along the left four coordinates. The detail procedure is summarised in Algorithm 8. The boundary in PC space is updated for each propagation to ensure every trajectory is included and the bias applied is inside the boundary.



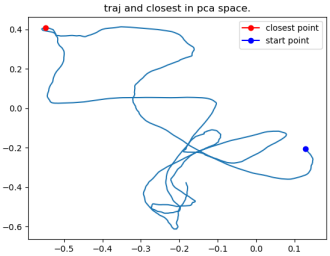
(a) Propagation 0 trajectories in PC space



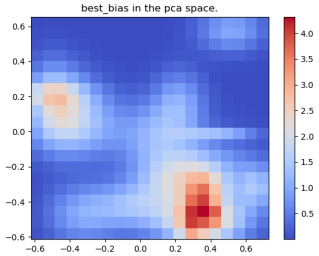
(b) Bias from random selects before optimisation



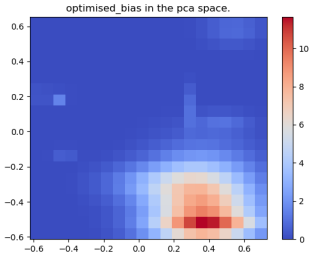
(c) Optimised bias



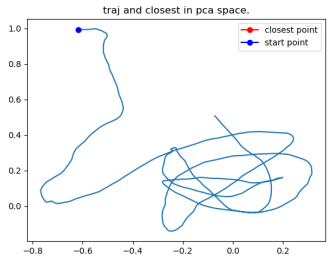
(d) Propagation 5 trajectories in PC space



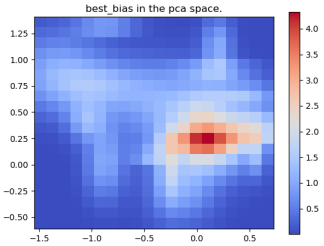
(e) Bias from random selects before optimisation



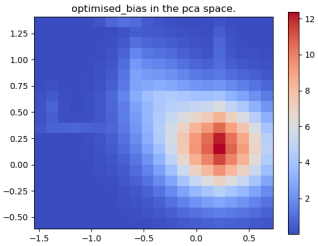
(f) Optimised bias



(g) Propagation 8 trajectories in PC space



(h) Bias from random selects before optimisation



(i) Optimised bias

Figure 4.4: These figures contain the representation of the movement of trajectories with the application of the optimal biases. (a), (d) and (e) are the trajectories in PC space for propagation 0, 5 and 8 respectively. (b), (e) and (h) are the best bias selected from 1000 random try. (c), (f) and (i) are the corresponding best bias after optimization with Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Box constraints (L-BFGS-B) method.

Figure 4.5 shows the plots of trajectories in PC space for propagation step $0, 2, 5, 8$ and 9 with the global target point. The local end point for each iteration step moves towards the target state after the optimised bias applied.

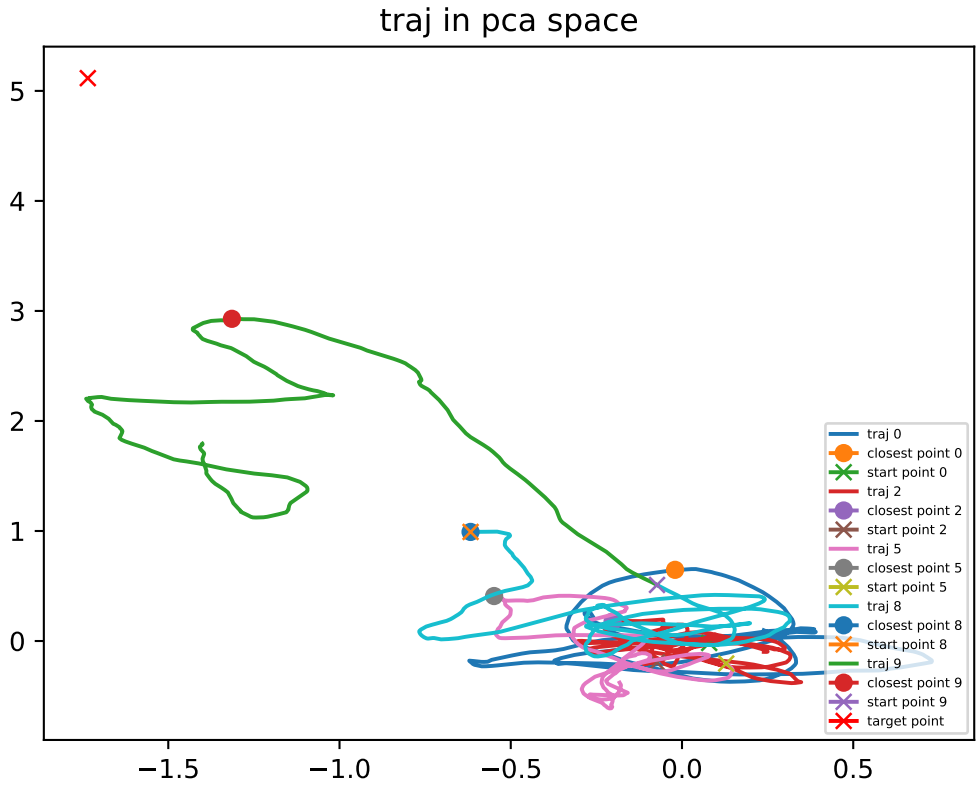


Figure 4.5: This figure illustrates the trajectories in PCA space for propagation steps 0, 2, 5, 8, and 9, highlighting the final target point.

5 Conclusion

In this dissertation, we have thoroughly examined the application of MSM in enhancing sampling techniques to overcome the challenges of MD simulations, particularly the issue of simulations becoming trapped in local potential minima. The study does not benchmark the effectiveness between the MFPT technique and other enhanced sampling. However, based on the result from Professor Edina Rosta's group [1], in sodium chloride system, kinetics-optimised enchanced sampling with MFPT provides a more effective approach than other enhanced sampling techniques, such as MetaDynamics. From the result of 1D and 2D langevin simulation, we can easily reach the global target state in several propagations. Computational time would greatly increase for high dimentional cases, in order to extend the dimensionality that the MFPT optimization technique can handle, dimension reduction techniques such as PCA is considered to project the collective variables onto a lower-dimensional state space. A program is developed to project 6D CV space to a 2D PCA space and in each iteration the applied optimal bias successfully push every start point to our target point. Linear dimensionality reduction techniques like PCA might be limiting if the data contains multiple and varying structures. The transformation approach can overlook important local variations in the data. Future work could explore the application of non-linear techniques such as Kernel PCA, which could better capture the dynamics of the system.

References

[1] T. Wei, B. Dudas, and E. Rosta, "Kinetics-optimized enhanced sampling using mean first passage times," 2024.

[2] D. Frenkel and B. Smit, *Understanding molecular simulation : from algorithms to applications / Daan Frenkel, Berend Smit.* Amsterdam: Academic Press, third edition. ed., 2023.

[3] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids / Michael P. Allen and Dominic J. Tildesley.* Oxford scholarship online, Oxford: Oxford University Press, second edition. ed., 2017.

[4] B. E. Husic and V. S. Pande, "Markov state models: From an art to a science," *Journal of the American Chemical Society*, vol. 140, no. 7, pp. 2386–2396, 2018.

[5] F. Noé and E. Rosta, "Markov Models of Molecular Kinetics," *The Journal of Chemical Physics*, vol. 151, p. 190401, 11 2019.

[6] G. Torrie and J. Valleau, "Nonphysical sampling distributions in monte carlo free-energy estimation: Umbrella sampling," *Journal of computational physics*, vol. 23, no. 2, pp. 187–199, 1977.

[7] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman, "The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method," *Journal of computational chemistry*, vol. 13, no. 8, pp. 1011–1021, 1992.

[8] A. Laio and M. Parrinello, "Escaping free-energy minima," *Proceedings of the National Academy of Sciences - PNAS*, vol. 99, no. 20, pp. 12562–12566, 2002.

[9] A. Barducci, M. Bonomi, and M. Parrinello, "Metadynamics," *Wiley interdisciplinary reviews. Computational molecular science*, vol. 1, no. 5, pp. 826–843, 2011.

[10] J. D. Chodera and F. Noé, "Markov state models of biomolecular conformational dynamics," *Current opinion in structural biology*, vol. 25, pp. 135–144, 2014.

[11] V. S. Pande, K. Beauchamp, and G. R. Bowman, "Everything you wanted to know about markov state models but were afraid to ask," *Methods (San Diego, Calif.)*, vol. 52, no. 1, pp. 99–105, 2010.

[12] J.-H. Prinz, H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, and F. Noé, "Markov models of molecular kinetics: Generation and validation," *The Journal of chemical physics*, vol. 134, no. 17, pp. 174105–174105–23, 2011.

[13] F. Noé, A. Tkatchenko, K.-R. Müller, and C. Clementi, "Machine learning for molecular simulation," *Annual review of physical chemistry*, vol. 71, no. 1, pp. 361–390, 2020.

[14] E. Rosta and G. Hummer, "Free energies from dynamic weighted histogram analysis using unbiased markov state model," *Journal of chemical theory and computation*, vol. 11, no. 1, pp. 276–285, 2015.

[15] G. Bowman, V. Pande, and F. Noé, *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*, vol. 797. Springer Dordrecht, 01 2014.

[16] A. Kells, V. Koskin, E. Rosta, and A. Annibale, "Correlation functions, mean first passage times, and the kemeny constant," *The Journal of chemical physics*, vol. 152, no. 10, pp. 104108–104108, 2020.

[17] N. G. v. Kampen, *Stochastic processes in physics and chemistry / N.G. Van Kampen.* North-Holland personal library, Amsterdam: Elsevier, 3rd ed. ed., 2007.

[18] R. B. Best and G. Hummer, "Coordinate-dependent diffusion in protein folding," *Proceedings of the National Academy of Sciences - PNAS*, vol. 107, no. 3, pp. 1088–1093, 2010.

[19] A. C. Pan and B. Roux, "Building markov state models along pathways to determine free energies and rates of transitions," *The Journal of chemical physics*, vol. 129, no. 6, pp. 064107–064107–8, 2008.

[20] A. Szabo, K. Schulten, and Z. Schulten, "First passage time approach to diffusion controlled reactions," *The Journal of chemical physics*, vol. 72, no. 8, pp. 4350–4357, 1980.

[21] R. Zwanzig, "Dynamical disorder: Passage through a fluctuating bottleneck," *The Journal of chemical physics*, vol. 97, no. 5, pp. 3587–3589, 1992.

[22] J. G. Kemeny and J. L. J. L. Snell, *Finite Markov chains / John G. Kemeny and J.Laurie Snell*. University series in undergraduate mathematics, Princeton, N.J. ;: Van Nostrand, 1960.

[23] D. J. Aldous, "Lower bounds for covering times for reversible markov chains and random walks on graphs," *Journal of theoretical probability*, vol. 2, no. 1, pp. 91–100, 1989.

[24] I. T. Jolliffe, *Principal component analysis / I.T. Jolliffe.* Springer series in statistics, New York: Springer, 2nd ed. ed., 2002.

[25] M. Ringner, "What is principal component analysis?," *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[26] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[27] C. M. Bishop, *Pattern recognition and machine learning / Christopher M. Bishop.* Information science and statistics, New York: Springer, 2006.

[28] J. J. Hunter, "The computation of the mean first passage times for markov chains," *Linear algebra and its applications*, vol. 549, pp. 100–122, 2018.

[29] D. S. D. S. Lemons and P. Langevin, *An introduction to stochastic processes in physics : containing "On the theory of Brownian motion" by Paul Langevin, translated by Anthony Gythiel / Don S. Lemons.* Baltimore ;: Johns Hopkins University Press, 2002.

[30] T. Schlick, *Molecular modeling and simulation : an interdisciplinary guide / Tamar Schlick.* Interdisciplinary applied mathematics ; v. 21, New York ;: Springer, 2nd ed. ed., 2010.

[31] E. Paquet and H. L. Viktor, "Molecular dynamics, monte carlo simulations, and langevin dynamics: A computational review," *BioMed Research International*, vol. 2015, no. 1, p. 183918, 2015.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.