

Freie Universität



Berlin



Max Planck Institute for
Biophysical Chemistry

Master's Thesis

On the Topic

**Using Brownian Dynamics
Simulation in the Construction of
Benchmark Trajectories for Use in
Optimizing Markov State Models
Generation**

written by

Peleg Sapir

at the Max-Planck-Institute for Biophysical Chemistry
as part of a Master of Science program at the Free University of Berlin

Referees:

Prof. Dr. Betina Keller

Prof. Dr. Helmut Grubmüller

Submission Date:

December 2, 2019

Abstract

Protein function can be predicted from their dynamics, however a more robust and accurate method is sought-after. An important tool in the analysis of protein dynamics are Markov state models, which approximate the meta-stable states of a system and the transition rates between these states. The construction of a Markov state model from a protein trajectory follows a protocol consisting of several steps: coarse graining the trajectory, counting the transitions between each states and estimating the corresponding transition rates. It is desirable to automate and optimize this process, using benchmark trajectories which should yield known meta-stable states and transition times. This Master's thesis describes the developement and validation of a Brownian dynamics simulation, utylizing adjustable potential landscapes. The simulation is used to generate trajectories functioning as said benchmarks. An example of applying the Markov state model generation method to a benchmark trajectory is discussed.

Keywords: Biophysics, Protein Dynamics, Dynasome, Brownian dynamic simualtion, Markov state models, PyEMMA.

Contents

1. introduction	1
2. Theoretical Foundation	5
2.1. Markov State Models of Protein Dynamics	5
2.2. Brownian Dynamics	13
3. Methods	23
3.1. Brownian Dynamics Simulation	23
3.2. Constructing Markov State Models from Simulated Protein Trajectories . .	29
4. Results and Discussion	33
4.1. Simulation Validation	33
4.2. Constructing Markov State Model from Simulated Brownian Dynamics Trajectories	47
5. Conclusions and Outlook	59
6. Acknowledgements	61
Bibliography	63
Appendices	65
A. Simulation Code	67
B. Derivation of Transition Rates via Kramer’s Approximation for Symmetric Double Log-Gaussian Potential Wells	75
C. Derivation of Transition Rates Between Two States	77
C.1. Single Transition	77
C.2. Many Transitions	80

1. Introduction*

As proteins are some of the fundamental building blocks of any biological system[1], and their diverse functions determine the very structure and behavior of all biological systems, the ability to accurately predict the function of a protein is a sought-after goal in many branches of the natural sciences. One common predictor of protein function is their structure, derived in large part from their amino acid sequence. Thus, the similarity of two amino acid sequences - their homology - can be used as a rough predictor of their biological functions[2]. Strong relations also exist between protein structure and their function.

Another connection that is sought-after is the relation between protein *dynamics* and their function. A promising approach to the prediction of protein function from their dynamic properties was made by Hensen et al (2012)[3]: they analyzed the trajectories of a group of 112 proteins, produced by molecular dynamic simulations. The proteins were simulated for 100 nano seconds, and their trajectories analyzed to yield 34 different observable quantities, which were used to assign points in a 34-dimensional corresponding to each protein. Using dimensionality reducing methods such as *Principal component analysis* (PCA), they were able to group proteins together and predict their function with a success rate of 57% to 61% based only on their proximity in the reduced dynamic space (Table 1.1). They refer to the space of a protein dynamics as the *Dynasome*.

In our research group, we employ a different method for comparing the dynamics of proteins. This method uses Markov state models (MSM) to approximate the meta-stable states of a system and the transition rates between them, from trajectories of proteins simulated by molecular dynamics. One protocol for derivation of an Markov state model uses time structure based independent component analysis (TICA) and k-means clustering to coarse-grain the trajectories, and employs a Bayesian method to estimate the transition rates in the system. This thesis is aimed at a first step of streamlining this process.

*Parts of the following introduction are based on a previous report I submitted as part of my work at the Theoretical and Computational Biophysics department at the Max Planck Institute for Biophysical Chemistry.

1. introduction

To that aim, suitable strategies need to be employed so Markov state models can be constructed in a consistent and robust way. However, it is difficult to form a single robust method of Markov state models generation without prior knowledge of the affects of different parameters and employed method in the Markov state model generation process. For example, how do different data clustering techniques perform? Is there an optimal technique that yields consistent, reliable results? A knowledge of the limitations of each step in the Markov state model generation process is also desirable.

Answering some of these and related questions can be done by developing a method that generate trajectories which yield known results when subjected to the Markov state models generation process. Brownian dynamic simulations provides a relatively simple way of generating trajectories that emulate the state space of a protein's dynamics and the transitions between the states.

By carefully choosing the potential landscapes in which the simulations are generated, trajectories of known thermodynamics and kinetic properties can be generated. An analysis of these trajectories by the same Markov state models generation method can then be performed, and the results compared to the expected results from the trajectories. This way, parameters and methods which are employed in the Markov state models generation process can be optimized, and the limits of each method can be tested.

This Master's thesis describes the development of a generalized Brownian dynamics simulation, and the methods employed to validate it. It uses a simple method of constructing potential landscapes that yield any number of meta-stable states in single-, two- and higher-dimensional spaces. It also discusses one example of employing the Markov state models generation method to a trajectory generated by the simulation and some limitations encountered in the process.

Table 1.1.: Prediction rate by function class in the Dynasome study by Hensen et al.
 Numbers in brackets indicate the result of a cross validation test.
 Taken with caption from [3].

Class	# of proteins	Correct predictions		
		Dynamics	Structure	Combined
Glycosidases	17	12 (11)	0 (0)	5 (4)
Esterases	13	4 (2)	6 (6)	0 (0)
Serine Proteinases	11	6 (6)	6 (5)	9 (9)
Metallopeptidases	6	4 (0)	0 (0)	5 (4)
Calcium-binding	5	2 (0)	4 (3)	4 (4)
Toxins	5	2 (0)	5 (5)	2 (0)
Signalling	5	4 (0)	1 (0)	2 (0)
DNA/Transcription	4	3 (2)	0 (0)	3 (0)
Peptidases	3	2 (0)	3 (3)	3 (3)
Total large groups (gray rows)	41	54 % (46 %)	29 % (27 %)	34 % (31 %)
Total small groups	28	61 % (7 %)	47 % (39 %)	68 % (39 %)
Total	69	57 % (30 %)	36 % (32 %)	48 % (35 %)

1. introduction

2. Theoretical Foundation

This chapter introduces the theoretical basis for the methods which were employed in the Master's thesis. The first section of the chapter is adapted with changes from the same section of a previous Master's thesis which was written by a member of our research group [4]. That work is in turn based on chapters 3 and 4 of a book on Markov state models [5].

Throughout this thesis, a specific set of units is used, both for theory and in simulations. We first set $k_B T = 1$, which means, since the SI units of energy are $\text{kg m}^2 \text{s}^{-2}$, that $\text{kg} = \text{s}^2 \text{m}^{-2}$. Next we set $\text{kg} = 1$, and thus $\text{m} = \text{s}$. Lastly, setting $\text{m} = 1$ yields a unit less scheme, used throughout the thesis, which measures energy in multiples of $k_B T$.

2.1. Markov State Models of Protein Dynamics

A commonly used method in the analysis of protein dynamics is estimating a Markov state model (MSM) that represents the structure of the system and the probabilities to transition between its different states. One advantage of using a Markov state model is that it can be approximated from a large set of relatively short trajectories, as opposed to using longer trajectories[6]. This allows for parallelization, by running several short simulations at the same time.

Examining a basic model can allow for a simple explanation of important characteristics of a Markov state model. The system is made of an ensemble of particles diffusing in a one-dimensional potential landscape with four potential wells (Figure 2.1A). We define an operator called the propagator, $P(\tau)$, which acts on the probability distribution of the ensemble at time t to yield the probability distribution of the system at time $t + \tau$, and is

2. Theoretical Foundation

defined as

$$\begin{aligned} P(\tau) \circ \rho_t(y) &= \rho_{t+\tau}(y) \\ &= \int_{\Omega} \rho_t(x) \cdot p(x, y; \tau) dx, \end{aligned} \quad (2.1)$$

where $p(x, y; \tau)$ is the transition probability density from state x to state y by a lag time τ .

The eigenvectors and eigenvalues of the propagator correspond to the processes in the system, in decreasing time scales: the first eigenvector, with corresponding eigenvalue λ_1 , is the equilibrium distribution of the system (the corresponding timescale of this process is $t_1 = \infty$). Since propagating the equilibrium distribution by time τ yields the same distribution, $\lambda_1 = 1$ (Figure 2.1(D)).

The next eigenvector describes the slowest transition - in the case of the simple model, this corresponds to crossing the middle barrier (Figure 2.1(A): green barrier, (C): function ϕ_2). The next two eigenvectors describe crossing the two smaller barriers, respectively (Figure 2.1(A): blue and orange barriers, (C): functions ϕ_3 and ϕ_4). The rest of the eigenvectors correspond to much faster processes: essentially, diffusion within the potential wells. The separation between the first 4 eigenvectors and the rest is clearly visible as a gap between λ_4 and λ_5 in Figure 2.1D.

Generally, the relation between the timescales of processes in a finite Markovian system and the eigenvalues of the propagator with time step τ is

$$t_i = -\frac{\tau}{\log(\lambda_i)}, \quad i = 1, 2, \dots, \alpha. \quad (2.2)$$

The equation above shows that the first eigenvalue $\lambda_1 = 1$ corresponds to the equilibrium process, with $t_1 = \infty$. Disregarding cases of degeneracy, the rest of the eigenvalues can be sorted in a descending order $\lambda_2 > \lambda_3 > \dots > \lambda_{\alpha+1} > \lambda_{\alpha}$, where for $i \geq 2$ the eigenvalues are in the open interval $(0, 1)$.

The assumption that the propagator is Markovian, i.e. that it has no memory, means that $P(\tau) \circ \rho_t(\mathbf{x})$ is dependent only on $\rho_t(\mathbf{x})$, and that the state of the system after time $k\tau$ is given by applying the propagator k times:

$$P(\tau)^k \circ \rho_t(\mathbf{x}) = \rho_{t+k\tau}(\mathbf{x}). \quad (2.3)$$

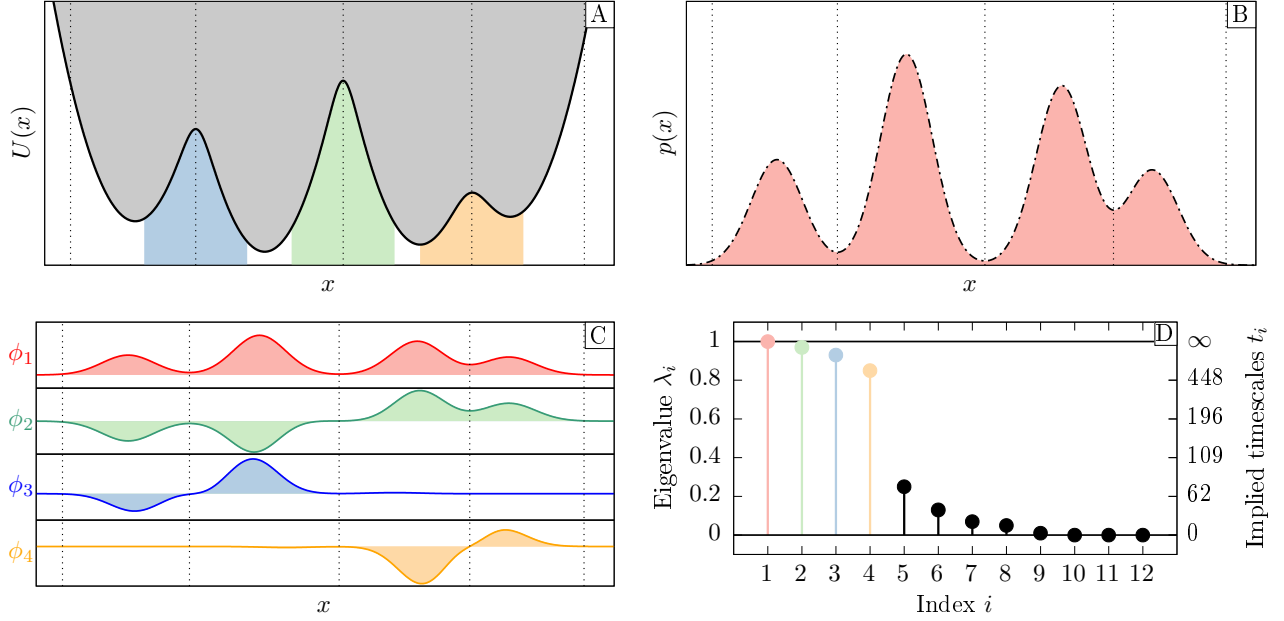


Figure 2.1.: An ensemble of particles diffusing between four potential wells. (A) the potential $U(x)$ with its three barriers signified with the colors blue, green and orange. (B) The equilibrium distribution of the system. (C) The first 4 eigenvectors, corresponding to the four slowest processes in the system. (D) The eigenvalues of the system, the first 4 eigenvalues corresponding to the 4 slowest processes. Figure recreated from [5].

This property is known as the Chapman-Kolmogorov equation, and it can be used to validate the Markovianity of the system. $P(\tau) \circ \rho_t(\mathbf{x})$ can be written as a projection on the eigenvectors $\{\phi_i\}$ of $P(\tau)$:

$$P(\tau) \circ \rho_t(\mathbf{x}) = \sum_{i=1}^{\infty} \lambda_i \cdot \langle \phi_i | \rho_t(\mathbf{x}) \rangle_{\mu} \cdot \rho_t(\mathbf{x}), \quad (2.4)$$

where the scalar product $\langle u(\mathbf{x}) | v(\mathbf{x}) \rangle_{\mu}$ is defined as

$$\langle u(\mathbf{x}) | v(\mathbf{x}) \rangle_{\mu} = \int_{\Omega} \frac{v(\mathbf{x})u(\mathbf{x})}{\mu(\mathbf{x})} d\mathbf{x}, \quad (2.5)$$

where Ω is the state space of the protein dynamics, and $\mu(\mathbf{x})$ its equilibrium distribution (i.e. $\mu = \phi_1$). This scalar product forces the projection to be hermitian, which means that the eigenvalues $\{\lambda_i\}$ are real and positive, and that a partition into discrete real and positive eigenvectors exists.

2. Theoretical Foundation

Using the Chapman-Kolmogorov equation, a more general form of Equation 2.4 can be written, valid for any whole number of successive applications of the propagator to the ensemble density:

$$\begin{aligned}\rho_{t+k\tau}(\mathbf{x}) &= P(\tau)^k \circ \rho_t(\mathbf{x}) \\ &= \sum_{i=1}^{\infty} \lambda_i^k \cdot \langle \phi_i | \rho_t(\mathbf{x}) \rangle_{\mu} \cdot \rho_t(\mathbf{x}).\end{aligned}\tag{2.6}$$

2.1.1. Construction of a Markov State Model from Protein Dynamics

A closely related operator to the propagator operator is the transfer operator $T(\tau)$. The transfer operator is defined on the functions

$$u_t(\mathbf{x}) = \frac{\rho_t(\mathbf{x})}{\mu(\mathbf{x})},\tag{2.7}$$

where $\mu(\mathbf{x})$ is the equilibrium probability of the system, and is defined as

$$T(\tau) = \frac{1}{\mu(\mathbf{y})} \int_{\Omega} p(\mathbf{x}, \mathbf{y}; \tau) \mu(\mathbf{x}) u_t(\mathbf{x}) d\mathbf{x}.\tag{2.8}$$

Unlike the propagator, which acts on a probability distribution at time t to yield a probability distribution at time $t + \tau$, the transfer operator yields functions that relate to the probability distributions by division in the equilibrium probability. Thus they are flat inside metastable states, and in essence show the transfer in probability distribution for a specific process. Figure 2.2 shows this case for the basic system discussed above.

The local flatness inside metastable states allows the approximation of the entire space Ω by a set of discrete domains (Figure 2.6). A Markov state model can then be approximated by a matrix by counting the transitions between the different states and how long the system spends in each state before transitioning, and employing, for example, a Bayesian estimation method to yield an estimation of the respective transition rates.

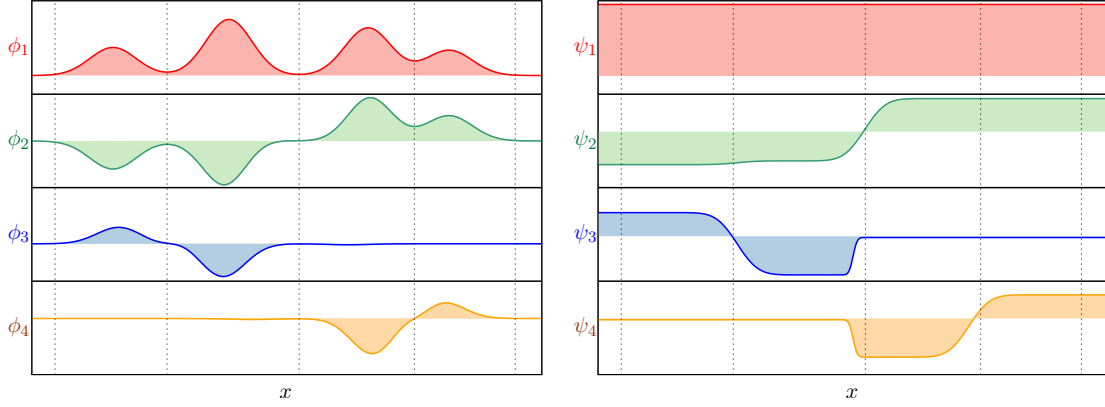


Figure 2.2.: First four eigenvectors of the propagator (left) and the analogous eigenvectors of the transfer operator (right). Figure recreated from [5].

2.1.2. Estimating Transition Rates from Transitions Counts

For a system with two states 1 and 2, and a transition $1 \rightarrow 2$ at time t_1 (Figure 2.3), an estimation of transition rate $k = k_{1 \rightarrow 2}$ by a Bayesian approach[7] is given by (see Appendix C for derivation)

$$p(k | t_1) = k e^{-kt_1}, \quad (2.9)$$

with the most likely value of k being

$$k_{\max} = \frac{1}{t_1}. \quad (2.10)$$

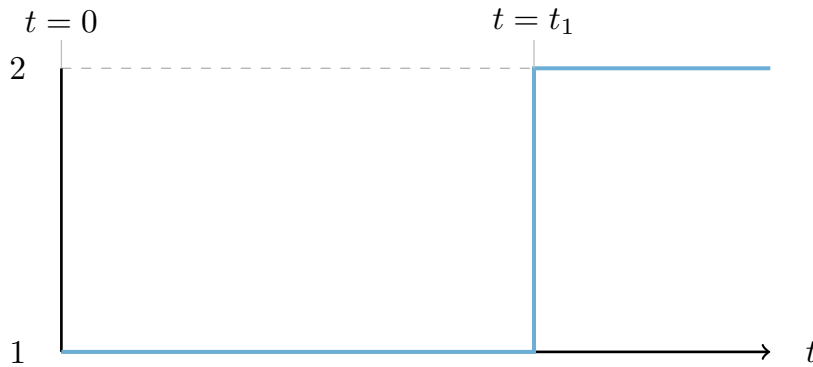


Figure 2.3.: A system transitioning from state 1 to state 2 at time $t = t_1$.

When n transitions $1 \rightarrow 2$ are observed at times $\{t_1, t_2, \dots, t_n\}$, and similarly $n - 1$ transitions $2 \rightarrow 1$ are observed at times $\{t'_1, t'_2, \dots, t'_n\}$ (Figure 2.5), we can define the time

2. Theoretical Foundation

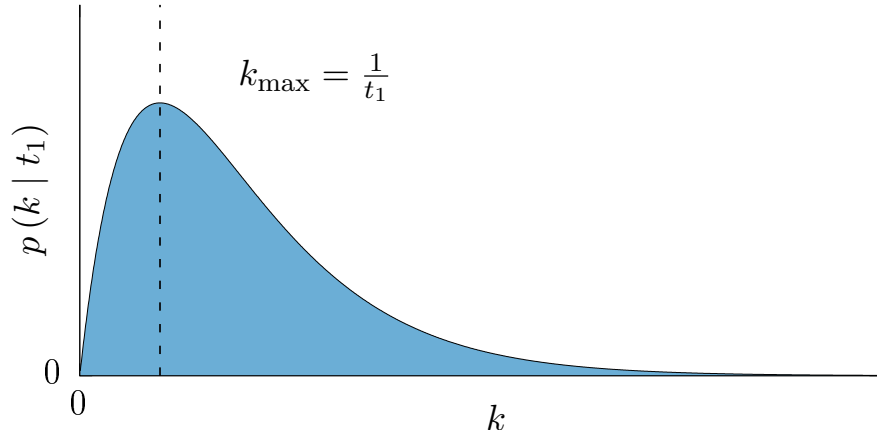


Figure 2.4.: Probability distribution of the transition rate value when observing a single transition at time $t = t_1$, using Bayesian estimation. The most likely value of the transition rate is $k_{\max} = \frac{1}{t_1}$.

periods the system spent in state 1 as

$$\Delta t_i = \begin{cases} t_1 & \text{if } i = 1 \\ t'_i - t_{i-1} & \text{if } i = 2, 3, \dots, n \end{cases}. \quad (2.11)$$

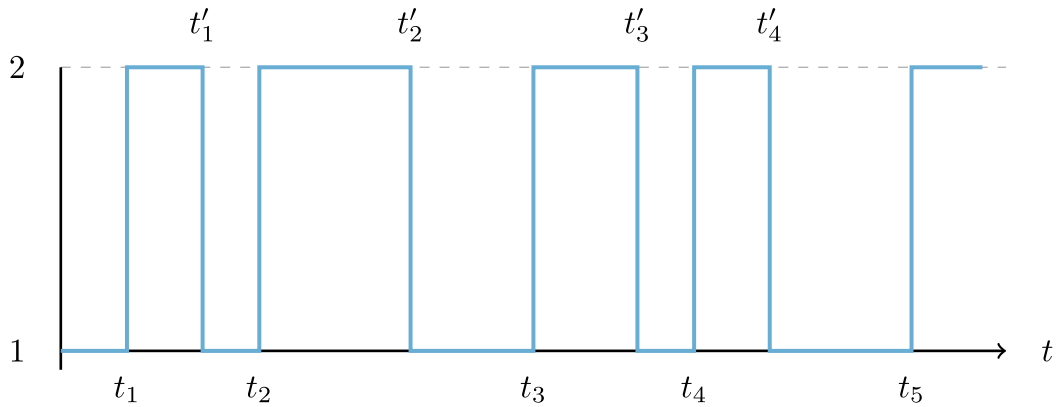


Figure 2.5.: A system transitioning from state 1 to state 2 at time $t = t_1$.

Using the same Bayesian approach as before, the probability distribution of the transition rate $k = k_{1 \rightarrow 2}$ is then

$$p(k | \{\Delta t_i\}) = k^n e^{-kn \langle \Delta t_i \rangle}, \quad (2.12)$$

with the most likely k being

$$k_{\max} = \frac{1}{\langle \Delta t_i \rangle}. \quad (2.13)$$

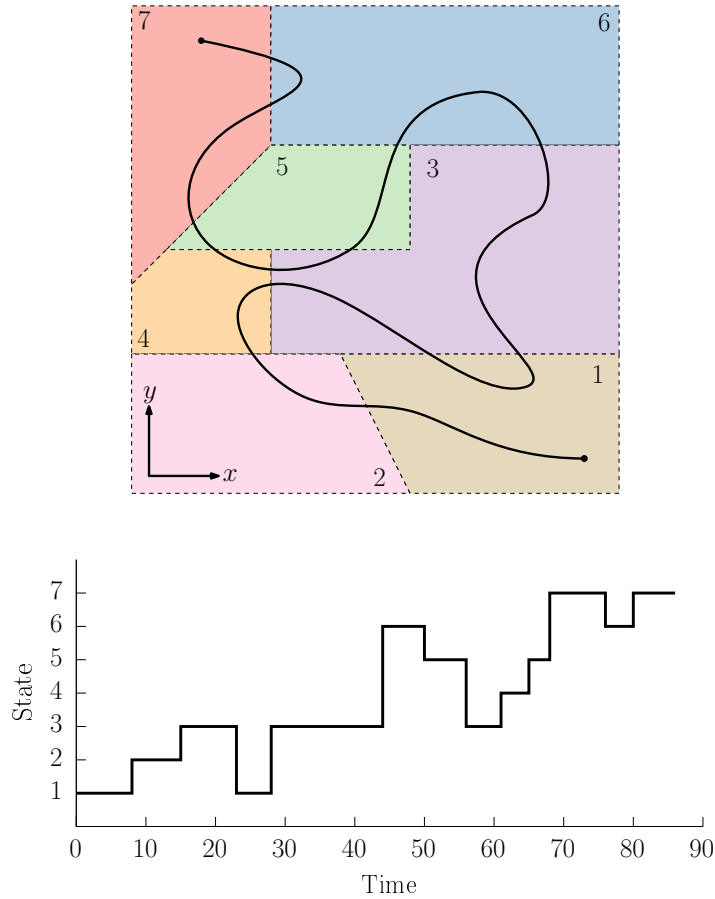


Figure 2.6.: Clustering a trajectory. Top: A trajectory in the space Ω , starting at the bottom right. Each color represents a different cluster of the space. Bottom: The resulting clustered trajectory, the state numbers correspond to the clusters in the top figure.

2.1.3. K-Means Clustering

One of the most widely used methods to group data into clusters is the k-means clustering method[8]. This method assigns n points to a given number $k < n$ of clusters, such that the positional variance in each cluster is minimized. Implementing an approximation to the

2. Theoretical Foundation

k-means method can be done using the following algorithm:

1. Choose k random points from the data set to function as initial centers for the clusters.
2. Assign each point in the data set to its closest center (usually using Euclidean distance), resulting in k clusters.
3. For each cluster, set the new cluster center as the positional mean of all points in that cluster.
4. Repeat steps 2 and 3 until convergence, or a pre defined number of iterations, is reached.

This algorithm is presented visually in Figure 2.7.

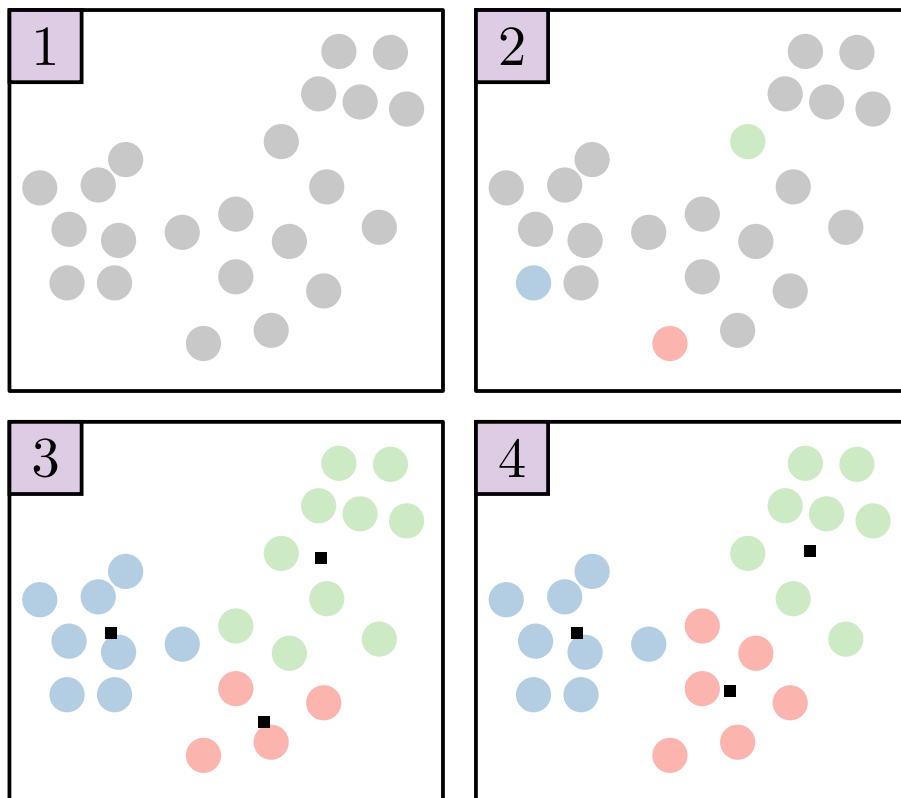


Figure 2.7.: Clustering a set of $n = 22$ points into $k = 3$ clusters using an algorithm which implements the k-means method. The steps 1-4 are described in the text.

Applying the k-means method on a space Ω results in a Voronoi tessellation of the space, having sharp boundaries between the clusters (Figure 2.8a). A slight change of position of a point can then result in it being assigned to a different cluster (Figure 2.8b).

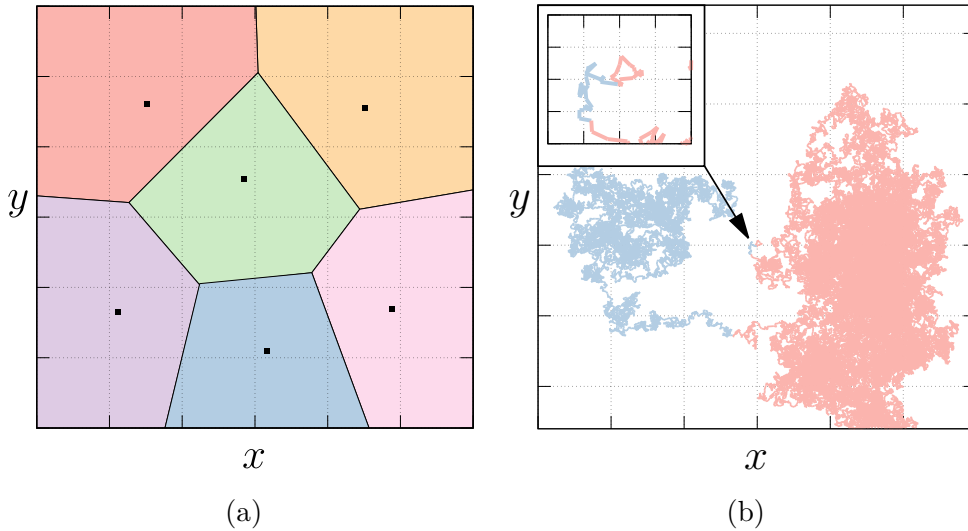


Figure 2.8.: The k-means method. Left: A two-dimensional space clustered into six clusters, their centers shown as black rectangles. Right: Clustering a trajectory into two clusters using the k-means method. Notice in the center of the figure (zoomed-in, top right) how some parts of the trajectory are clustered as belonging to the left set, while a human would probably cluster this part as belonging to the right set.

2.2. Brownian Dynamics

We wish to automate the above method of constructing Markov state models from protein trajectories and to optimize the methods employed in each step. To that aim, it is useful to compare the results of applying the method to some benchmark trajectories, which are expected to yield known results. These benchmark trajectories can be generated by simulating Brownian dynamics using potential landscapes that can be adjusted for the number of potential wells and their depth, and the height of the potential barriers between them. That way, trajectories with known number of metastable states and transition rates between them can be easily generated.

In Brownian dynamics, particles are modeled as being point-like with a distinct position, and a velocity that arises from two forces: a potential-derived force referred to as *drift force*, and *thermal noise*. The base equation of motion is[9]

$$\dot{\mathbf{x}}(t) = -\frac{D}{k_{\text{B}}T} \nabla U(\mathbf{x}) + \sqrt{2D}R(t), \quad (2.14)$$

2. Theoretical Foundation

where D is the diffusion coefficient, $k_{\text{B}}T$ is the Boltzmann constant, T is the temperature of the system and $R(t)$ is a Gaussian process with $\mu = 0, \sigma = 1$ and $\langle R(t)R(t') \rangle = \delta(t - t')$ - i.e. at any two times t and t' the values of R are uncorrelated.

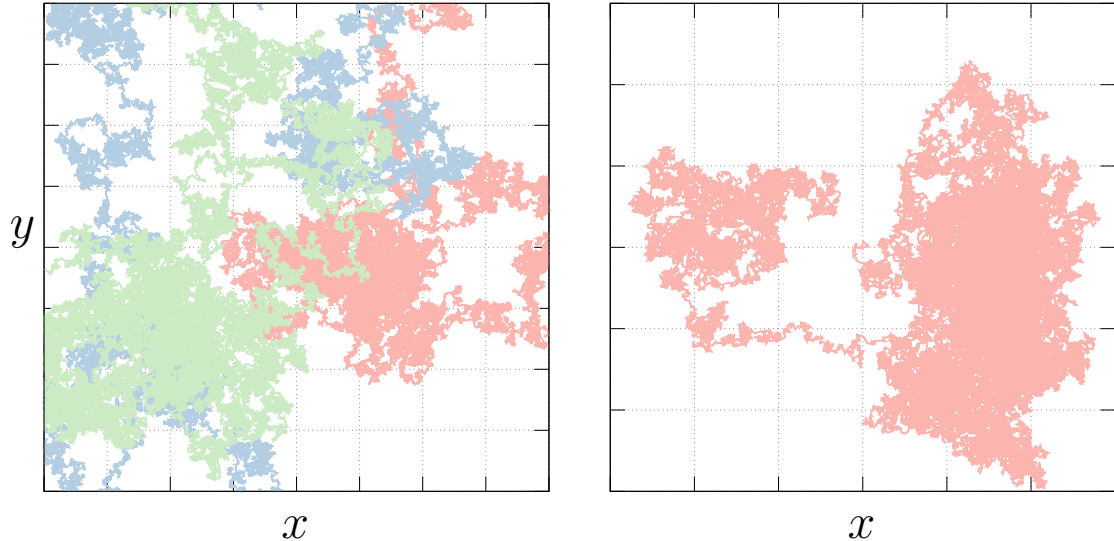


Figure 2.9.: Brownian dynamics trajectories in two dimensions. Left: three trajectories without a drift force. Right: one trajectory with a drift force that has two potentials wells.

2.2.1. Adjustable Potential Landscapes

Generating trajectories with known amount of metastable states and transition rates between these states using a Brownian dynamics simulation can be done by choosing potential landscapes that would yield such trajectories. One way to approach the generation of adjustable potentials is by "reverse-engineering" of the Boltzmann distribution of a system in a potential landscape $U(x)$. The Boltzmann distribution is given by[10]

$$p(x) = \frac{1}{Z} e^{-\frac{U(x)}{k_{\text{B}}T}}, \quad (2.15)$$

where k_{B} is the Boltzmann constant, T is the temperature of the system, and Z a normalization term called the partition function:

$$Z = \int_{-\infty}^{\infty} e^{-\frac{U(x)}{k_{\text{B}}T}} dx. \quad (2.16)$$

Isolating $U(x)$ from the equation yields*

$$U(x) = -k_B T \log(p(x)), \quad (2.17)$$

which shows that by a right choice of a desired equilibrium distribution, a potential that will yield the distribution can be constructed. A convenient choice of $p(x)$ is a sum of Gaussian functions of the form:

$$\mathcal{G}(x) = A e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (2.18)$$

since it is simple to set the center of the Gaussian via μ , its width via σ and its maximum height via A . In addition, these functions are relatively simple to derive and their integrals can be calculated analytically. Figure 2.10 shows an example of a probability distribution constructed from three such Gaussian functions, and the potential resulting by applying to it Equation 2.17.

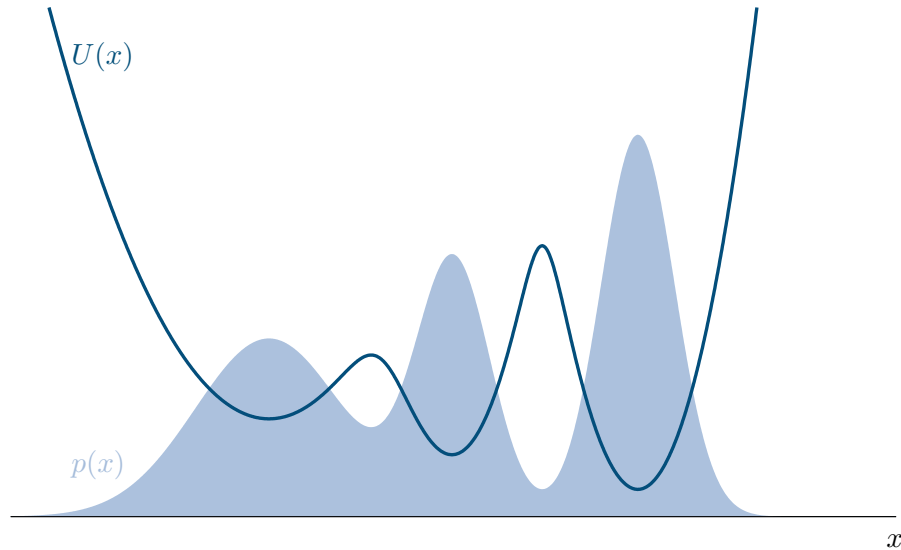


Figure 2.10.: A potential $U(x)$ constructed from a sum of three Gaussian functions, by the application of Equation 2.17, and the corresponding probability distribution $p(x)$.

Another property that makes sums of Gaussian functions desirable for constructing potential landscapes is that generalizing them to higher dimensions is fairly straight-forward: a general

*The partition function is absent from this expression since any constant addition to the entire potential does not affect the force derived from the potential.

2. Theoretical Foundation

N -dimensional Gaussian function is of the form

$$\mathcal{G}(x_1, x_2, \dots, x_n) = \prod_{d=1}^n \mathcal{G}(x_d), \quad (2.19)$$

which preserves many of the analytical properties of one-dimensional Gaussian functions due to the independence of its dimensional components.

2.2.2. Numerical Integration of the Equation of Motion

In order to simulate Brownian dynamics on a computer, a proper time discretization of Equation 2.14 is required. This can be done by approximating the motion of a Brownian particle by applying the linear solution of the Smoluchowski equation to small time steps.

The general one-dimensional Smoluchowski equation is of the form[11]

$$\frac{\partial}{\partial t} p(x, t | x_0, t_0) = D \left[\frac{\partial^2}{\partial x^2} - \beta \frac{\partial}{\partial x} F(x) \right] p(x, t | x_0, t_0). \quad (2.20)$$

For a constant potential $U(x) = c$, an analytical solution is known[11]:

$$p(x, t | x_i, t_i) = \frac{1}{\sqrt{4\pi D \Delta t}} \exp \left[-\frac{(x - x_i + D\beta c \Delta t)^2}{4\pi D \Delta t} \right], \quad (2.21)$$

where $\Delta t = t - t_i$. This is a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with

$$\begin{aligned} \mu &= x_i - D\beta c \Delta t, \\ \sigma^2 &= 2D\Delta t. \end{aligned} \quad (2.22)$$

Using small Δt , one can place a particle in position x_i and using Equation 2.21 choose a random number R from a Gaussian distribution with mean and variance as in equation 2.22 as the next position of the particle, i.e.

$$x_{i+1} = R = \mu + \sigma \bar{R}, \quad (2.23)$$

where \bar{R} is a random number from a normalized Gaussian distribution $\mathcal{N}(0, 1)$. Substituting

Equation 2.22 into 2.23 yields

$$x_{i+1} = x_i - D\beta c\Delta t + \sqrt{2D\Delta t}R, \quad (2.24)$$

and substituting $c = -\nabla U(x_i)$ and $\beta = \frac{1}{k_B T}$ yields the complete one dimensional discrete integration scheme for Brownian dynamics in a small time step Δt ,

$$x_{i+1} = x_i - \frac{D}{k_B T} \nabla U(x_i) \Delta t + \sqrt{2D\Delta t}R. \quad (2.25)$$

Equation 2.25 can be generalized to any number of dimensions by substituting the position x with a position vector \mathbf{x} and substituting R with a multivariate normal distribution \mathbf{R} :

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{D}{k_B T} \nabla U(\mathbf{x}_i) \Delta t + \sqrt{2D\Delta t}\mathbf{R}. \quad (2.26)$$

2.2.3. Properties of Brownian Systems

Some known analytical properties of Brownian systems will now be briefly discussed. These properties were used to validate the correctness of the simulation described in the thesis. The derivation of some properties shown here are found as appendices to the text.

Mean-Squared Displacement

For the case of a flat potential $U(x) = c$, i.e. when the drift is zero and thermal noise is the only force present in the system, the mean-squared displacement (MSD) of an ensemble of non-interacting Brownian particles each with a starting positions x_0 is linear with time, proportional to two times the diffusing constant D (Figure 2.11):

$$\langle (x(t) - x_0)_i^2 \rangle = 2Dt. \quad (2.27)$$

When the particles are allowed to diffuse in N -dimensions, the coefficient 2 becomes $2N$, yielding[12]

$$\langle (\mathbf{x}(t) - \mathbf{x}_0)_i^2 \rangle = 2NDt. \quad (2.28)$$

2. Theoretical Foundation

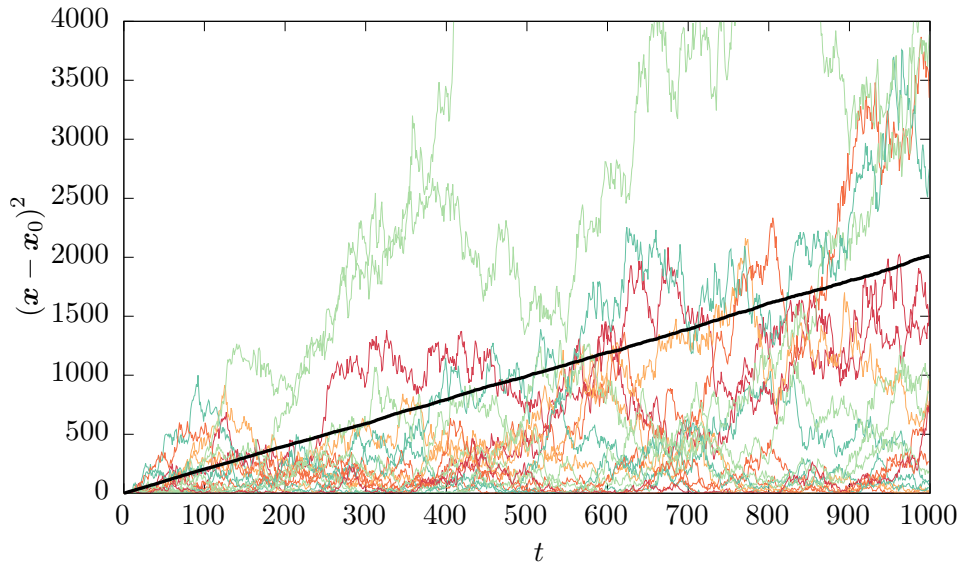


Figure 2.11.: 1000 non-interacting Brownian particles diffusing without a drift force in one dimension. The squared displacements of 10 of these particles are shown as colored lines. The mean-squared displacement (MSD) of the entire ensemble is shown as a black line.

Equilibrium Distribution

As discussed in Section 2.2.1, when a drift force derived from a potential $U(x)$ is present in the system, the probability distribution of finding a particle at position x in equilibrium is given by the Boltzmann distribution (Equation 2.15). Thus, the probability density of finding the particle in the closed interval $[a, b]$ is

$$P_{[a,b]} = \int_a^b e^{-\frac{U(x)}{k_B T}} dx. \quad (2.29)$$

Substituting $U(x)$ for a potential of the form discussed in Section 2.2.1 yields

$$P_{[a,b]} = \frac{1}{\sqrt{2\pi}} \sum_i A_i \sigma_i \left[\operatorname{erf} \left(\frac{b - \mu_i}{\sqrt{2}\sigma_i} \right) - \operatorname{erf} \left(\frac{a - \mu_i}{\sqrt{2}\sigma_i} \right) \right]. \quad (2.30)$$

Since the different dimensional components of an N -dimensional potential of a log-Gaussian form (the potentials described in Section 2.2.1) are independent, the general N -dimensional integral over an interval $I = I_1 \times I_2 \times \cdots \times I_n = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N]$ is a product

of the respective one-dimensional integrals; i.e. for a single Gaussian function

$$\begin{aligned} P_I &= \prod_{d=1}^N \int_{I_d} \mathcal{G}^d(x_d) dx_d \\ &= (2\pi)^{-\frac{N}{2}} \prod_{d=1}^N A_d \sigma_d \left[\operatorname{erf} \left(\frac{b_d - \mu_d}{\sqrt{2}\sigma_d} \right) - \operatorname{erf} \left(\frac{a_d - \mu_d}{\sqrt{2}\sigma_d} \right) \right], \end{aligned} \quad (2.31)$$

and for a generalized potential composed of m such Gaussian functions:

$$P_I = (2\pi)^{-\frac{N}{2}} \sum_{i=1}^m \prod_{d=1}^N A_{i,d} \sigma_{i,d} \left[\operatorname{erf} \left(\frac{b_d - \mu_{i,d}}{\sqrt{2}\sigma_{i,d}} \right) - \operatorname{erf} \left(\frac{a_d - \mu_{i,d}}{\sqrt{2}\sigma_{i,d}} \right) \right]. \quad (2.32)$$

Ornstein-Uhlenbeck Process

When the drift potential is an harmonic potential $U(x) = \frac{1}{2}kx^2$, the system can be described as an Ornstein-Uhlenbeck process, where the distribution of positions of an ensemble of particles with the same starting point x_0 is a Gaussian of the form[11]

$$p(x, t | x_0, t_0) = \mathcal{N}(\mu(t), \sigma(t)), \quad (2.33)$$

where

$$\begin{aligned} \mu(t) &= x_0 e^{-D\beta kt} \\ \sigma^2(t) &= \frac{1}{\beta k} (1 - e^{-2D\beta kt}), \end{aligned} \quad (2.34)$$

and $\beta = \frac{1}{k_B T}$. Figure 2.12 depicts the change over time of the mean and variance in position for several ensembles of particles.

A harmonic potential can be constructed from a log-Gaussian potential by using only one Gaussian function with $\mu = 0$:

$$U(x) = -k_B T \log \left(A e^{-\frac{x^2}{2\sigma^2}} \right) = k_B T \frac{x^2}{2\sigma^2}. \quad (2.35)$$

2. Theoretical Foundation

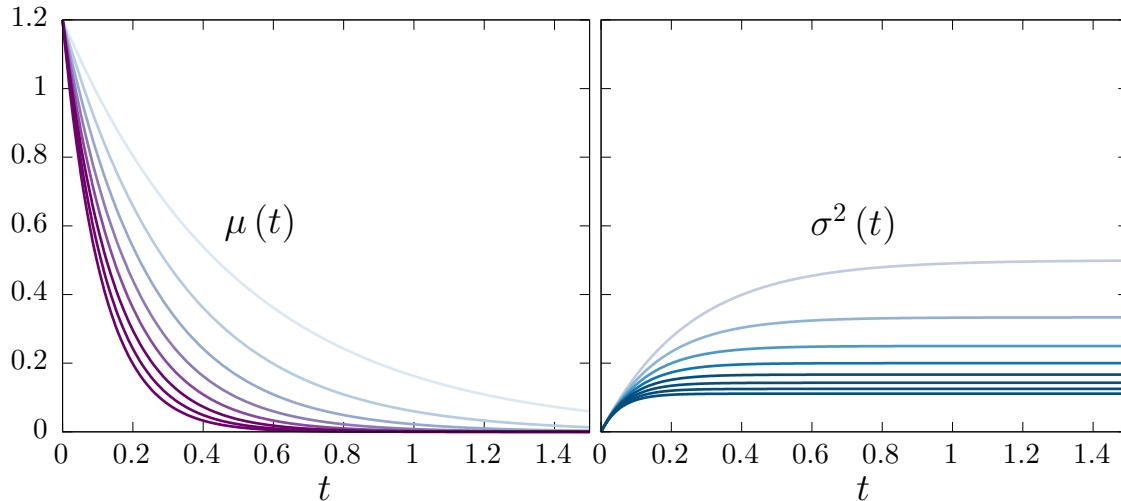


Figure 2.12.: Mean and variance of the positions of several ensembles of particles diffusing in an harmonic potential $U(x) = \frac{1}{2}kx^2$, with darker colors indicating increasing values of the product $D\beta k$.

Equating to a standard harmonic potential $U(x) = \frac{1}{2}kx^2$ and solving for k yields

$$k = \frac{k_{\text{B}}T}{\sigma^2} = \frac{1}{\beta\sigma^2}, \quad (2.36)$$

meaning that the effective harmonic coefficient k is governed by both the temperature and the variance of the single Gaussian used to construct the potential.

Kramer's Approximation

The rate at which a Brownian particle escapes a potential well is depended on the shape of both the well, the energy barrier and their relative heights (see Figure 2.13). This system has two distinct time scales:

1. The equilibration time τ_T within the well, i.e. the convergence time to the Boltzmann distribution assuming the barrier is infinite.
2. The escape time τ_E , i.e. the mean time it takes the particle to go from the well through a barrier and to a neighboring well (in Figure 2.13 at $x = C$).

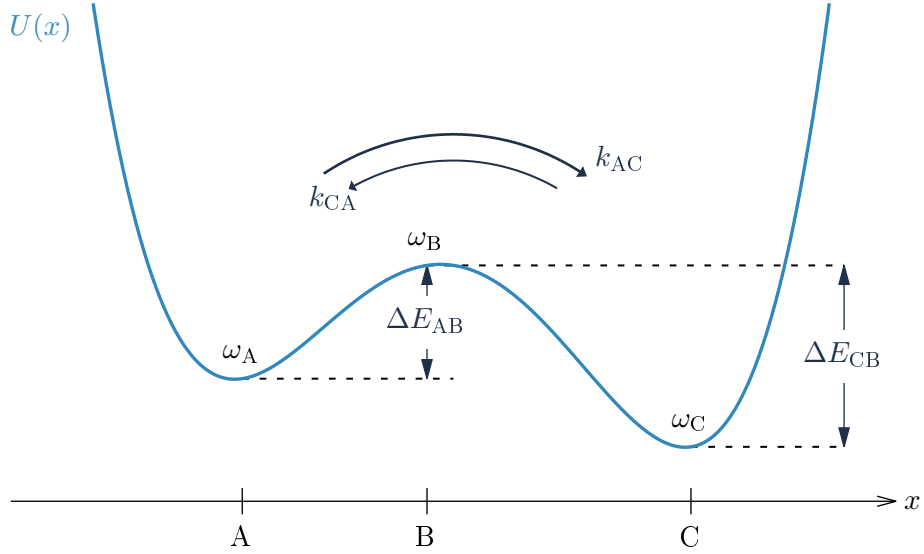


Figure 2.13.: Two potential wells of different depth at $x = A, C$, with a barrier at $x = B$. The barrier height is E_{AB} relative to the well at $x = A$ and E_{CB} relative to the well at $x = C$. Redrawn from [11].

For systems in which $\tau_E \gg \tau_T$, the barrier must be much greater than $k_B T$, i.e.

$$\Delta E_{AB} \gg k_B T, \quad (2.37)$$

otherwise the transition occurs at the same timescale as the equilibration inside the well.

When the particle has total energy smaller than ΔE_{AB} , it oscillates inside the well A with frequency ω_A . When its total energy is bigger than ΔE_{AB} , the exchange between kinetic and potential energy is characterized by an oscillation with frequency ω_B . These frequencies depend on the shape of the well (or the barrier), and the mass of the particle:

$$\begin{aligned} \omega_A &= \sqrt{\frac{U''(x=A)}{m}}, \\ \omega_B &= \sqrt{\left| \frac{U''(x=B)}{m} \right|}. \end{aligned} \quad (2.38)$$

In the simulation presented in the thesis, friction is dominant, i.e. $\frac{\gamma}{m} \gg \omega_B$, and thus the

2. Theoretical Foundation

transition rate $k_{A \rightarrow C}$ is approximated by[11]

$$k_{A \rightarrow C} = \frac{m\omega_A\omega_B}{2\pi\gamma} e^{-\frac{\Delta E_{AB}}{k_B T}}. \quad (2.39)$$

3. Methods

3.1. Brownian Dynamics Simulation

3.1.1. Process

The program uses the integration scheme in Equation 2.25 to discretize the Brownian dynamics Equation of motion (Equation 2.14). The user feeds the simulation with the following simulation parameters: number of particles n , number of dimensions N , initial positions $\{\mathbf{x}_i\}$, diffusion coefficient D , temperature β , number of steps τ and time step size Δt .

In addition, the user specifies the potential $U(\mathbf{x})$ by passing the program a list of parameters for N -dimensional Gaussian functions. The parameters are (for each Gaussian) the amplitude \mathbf{A} , center $\boldsymbol{\mu}$ and width $\boldsymbol{\sigma}$ of each N -dimensional Gaussian.

The program returns a **NumPy ndarray** of dimensions $\tau \times N \times n$, which represents the entire N -dimensional trajectories of all the particles.

A general scheme of the program is shown in Figure 3.1. The code for the program can be found in Appendix A.

3.1.2. Validation

Before using the program in further steps, it had to be validated. The validation was done by comparing the programs output to four known analytical properties of Brownian systems, discussed in Section 2.2.3. The validation tests cover both thermodynamic and kinetic properties of Brownian systems.

3. Methods

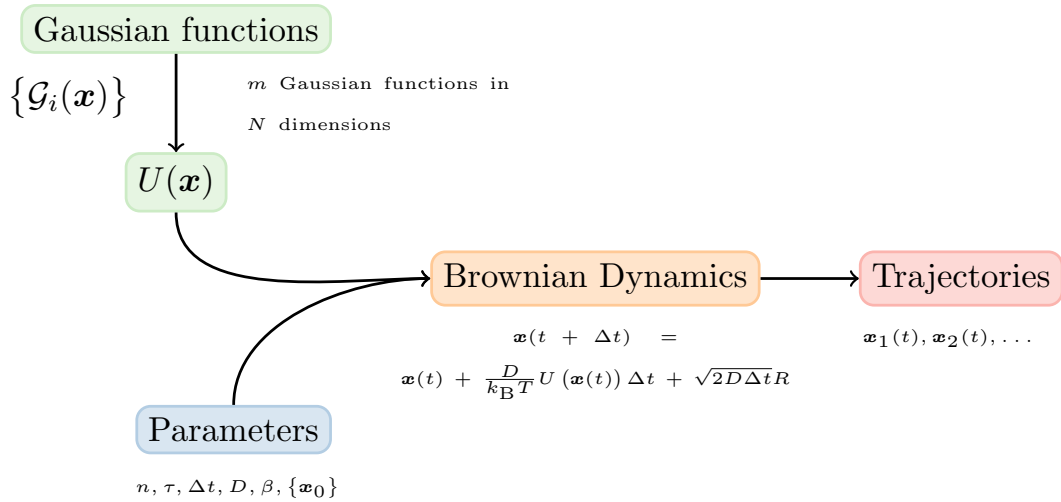


Figure 3.1.: Simulation flow: a set of N -dimensional Gaussian functions and other parameters, such as the number of particles n , time step Δt , temperature via $k_B T$, the diffusion coefficient D and starting positions $\{\mathbf{x}_0\}$, are fed into the simulation. After performing τ simulation steps using the integration method (Equation 2.26), the results are output as a **NumPy ndarray**.

Mean-Squared Displacement

Validation of the thermal noise component of the force was done by simulating an ensemble of 10^4 non-interacting particles for 500 time steps in a flat potential, calculating for each time step the squared displacement from the starting step \mathbf{x}_0 for each particle, and taking the mean of these squared displacements:

$$\text{MSD} = \langle (\mathbf{x}(t) - \mathbf{x}_0)^2 \rangle. \quad (3.1)$$

A linear regression of the ensemble mean-squared displacements vs. t was then performed using one standard deviation in the ensemble square displacements as the error in each time step. The resulting fit coefficients were recorded, and the process was repeated for 100 different values of the diffusion coefficient uniformly distributed between $D = 1$ and $D = 5$.

The slope and error from each linear regression were used as inputs for a final regression, yielding the relation coefficients between the mean-squared displacement and the diffusion coefficient D . This relation was then compared to theory (Equation 2.28).

Table 3.1.: Parameters of the different potentials used in the validation of the equilibrium distribution of the simulation.

Description	Amplitudes	Positions	Widths
One-dimensional Potentials			
Single well	1	0	1
Double well	1,1	-3, 3	1, 1
Two wells	2, 1.5	-3, 2	1, 2
Three wells	2, 0.7, 1	-5, 0, 5	0.7, 1.5, 1
Two-dimensional Potentials			
Single well	1,1	0,0	1,1

Equilibrium Distribution

Next, the simulation was tested to check whether it converged to the expected equilibrium distribution (the Boltzmann distribution, Equation 2.15). Six simulations were run, each with a different potential, $k_{\text{B}}T = 1$, $D = 1$, and other parameters as summarized in Tables 3.1 and 3.2.

Analyzing each simulation was done following these steps:

1. Calculating a positional histogram for each particle in the simulation domain, for the last N steps and using a number of bins as in Table 3.2.
2. Calculating for each bin the mean number of counts μ and their variance.
3. Estimating the standard error by $\sigma_{\bar{x}} = \sqrt{\frac{\text{Var}}{n}}$, where n is the number of particles.
4. Counting the percentage of bins where the theoretical mean counts is within $\mu \pm \sigma_{\bar{x}}$.

Ornstein-Uhlenbeck Process

A simple Ornstein-Uhlenbeck process was used as a validation test for the kinetic behavior of the simulation. Within the program, constructing a one-dimensional harmonic potential can be done by using a single Gaussian function. Choosing $\mu = 0$ yields an harmonic

3. Methods

Table 3.2.: Parameters of the different simulations used in the validation of the equilibrium distribution of the simulation. N is the number of steps, n is the number of particles, I is the histogram domain and b is the number of bins within I .

Description	N	Δt	n	I	b
One-dimensional Potentials					
Single well	10^4	10^{-2}	5×10^3	$[-4, 4]$	200
Symmetric double well				$[-8, 8]$	
Two wells	2.5×10^4	10^{-3}	10^3	$[-7, 5]$	400
Three wells				$[-9, 7]$	
Two-dimensional Potentials					
Single well	3×10^5	10^{-3}	50	$[-3, 3] \times [-3, 3]$	100×100
Two wells				$[-6, 6] \times [-6, 6]$	

potential

$$U(x) = \frac{k_B T}{2\sigma^2} x^2, \quad (3.2)$$

where the harmonic coefficient is $k = \frac{k_B T}{\sigma^2}$. Setting $k = 1$ yields

$$\sigma = \sqrt{k_B T} = \frac{1}{\sqrt{\beta}}. \quad (3.3)$$

This means that in order to set $k = 1$ one needs to set the width of the single Gaussian function to be $\sigma = \sqrt{k_B T}$.

Using the above method, 10 sets of simulations were carried out, each set of simulations was given a different temperature $\frac{1}{k_B T} = \beta \in \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5\}$, and a harmonic potential corresponding to β . Each set simulated 15 ensembles of non-interacting particles, composed of between 10 and 5×10^3 particles each, all starting at $x_0 = 1.5$. These ensembles were then simulated for 5×10^3 with a step $\Delta t = 0.001$.

The ensemble mean and variance over time for all sets were recorded. In order to check that no systematic errors affect the system during relaxation, a time interval between far enough from the start of the simulation on one hand, and from equilibrium on the other hand, was chosen for each set (Table 3.3). The root mean-squared deviation (RMSD) between the ensemble mean and theoretical mean (Equation 2.34) was then calculated, and the resulting relations between the RMSD and the number of particles N for each set were then compared

Table 3.3.: Ornstein-Uhlenbeck process: time intervals for calculating RMSD between the ensemble mean and theoretical mean against the number of particles N , for each set of simulations.

β	Time interval
0.25	[3000, 4000]
0.5	[3000, 4000]
0.75	[2000, 3000]
1	[500, 1500]
1.25	[500, 1500]
1.5	[500, 1500]
1.75	[500, 1500]
2	[500, 1000]
2.25	[500, 1000]
2.5	[500, 1000]

to $\frac{1}{\sqrt{N}}$, the expected convergence when no systematic errors are present in the system.

Kramer's Approximation

Simulating a system that conforms to Kramer's approximation was done by generating a log-Gaussian potential utilizing two Gaussian functions with $\mu = \pm M$ (Figure 3.2). This configuration yields a barrier height

$$\Delta E = \frac{M^2}{2} - \log(2) + \log(1 - e^{-2M^2}), \quad (3.4)$$

and second derivatives

$$\begin{aligned} \omega_A &= \sqrt{1 - \left(\frac{M}{\cosh(M^2)}\right)^2}, \\ \omega_B &= \sqrt{|1 - M^2|}. \end{aligned} \quad (3.5)$$

(the derivation of these quantities is found in Appendix B)

For each value $M \in \{3.5, 3.51, 3.52, \dots, 4.0\}$, a single particle positioned at $x_0 = -M$ was

3. Methods

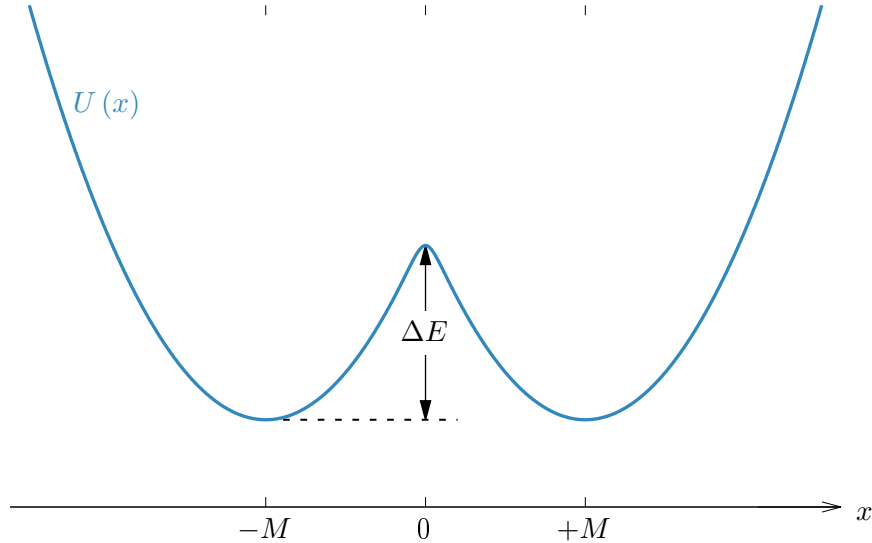


Figure 3.2.: Kramer’s theorem from log-Gaussian potentials: a potential derived from two Gaussian functions with $A = 1$, $\sigma = 1$ and $\mu = \pm M$ is a symmetric double-well potential.

simulated for a number of steps that were expected to yield ≈ 50 transitions according to Kramer’s theorem. The following parameters were used: $D\Delta t = 0.001$, $k_B T = 1$.

Each trajectory was then coarse-grained by keeping each 1000th step, and was analyzed using *PyEMMA*[13] with the following steps:

1. The k-mean method was used to cluster the trajectory into two clusters.
2. The lag time was chosen from a uniformly distributed number of steps between 5 and 250, corresponding to the trajectory length.
3. Using *PyEMMA*’s `pyemma.msm.bayesian_markov_model` function, a MSM was created from each clustered trajectory, and to get an estimation of the timescale of the crossing process by use of the function `bayesian_msm.sample_mean` and `bayesian_msm.sample_conf` for the confidence interval (68% confidence).
4. The most likely transition rate k_{\max} and the confidence interval $[k^-, k^+]$ were calculated as the inverse of the timescale, and the confidence values of the time scales, respectively.

The trajectories were then modified by removing any transition that lasted less than 100 steps (corresponding to $\Delta t = 0.1$) and were analyzed using the same method as above.

3.2. Constructing Markov State Models from Simulated Protein Trajectories

Next, the trajectories (with the short-lasting transitions removed) were used to estimate the transition rates by a Bayesian method:

1. The most likely transition time was estimated as $k_{\max} = \frac{1}{\langle \Delta t_i \rangle}$, where $\{\Delta t_i\}$ are the periods of time the particle spent in one well before transitioning to the next well (Equation 2.12).
2. The confidence interval $[k^-, k^+]$ for the value of k_{\max} was calculated numerically via **SciPy**'s `integrate.quad` function, using 1000 values distributed logarithmically between $[0, k_{\max}]$ and $[k_{\max}, 5 \cdot k_{\max}]$ and solving the following equations for k^-, k^+ :

$$\int_{k^-}^{k_{\max}} p(k | \{\Delta t_i\}) dk = 0.68 \int_0^{k_{\max}} p(k | \{\Delta t_i\}) dk \quad (3.6)$$

$$\int_{k_{\max}}^{k^+} p(k | \{\Delta t_i\}) dk = 0.68 \int_{k_{\max}}^{5 \cdot k_{\max}} p(k | \{\Delta t_i\}) dk. \quad (3.7)$$

3. The values k_{\max}, k^- and k^+ were recorded.

The resulting transition rates and their respective confidence intervals, using both the PyEMMA estimation method and the Bayesian method, were plotted against M , and compared to the expected transition rates according to Kramer's approximation.

3.2. Constructing Markov State Models from Simulated Protein Trajectories

A brief graphical overview of the general method used in our research group to generate Markov state models from molecular dynamics trajectories is presented in Figure 3.3.

3.2.1. Analyzing a 2-Dimensional Trajectory Generated by Brownian Dynamics Simulation

A two-dimensional potential was constructed from four Gaussian functions with parameters $\boldsymbol{\mu}_1 = (-6, -6)$, $\boldsymbol{\mu}_2 = (0, 0)$, $\boldsymbol{\mu}_3 = (1, 5)$, $\boldsymbol{\mu}_4 = (5, 1)$ and $\boldsymbol{\sigma}_1 = (1.5, 1.5)$, $\boldsymbol{\sigma}_2 = \boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_4 =$

3. Methods

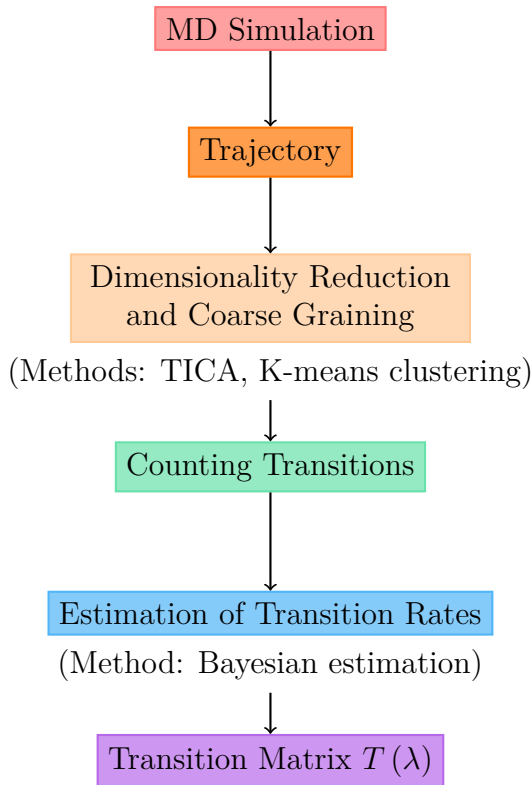


Figure 3.3.: General steps in the construction of Markov state models from protein trajectories used in our research group.

(1, 1) (See Figure 3.4). A single particle starting at $\mathbf{x}_0 = (0, 0)$ was allowed to diffuse for 10^7 steps of $\Delta t = 0.01$, and with the parameters $D = 1, k_B T = 1$.

The analysis of the trajectory was done using the **PyEMMA** library [13]. The trajectory was grouped into four clusters using the k-means method via `pyemma.coordinates.cluster_kmeans` and the number of trajectory points in each cluster was counted, and the ratios between all clusters computed. Using `pyemma.plots.plot_free_energy`, a free energy estimation for the system was calculated and plotted..

The implied timescales for the slowest transition process were calculated via the function `pyemma.msm.its` for 20 time lags uniformly distributed over $[1, 700]$. A lag time $\tau = 75$ was chosen, and the mean timescale with its confidence intervals were calculated by generating a MSM from the clustered trajectory for lag time τ (via the functions `pyemma.msm.bayesian_markov_model`, `bayesian_msm.sample_mean` and `bayesian_msm.sample_conf`, respectively).

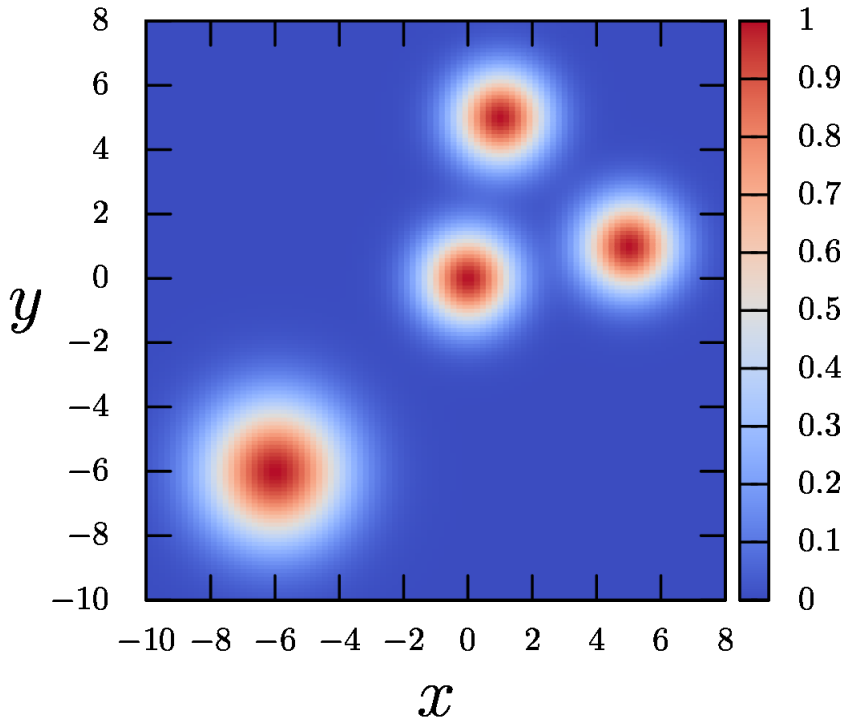


Figure 3.4.: Generating trajectories: the four Gaussian functions that were used to construct a two-dimensional potential for the validation of the MSM estimation method. The color indicates the value of the sum of Gaussian functions, $\sum_{i=1}^4 \mathcal{G}_i(x, y)$.

The right eigenvectors and eigenvalues of the Markov state model were calculated via `msm.eigenvectors_right`, an estimation of the model was generated via `pyemma.msm.estimate_markov_model` and the model was validated for conforming to the Chapman-Kolmogorov equation (Equation 2.3) via `pyemma.msm.plot_cktest`.

3.2.2. 2-Dimensional Trajectory Cut to a Single Transition

The trajectory generated in the previous step was cut to the time interval $[0, 3.73 \times 10^3]$, such that only one transition forward and one transition back were observed for the slowest transition process. The trajectory was clustered with the same cluster centers as for the full trajectory, and the implied timescales were then calculated for the same lag times as for the full trajectory. Short-lived transitions were then removed from the cut clustered trajectory, and the implied timescales calculated again in the exact same way as before. The mean timescale for this process, along with its confidence intervals,

3. *Methods*

were calculated by generating a MSM from the clustered trajectory for lag time τ (via the functions `pyemma.msm.bayesian_markov_model`, `bayesian_msm.sample_mean` and `bayesian_msm.sample_conf`, respectively).

4. Results and Discussion

The results are given in two parts: the first part discusses the validation of the simulation written as part of this thesis. The second part presents one example of applying the Markov state model formation method to a trajectory generated by the simulation, and a case of limiting the trajectory to a single transition of its slowest transition process.

4.1. Simulation Validation

Each of the four known analytical properties of Brownian systems discussed in Section 2.2.3 were used to validate the simulation. These validation tests checked both the thermodynamic and kinetic behaviour of the simulation, and ensure that each of its components conform to the known properties of Brownian dynamics.

4.1.1. Mean-Squared Displacement

The mean-squared displacement (MSD) validation test was used to determine whether the thermal noise force term in the simulation behaves according to theory. The simulation calculates this force in each step as

$$F_{\text{Noise}} = \sqrt{2D\Delta t}\mathbf{R}, \quad (4.1)$$

where the diffusion coefficient D and the time step Δt are given as inputs to the program, and \mathbf{R} is a random number generated at each step from a uniform Gaussian distribution.

Figure 4.1 shows the MSDs over time for four simulations (out of 100), where the non-interacting particles experience only thermal noise and no drift and were diffusing in three dimensions. It can be seen that the MSD for each time step in each of these simulations is much closer to its theoretical value of $6Dt$ than one standard deviation (bright blue area).

4. Results and Discussion

This was consistent throughout all 100 simulations.

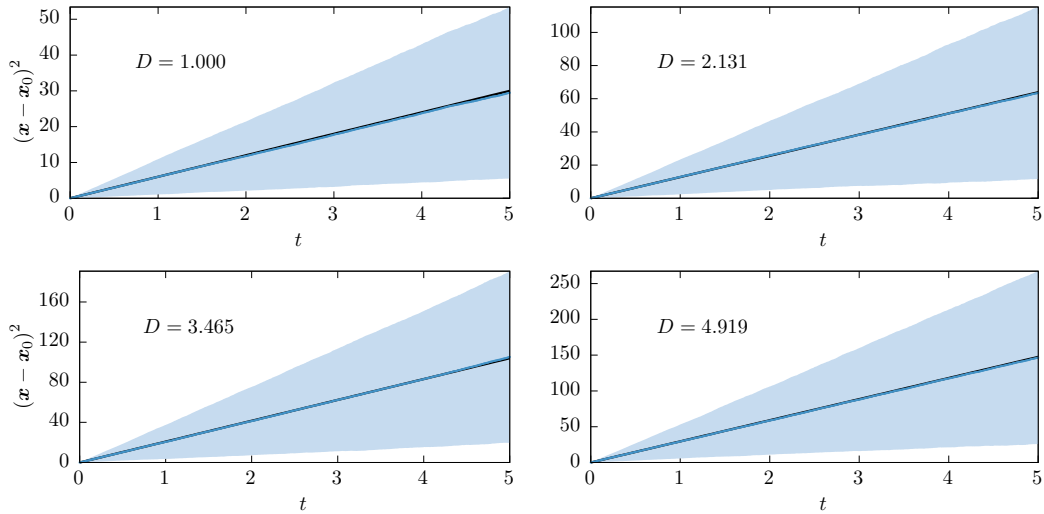


Figure 4.1.: The mean-squared displacement over time for 4 ensembles, each of 10^4 non-interacting particles diffusing in a three-dimensional flat potential. The diffusion coefficient D of each ensemble is given in the plot. The black line in each plot represents the theoretical MSD over time, the blue line the ensemble MSD and the bright blue area is one standard deviation from the ensemble MSD.

The calculated slopes also agree with the theory: for example, for the simulation with $D = 1$, the regression calculated slope is 5.9 ± 0.2 , with the expected slope being exactly 6. The y -intercept is calculated by the regression to be 0.002 ± 0.04 , which is also within the range of its expected value of zero. This match was observed across all 100 simulations.

To extract back from the simulation the ratio between the slope of the MSD and the diffusion coefficient D , all calculated slopes and their errors were plotted against the respective diffusion coefficients in each simulation (Figure 4.2). This yielded a ratio of 5.98 ± 0.05 , which is in agreement with the theoretical value of 6 for three dimensions.

It can also be seen in Figure 4.2 that the errors in the slopes grow with the diffusion coefficient: this is expected since a higher diffusion coefficient causes the positional variance of the ensemble to spread faster.

The results of this test demonstrate that within a statistical error, the noise term of the force in the simulation indeed conforms to theory.

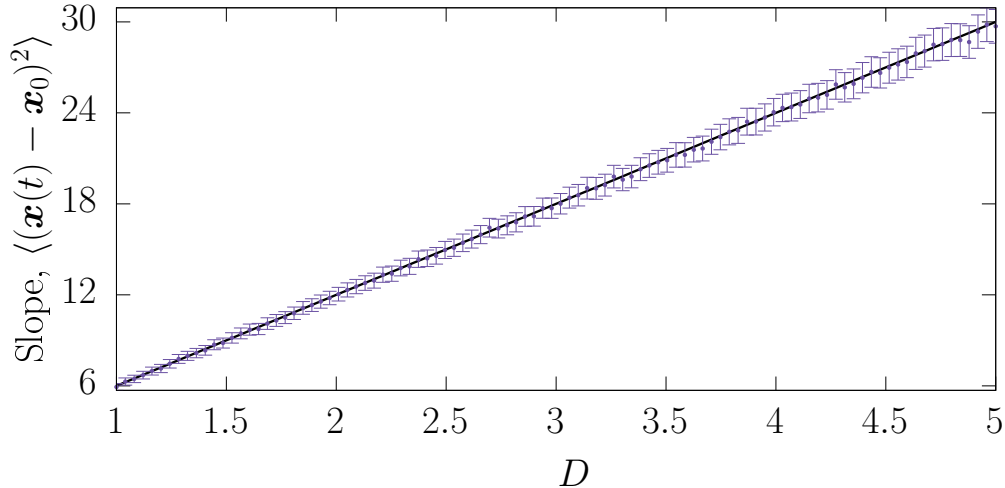


Figure 4.2.: Slope of the mean-squared displacement over time plotted against the diffusion coefficient D for 100 simulations, each of 10^4 non interacting particles in a three dimensional flat potential. The theoretical value $\langle (\mathbf{x}(t) - \mathbf{x}_0)^2 \rangle = 6Dt$ is shown as a black line. The resulting ratio of slopes to D is 5.98 ± 0.05 .

4.1.2. Equilibrium Distribution

The tests done to validate the drift contribution in the Brownian motion employed potentials in one- and two-dimensions, constructed from varying amounts of Gaussian functions with different amplitudes, means and variances. This was done in order to test that all the parameters defining the potentials affect the simulations in the correct way.

The systems were run for a length of time that allowed them to reach equilibrium. This length of time was calculated using Kramer's approximation: for each potential, the mean escape time from a potential well over the highest potential barrier was calculated, and the simulation ran for at least 10 times longer. This ensured that each particle, no matter its starting point, would have had enough time to escape a well and diffuse to other areas of the simulation domain.

A histogram of each particle's trajectory for the last N steps (summarized in Table 3.2) was then generated, and for each histogram bin the mean and variance of the counts from all particles were calculated. These quantities were then compared to the expected mean counts according to the Boltzmann distribution, by multiplying the probability distribution in the histogram bin with the number of steps N . The results for the one-dimensional

4. Results and Discussion

potentials can be seen in Figure 4.3.

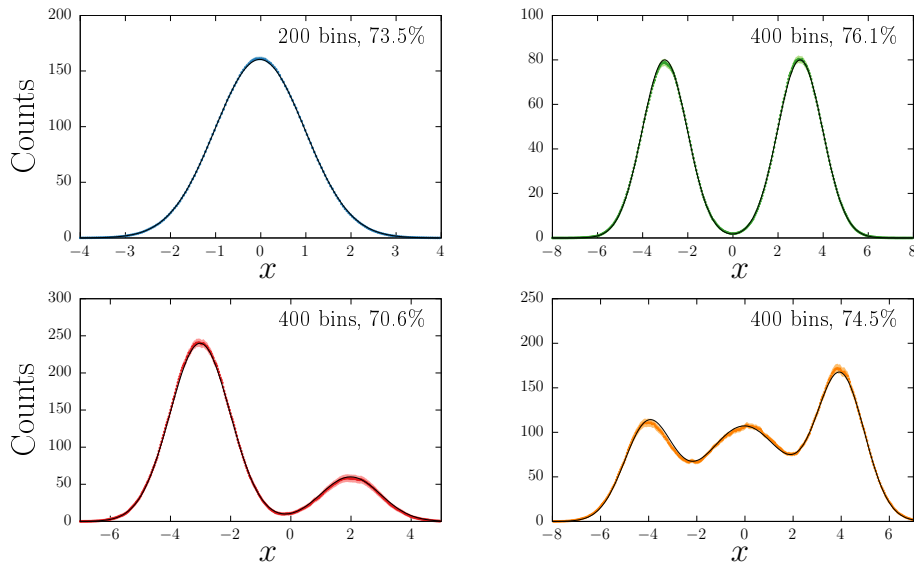


Figure 4.3.: Ensemble positional histograms for the one-dimensional simulations listed in Table 3.2. The black dots represent the theoretical mean counts in each histogram bin, the colored area represents a 1σ deviation from the simulated ensemble mean position in each histogram bin, exaggerated graphically so that it can be seen in the figure. Each simulation figure also displays the number of histogram bins used in the histogram, and the percent of the histogram bins for which the ensemble mean is within one standard deviation from the theoretical mean.

These results show that for each one-dimensional potential, the simulated system in equilibrium was for at least 70% of the histogram bins within a single standard deviation from the theoretical mean, measured by the standard error. This in turn means that the only source of deviation from the theory is statistical noise, otherwise we would expect to see less than 68% match.

The same is true for the two-dimensional simulations, the results of which can be seen in Figure 4.4. In the case of the single well simulation, the ensemble mean was within the standard error of the theoretical mean in 68.8% of the histogram bins. For the double well simulation, the percentage was a bit higher, at 70.4%.

Contrary to the one-dimensional simulations, the two-dimensional simulations used a smaller number of particles (on the order of magnitude of several tens of particle, compared to 10^3 particles for the one-dimensional simulations), and were run for longer times. This

was due to a limitation arising from the calculation of the standard error: increasing the number of particles n decreases the standard error as a function of $\frac{1}{\sqrt{n}}$, but only increases the convergence to the theoretical equilibrium distribution up to a limit, which is dictated by the time step Δt . Thus, if the equilibration takes longer (and due to computational time constraints a smaller time step is not desirable) the number of particles must be lowered.

This is evident in the complete standard error form:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}} = \sqrt{\frac{1}{(n-1)(N\Delta t-1)} \sum_{i=1}^n (x_i - \bar{x})^2}, \quad (4.2)$$

where n is the number of particles, Δt the time step and N the number of time steps. Since the number of particles and the product $N\Delta t$ affect the standard error in the same way, an increase in N , if not decreasing Δt , must be balanced by a decrease in n .

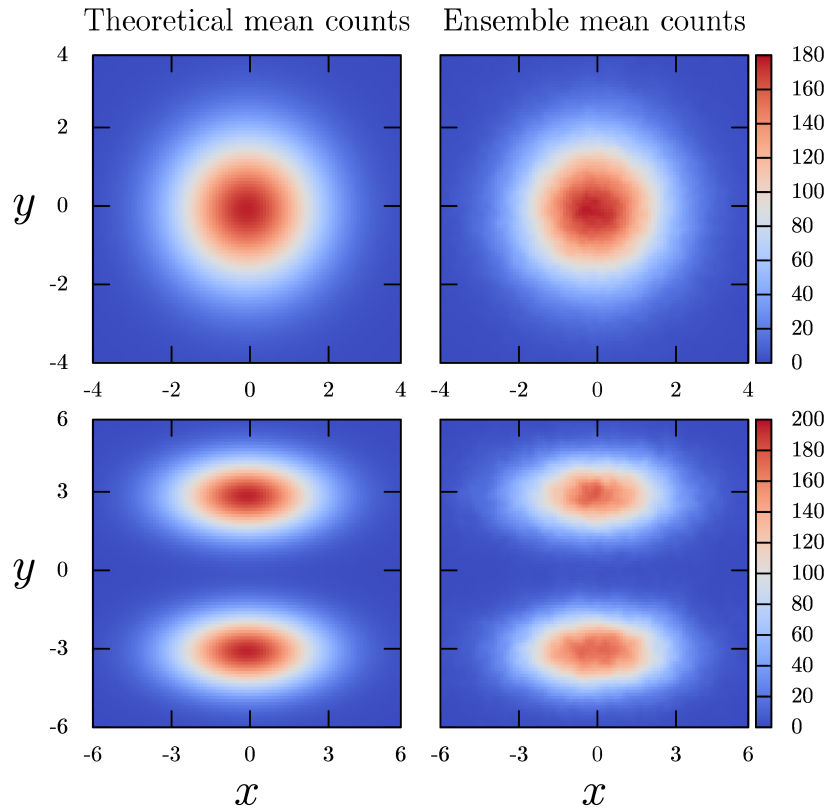


Figure 4.4.: Two-dimensional equilibrium distributions: expected mean counts on the left, ensemble mean counts on the right. Number of histogram bins and domains are specified in Table 3.2

The overall results for this validation test confirm that the simulation converges to the

4. Results and Discussion

expected equilibrium distribution in one and two dimensions.

4.1.3. Ornstein-Uhlenbeck Process

The Ornstein-Uhlenbeck process, in which a particle is diffusing in an harmonic potential, is one of the cases for which a complete analytical solution for the distribution of the particle's position over time is known. Therefore, it was chosen as a validation test for the kinetic behaviour of the simulation.

The mean and variance of 10 ensembles of 10^4 particles diffusing in harmonic potentials, each ensemble experiencing a different temperature from $\frac{1}{k_B T} = \beta \in \{0.25, 0.5, \dots, 2.25, 2.5\}$, is shown in Figure 4.5 together with their theoretical values according to Equation 2.34. The system was tested for different temperatures since this is the main contributing reason for the rate of relaxation in both the ensemble mean and variance, except the diffusion coefficient. Since the diffusion coefficient was already validated (Section 4.1.1), it was unnecessary to validate it again.

Figure 4.5 shows that qualitatively both the ensemble means and ensemble variances follow their respective theoretical curve. However, it is important to verify that the agreement is not only qualitative, but that the only deviations from theory occur due to statistical noise, and have no other source. This was done by varying the amount of particles for each value of β , keeping all other parameters the same. Figure 4.6 shows for each value of β the corresponding relation between the number of particles and the root mean-squared deviation (RMSD) from theory, for a time interval that is dependent on the temperature (See Table 3.3).

The reason for using different time intervals and not the entire trajectory was two folds: first, since all trajectories started from an exact location $x_0 = 1.5$, the start of each simulation is a period where the ensemble had a very short time to "spread out" and increase its variance. This can cause a bias in the resulting RMSD, shifting it to a lower value. Second, using a period in which the ensemble already equilibrated missed the point of this validation, which is to test the kinetic behaviour of the simulation: after all we already know that in equilibrium the simulation agrees with the theory (as discussed in Section 4.1.2). Therefore, in each set of simulations the time interval was chosen such that the system had enough time to diverge from the exact values given, but also it had not yet reached equilibrium.

If indeed the only source of the deviations is statistical noise, we expect the RMSD to converge with the number of particles N as $\frac{1}{\sqrt{N}}$. This is translated in a log-log plot as a linear correlation with slope $m = -\frac{1}{2}$. Figure 4.7 shows all the resulting slopes of the log-log linear regressions for all systems. It demonstrates that within a statistical error all systems indeed converged to the theoretical ensemble mean values with the correct slope, which is a strong indication that there are no systematic errors present in the system.

Moreover, the convergence slope indicates that in order to get a desirable accuracy in kinetic behaviour, a large enough ensemble size should be chosen. This is consistent with the expected behaviour of a numerical simulation.

4. Results and Discussion

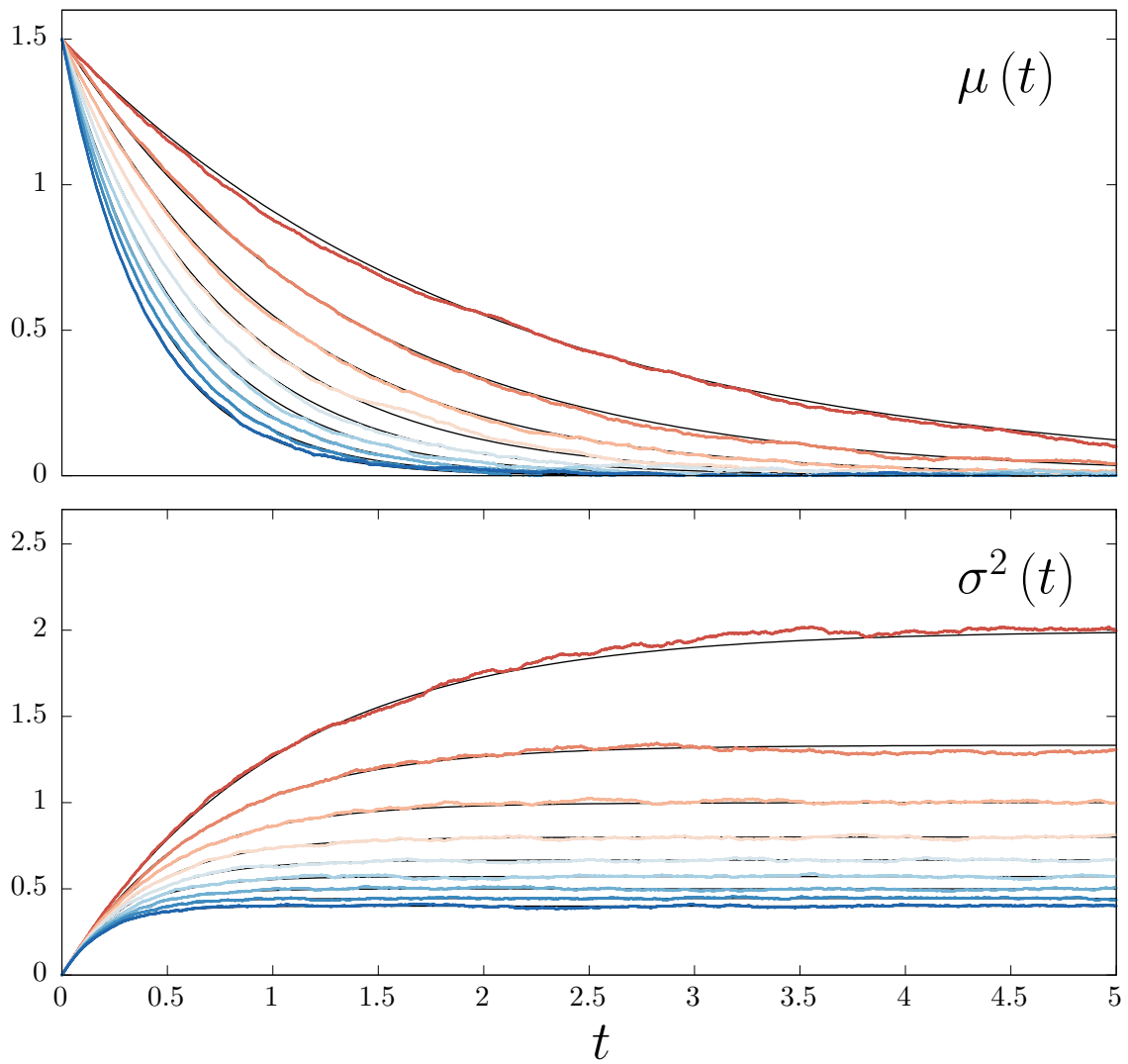


Figure 4.5.: Ornstein-Uhlenbeck process: comparison of ensemble means $\mu(t)$ and variance $\sigma^2(t)$ to their theoretical values (black, Equation 2.34) for 10 ensembles of 10^4 non-interacting particles, each ensemble experiencing a different temperature in the range $\beta = 0.25$ (red) to $\beta = 2.5$ (blue).

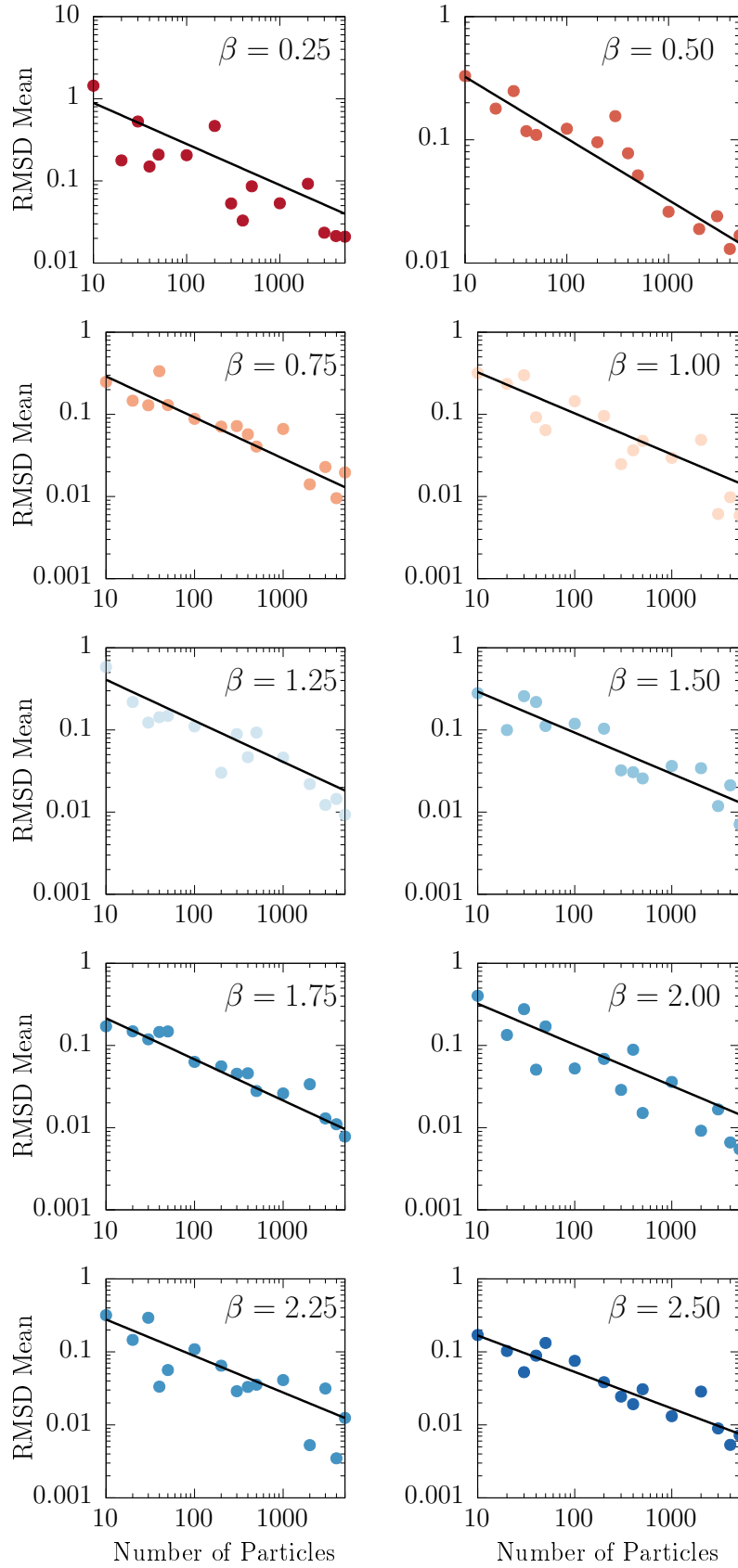


Figure 4.6.: Ornstein-Uhlenbeck process: RMSD of ensemble mean vs. number of particles N for 10 different sets of simulations, each with a different β . Colors corresponding to the same β values in Figure 4.5.

4. Results and Discussion

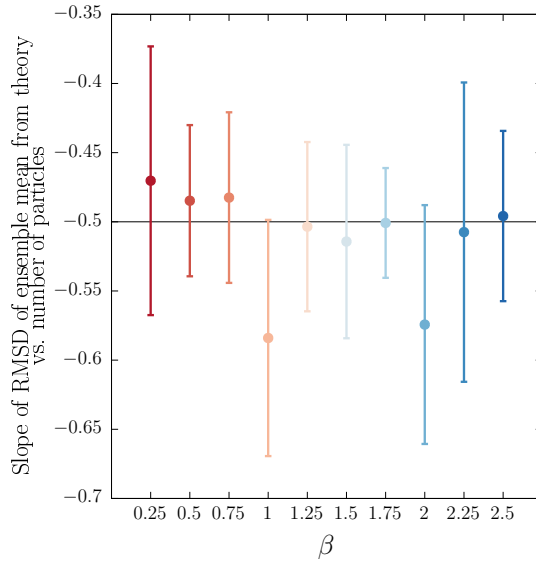


Figure 4.7.: Ornstein-Uhlenbeck process: linear regressions for the curves of RMSD vs. number of particles (log-scale) for each of 10 simulation sets. The theoretical value -0.5 , which corresponds to a slope of $\frac{1}{\sqrt{N}}$ is shown as a black line. The data points represent each set with colors corresponding to the same β values as in Figure 4.5. The error in slope measurements were taken as the standard error from the regression.

4.1.4. Kramer's Approximation

An important step in the generation of a Markov state model for a system is the correct estimation of transition probabilities between its different states. This can be done by counting the transitions between the states and the transition times, and using an estimation method (e.g. a Bayesian method) to yield the most-likely transition rate and its corresponding confidence intervals. Therefore, the simulation was validated for correct transition rates according to Kramer's approximation, which states a verifiable numerical quantity for the transition rates.

Kramer's approximation applies for a particle escaping a potential well of a known analytical shape by crossing a barrier, also having a known analytical shape. This configuration was implemented as a log-Gaussian potential derived from two Gaussian functions with amplitudes $A = 1$, width $\sigma = 1$ and centers $\mu = \pm M$, where M was varied.

There were two constraints on the possible values of M :

1. The barrier height must be greater enough than $k_B T$ (Equation 2.37) for Kramer's approximation to apply. Choosing a barrier height $\Delta E \geq 5k_B T$, which can be regarded as greater enough than $k_B T$ in this context, results in approximately $M \geq 3.375$.
2. The time scales of the mean escape time increase exponentially as M^2 , which means that the number of steps needed for sampling enough transitions grows extremely fast with the value of M . For example, for $M = 4.5$, $\Delta t = 0.001$, $k_B T = 1.0$, the number of steps needed for 50 transitions to be sampled is $\approx 9 \times 10^8$. This presents a computational limit on the value of M which would yield a result in a reasonable time period.

Therefore, the values for M were chosen in the interval $[3.5, 4]$, and for each M a single particle was placed at $x_0 = M$ and was allowed to diffuse until it transitioned about 50 times between the two wells. Each trajectory was then analyzed using *PyEMMA*, according to the method in 3.1.2. The resulting transition rates can be seen in Figure 4.8.

The transition rates calculated by this method overestimate their theoretical values as expected from the Kramer's approximation. This in turn means that the estimated time scales over-estimate their theoretical values. A look at an example clustered trajectory reveals the reason: some transitions were extremely short-lived (Figure 4.9). This shifted down the estimation of the transition times, thus the calculated transition rates increased in magnitude.

A possible reason for this behaviour is the clustering method. As mentioned in the theoretical background (Section 2.1), the borders between clusters created by the k-means method are sharp, which causes some short "intrusions" of the trajectory from one cluster to another to be counted as full transitions between the clusters. When a particle crossed the potential barrier at $x = 0$ and immediately returned to the well it came from, the rate estimation method counted this as a full transition, when clearly it did not conform to the conditions set by Kramer's theorem; specifically, the escape time is defined as the time it takes a particle which is *in equilibrium* in one well to jump to the other well and remain in it.

4. Results and Discussion

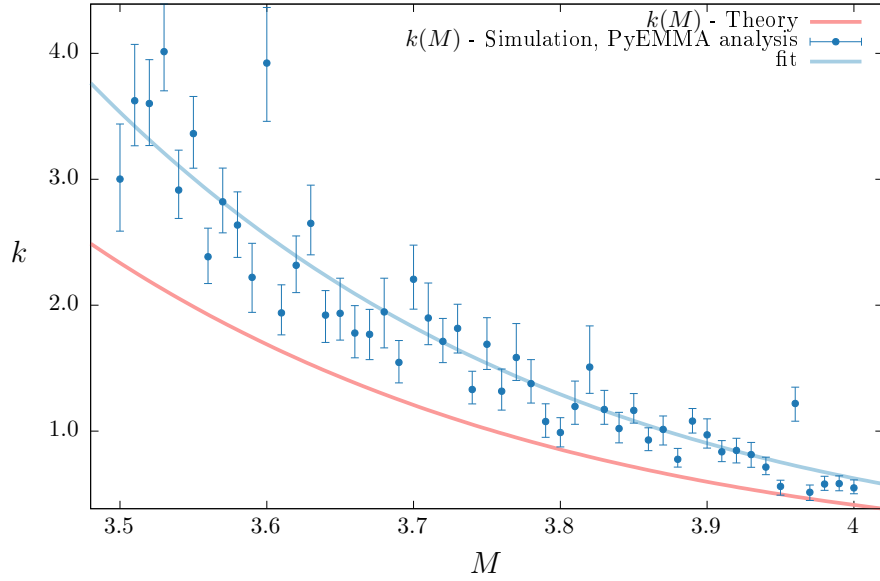


Figure 4.8.: Transition rate estimation by analysis with PyEMMA, for a single particle in a symmetric double well potential (blue dots, confidence interval corresponding to 68%). The rates calculated by the Kramer's approximation are given as a red line.

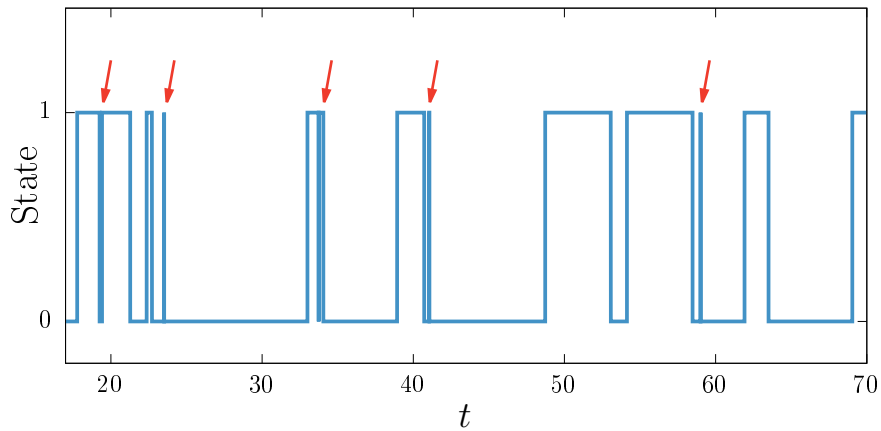


Figure 4.9.: An example of a trajectory with short-lived transitions between states (marked with red arrows). The trajectory was generated by the Kramer's approximation validation simulation with $M = 4$.

Therefore, the short-lived transitions were discarded from the trajectory. A "short-lived" transition was defined as a transition that lasted for less than $\Delta t = 0.1$ (100 steps). The resulted transition rates were still overestimated in relation to the theory, but closer than

without the removal of the short-lived transitions (Figure 4.10).

On the other hand, when the transition rates were estimated using a purely Bayesian method, they were underestimated (Figure 4.11). This method estimated the transition periods $\{\Delta t_i\}$ by simply counting the number of concurrent steps in which a particle is in a positive or negative coordinate before transitioning to a negative or positive coordinate, respectively.

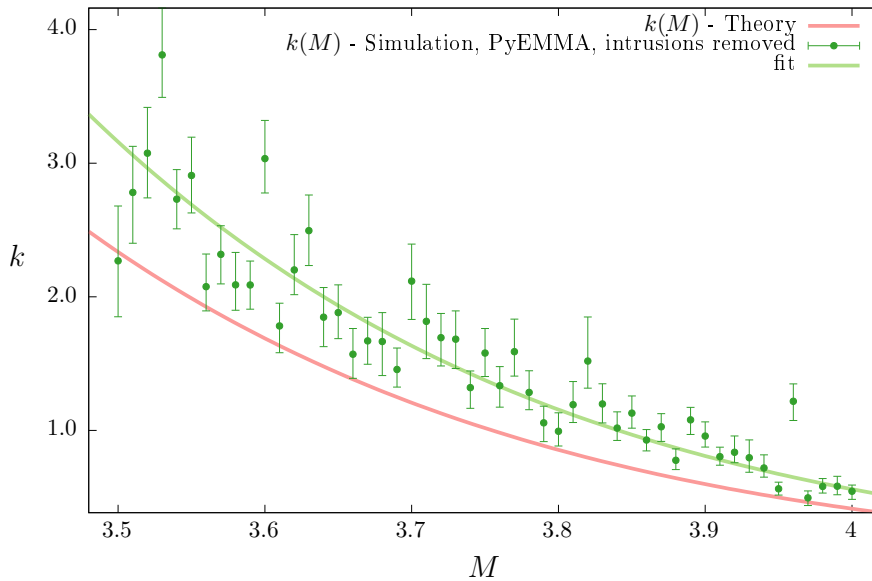


Figure 4.10.: Transition rate estimation by analysis with PyEMMA, for a particle in a symmetric double well potential and removing transitions periods of less than $\Delta t = 0.1$ (green dots, confidence interval corresponding to 68%). The rates calculated by the Kramer's approximation are given as a red line.

The overall result is that the transition rates were either overestimated (by the PyEMMA method) or under-estimated (by the pure Bayesian approach), as seen in Figure 4.12. This means that the simulation did not pass the Kramer's approximation test. However, by tuning the threshold for discarding short-lived transitions, adjusting the purely Bayesian method by discarding only much shorter lived transitions, or perhaps some averaging of both - an optimized method which will yield the correct transition rates might be possible. However, this was not in the scope of this Master's thesis.

4. Results and Discussion

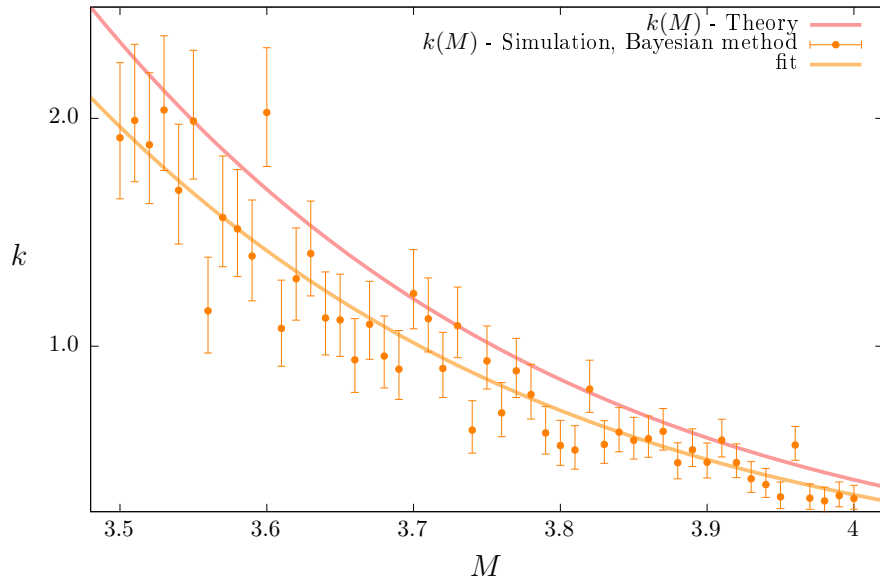


Figure 4.11.: Transition rate estimation by a purely Bayesian approach, for a particle in a symmetric double well potential and removing transitions periods of less than $\Delta t = 0.1$ (orange dots, confidence interval corresponding to 68%). The rates calculated by the Kramer's approximation are given as a red line.

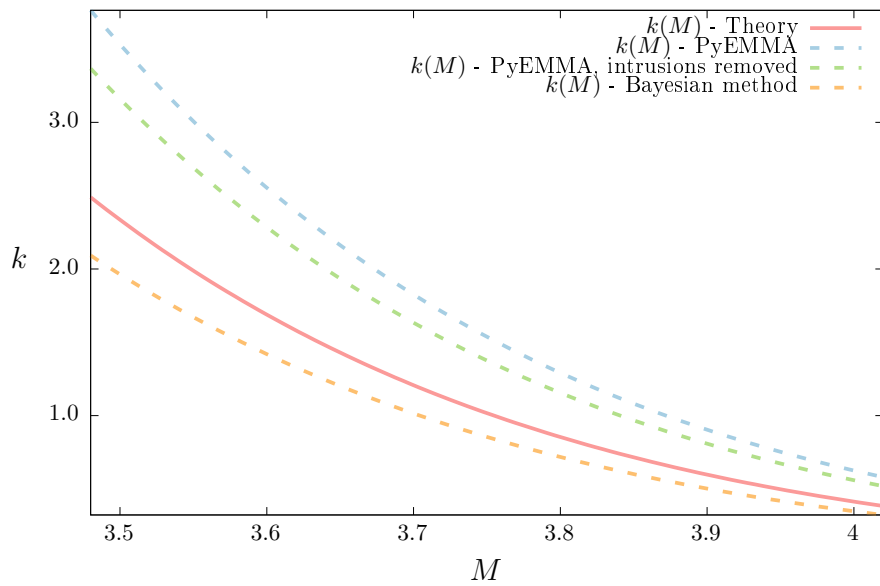


Figure 4.12.: Comparison of three methods of estimation of the transition times between two symmetric wells, from simulated data. The theoretical rate by Kramer's approximation is given as a red line. The dashed lines represent the results of fitting calculated transition rates from three separate methods to the theoretical rate via a single coefficient - i.e. $\alpha \cdot k(M)$, where $k(M)$ is the theoretical transition rate as a function of the distance between the center of the wells and $x = 0$. The three separate methods are discussed in the above text.

4.1.5. Summary of Validation Process

With the exception of the Kramer's approximation test, the simulation passed the validation tests it was subjected to. This means that it correctly simulates both the thermodynamic and also the kinetic properties expected from a Brownian system. While there was no quantitative agreement to theory in the case of Kramer's approximation, it was shown that the probable causes can be solved by optimizing the transition rate estimation method.

As a result, it was deemed possible to use the simulation in the next step discussed in this thesis: generating trajectories that will be used as benchmarks in the automation and optimization of the method used to generate Markov state models from protein dynamics.

4.2. Constructing Markov State Model from Simulated Brownian Dynamics Trajectories

The trajectories generated by the simulation can be interpreted as existing in the state space of a protein's dynamics, and be used as benchmarks for testing the different steps in the process of generating a Markov state model. This part, therefore, presents the generation of a trajectory that was expected to yield specific states and transition processes, and validated these properties against the results of the application of the Markov state model generation method.

4.2.1. Full 2-Dimensional Trajectory

Using the simulation, a trajectory of a single particle was generated using a potential from four Gaussian functions (Figure 3.4). The potential was set up so that trajectories generated by it would have two groups of metastable states: the first composed of a single state, and the second composed of three separate states with equal barriers between them. These barriers are smaller than the barrier between the three states and the first state. Each 100th point of the full trajectory, colored according to its cluster, can be seen in Figure 4.13, and the separate x - and y -coordinates of its first 10^5 steps ($t \in [0, 10^4]$) in Figure 4.14.

4. Results and Discussion

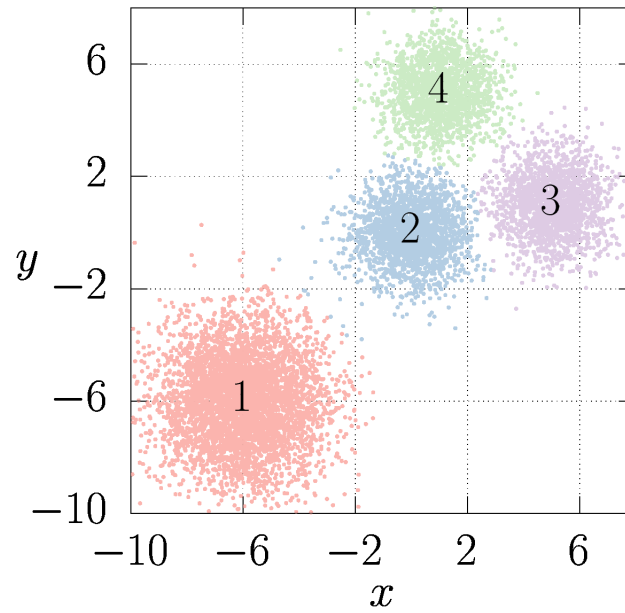


Figure 4.13.: 2D trajectory points generated by using the potential seen in Figure 3.4, colored according to their clustering by the k-means method, with the different clusters labeled.

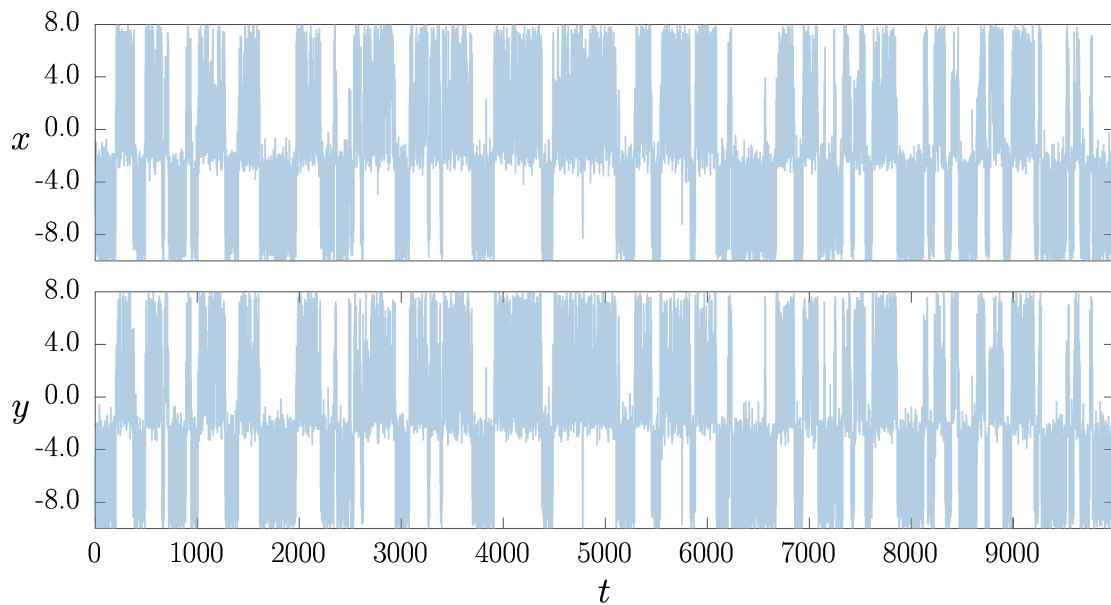


Figure 4.14.: The same trajectory as in Figure 4.13, shown as a function of time separated to the x - and y -axes. Many transitions between state 1 (at $[-6, -6]$) and states 2 – 4 can be seen.

4.2. Constructing Markov State Model from Simulated Brownian Dynamics Trajectories

It is evident from Figure 4.13 that the clusters found correspond to the centers of the wells. This is confirmed by looking at the cluster centers: $(-6.00, -6.03)$, $(1.0, 5.0)$, $(5.0, 1.01)$ and $(0, -0.04)$. These values are almost identical to the centers of the potential wells at $(-6, -6)$, $(1, 5)$, $(5, 1)$ and $(0, 0)$.

When counting all trajectory points which composed the four clusters, we get approximately

$$\begin{aligned} N_1 &= 4.4 \times 10^6 [\text{points}] \\ N_2 &= 2 \times 10^6 [\text{points}] \\ N_3 &= 1.8 \times 10^6 [\text{points}] \\ N_4 &= 1.9 \times 10^6 [\text{points}]. \end{aligned} \tag{4.3}$$

Thus, according to this clustering, the approximate probability ratios between the states were

$$p_{ij}^{\text{count}} \approx \begin{pmatrix} 1 & 0.5 & 0.4 & 0.4 \\ 2.2 & 1 & 0.9 & 1 \\ 2.5 & 1.1 & 1 & 1.1 \\ 2.3 & 1 & 0.9 & 1 \end{pmatrix}, \tag{4.4}$$

while by integration of the Gaussian functions (according to Equation 2.32) one expects to see

$$p_{ij}^{\text{theory}} = \begin{pmatrix} 1 & 0.4 & 0.4 & 0.4 \\ 2.3 & 1 & 1 & 1 \\ 2.3 & 1 & 1 & 1 \\ 2.3 & 1 & 1 & 1 \end{pmatrix}, \tag{4.5}$$

which is very close to p_{ij}^{count} .

Using the function `pyemma.plots.plot_free_energy`, the free energy can be estimated from the trajectory (Figure 4.15). This free energy corresponds well to the potential used (cf. Figure 3.4).

The implied time scale of the slowest transition process, seen in Figure 4.16, seemed to converge at lag time just smaller than $\tau = 75$, and therefore $\tau = 75$ was chosen as the lag time for generation of a Markov state model from the trajectory. This yielded a time scale estimation of 552, with a 95% confidence interval [520, 593].

4. Results and Discussion

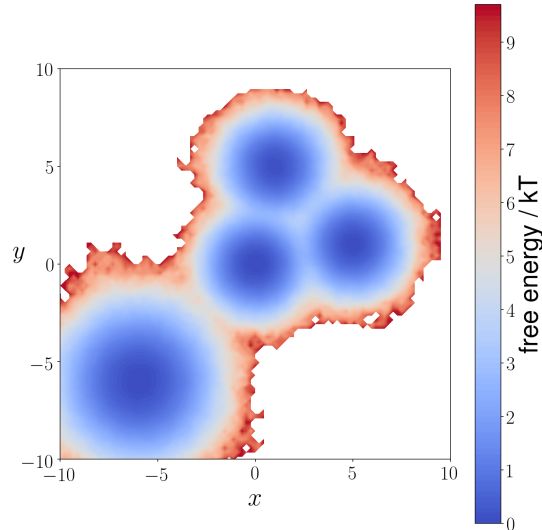


Figure 4.15.: Free energy estimation from the trajectory, generated via PyEMMA's `pyemma.plots.plot_free_energy` function.

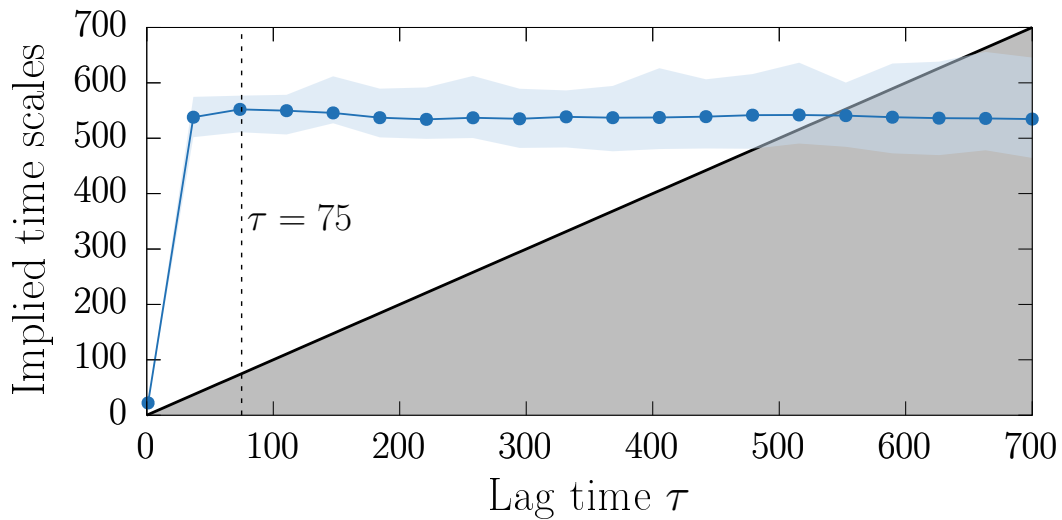


Figure 4.16.: Implied time scales of the slowest transition process in the system as a function of the lag time τ . The light blue area represents a 95% confidence interval for the values of the implied time scales. The gray area represents implied time scales smaller than the lag time. The lag time $\tau = 75$ was chosen for estimation of the transition matrix $T(\tau)$.

According to the analysis, the slowest transition process in the system is the transition between state 1 (at $[-6, -6]$) and states 2 – 4. This can be seen in Figure 4.17, top.

4.2. Constructing Markov State Model from Simulated Brownian Dynamics Trajectories

As explained in Section 2.1.1, the eigenvectors of the transition matrix approximate the eigenvectors of the transfer operator. In the figure the 2nd eigenvector had a positive value for state 1 (red color), and negative values for states 2 – 4 (blue color). This means that the 2nd eigenvector represented a transfer of probability density from state 1 to states 2 – 4, and vice versa.

The next eigenvector had zero value for state 1, and the transfer of probability density is between states 2 – 3 and 4 - meaning that it corresponds to the process $2, 3 \leftrightarrow 4$. In a similar way, the next eigenvalue after that represents a probability density transfer between states 2 and 3, without the other states participating.

This result is somewhat expected from this system, as the distance (and, thus, the energetic barrier) between state 1 and states 2 – 4 is greater than the distance within states 2 – 4. Therefore, the slowest process indeed should be a transition between state 1 and states 2 – 4. Since the energetic barriers between the states 2, 3 and 4 are all smaller, the next slowest processes are expected to be between these states. However, we expect to see a symmetry: the barriers between states 2, 3 and 4 are equal, and so next three processes ($2 \leftrightarrow 3$, $2 \leftrightarrow 4$ and $3 \leftrightarrow 4$) should be identical. However, due to unequal sampling between these three states (as evident by p_{ij}^{count}), the result is two eigenvectors which together represent all possible transitions between the three states (a transition $2, 3 \leftrightarrow 4$, and a transition between the states $3 \leftrightarrow 4$).

PyEMMA estimated the implied time scale of the slowest transition process to be $t_1 = 5522$, with a 95% confidence interval [5194, 5926].

Testing whether the system at a lag time $\tau = 7.5$ conformed to the Chapman-Kolmogorov equality (Equation 2.3) can be seen in Figure 4.18. The test checks the that the probability of being at a state j at times $t + k\tau$ if the system was at state i at time t from the trajectory data is, within an error estimation, the same that is predicted from the Markov state model by k successive applications of the transition matrix $T(\tau)$ to the state of the system. In this case, the transitions between the state 1, 2 and 3 were tested, and the Markov state model generated predicted the transition probabilities correctly within a 95% confidence interval.

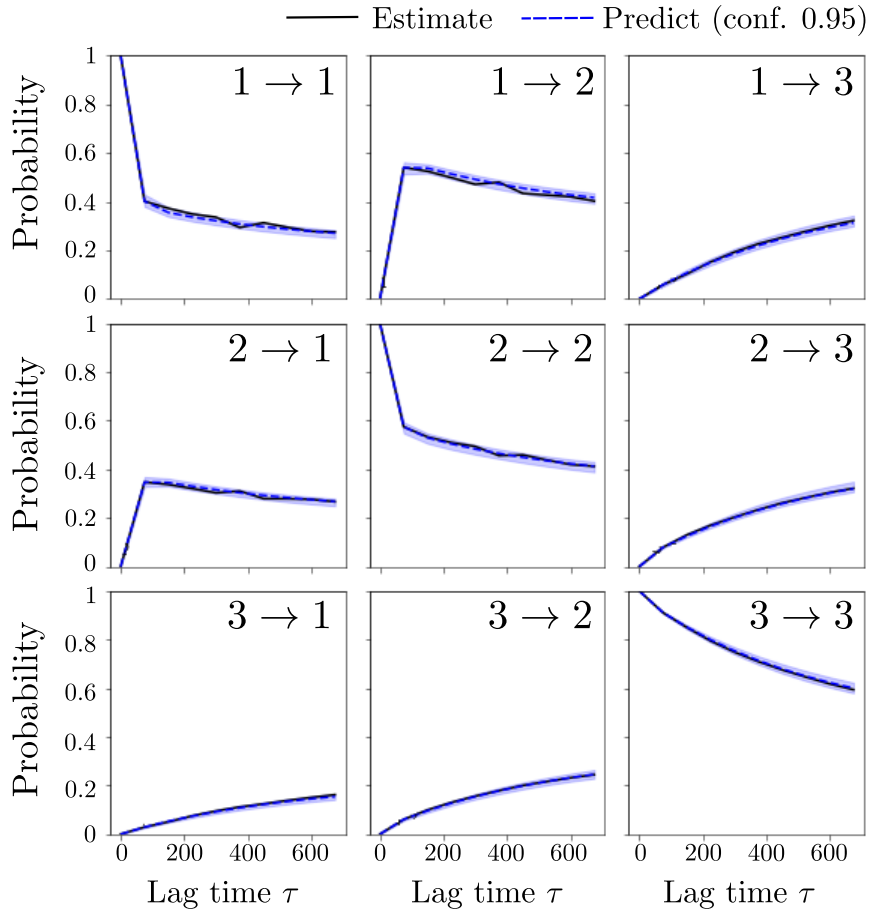


Figure 4.18.: Testing that the system conforms to the Chapman-Kolmogorov equality (Equation 2.3) for lag time $\tau = 7.5$. Each possible transition between the states 1, 2 and 3 is tested for the probability of it occurring in different lag times τ , both from the trajectory data and from the transition matrix $T(\tau = 0.75)$ by applying it k successive times to the state of the system.

4.2.2. Cutting the Trajectory to a Single Full Transition of the Slowest Process

Taking the full trajectory from above and cutting it such that only the first transition from state 1 to states 2 – 4 and the first transition back from states 2 – 4 to state 1 were present, yielded the trajectory seen in Figure 4.19.

Using the same method as for the full trajectory, the implied time scales of the slowest transition process as a function of the lag time τ had "holes" in them (Figure 4.20). Similarly to the case of the Kramer's approximation validation test (Section 4.1.4), looking at the

clustered trajectory, it was evident that there are many short-lived transitions, which were caused by the trajectory crossing the sharp edges between the clusters for a brief moment (Figure 4.21).

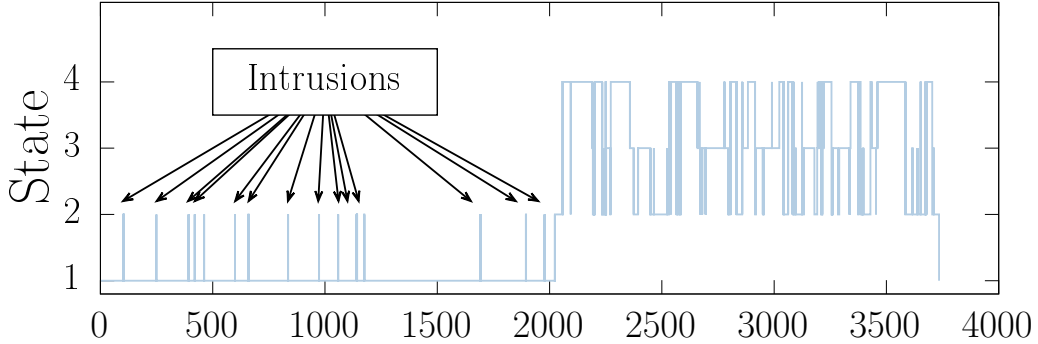


Figure 4.21.: The cut trajectory, clustered. Many short-lived transitions from state 1 to state 2 can be seen (called here "intrusions").

Removing these short-lived transitions resulted in implied time scales that seemed to converge to a value around $t = 2000$, but then dropped to a much smaller value starting at $\tau = 1.15$ (Figure 4.22). This lower value matched exactly to those of the second slowest process, and indeed starting at a lag time $\tau = 1.15$ PyEMMA found only two transition processes instead of three (Figure 4.23).

Using the knowledge about a single transition $1 \rightarrow 2$ at time $t = 2000$, we can use Equation 2.9 to calculate that the resulting time scale should be estimated as $t_1^{\text{Bayes}} = 2000$ as well. When choosing a lag time $\tau = 1$, PyEMMA estimated t_1 to be $t_1^{\text{PyEMMA}} = 1972$, with a 95% confidence interval $[1646, 2464]$, in agreement with t_1^{Bayes} .

4.2.3. Summary of the Markov State Model Generation Tests

The process of constructing a Markov state model from trajectories generated by the simulation discussed in the thesis show that the trajectories indeed generate the expected number of states. The number of processes expected was not observed, however the resulted processes covered all expected transition between the metastable states of the system. In addition, it was shown that the system conformed to the Chapman-Kolmogorov equation, and therefore the Markov state model that was generated could be regarded as a good

4. Results and Discussion

approximation of the trajectory. Combined with the results of the Karner's approximation validation test in the previous section, the transition rates might also be included in the set of confirmed expected properties from the generated trajectories.

All together, these results show that the simulation can be used to generate trajectories which can be used as benchmarks in both testing and optimizing the Markov state model generation method.

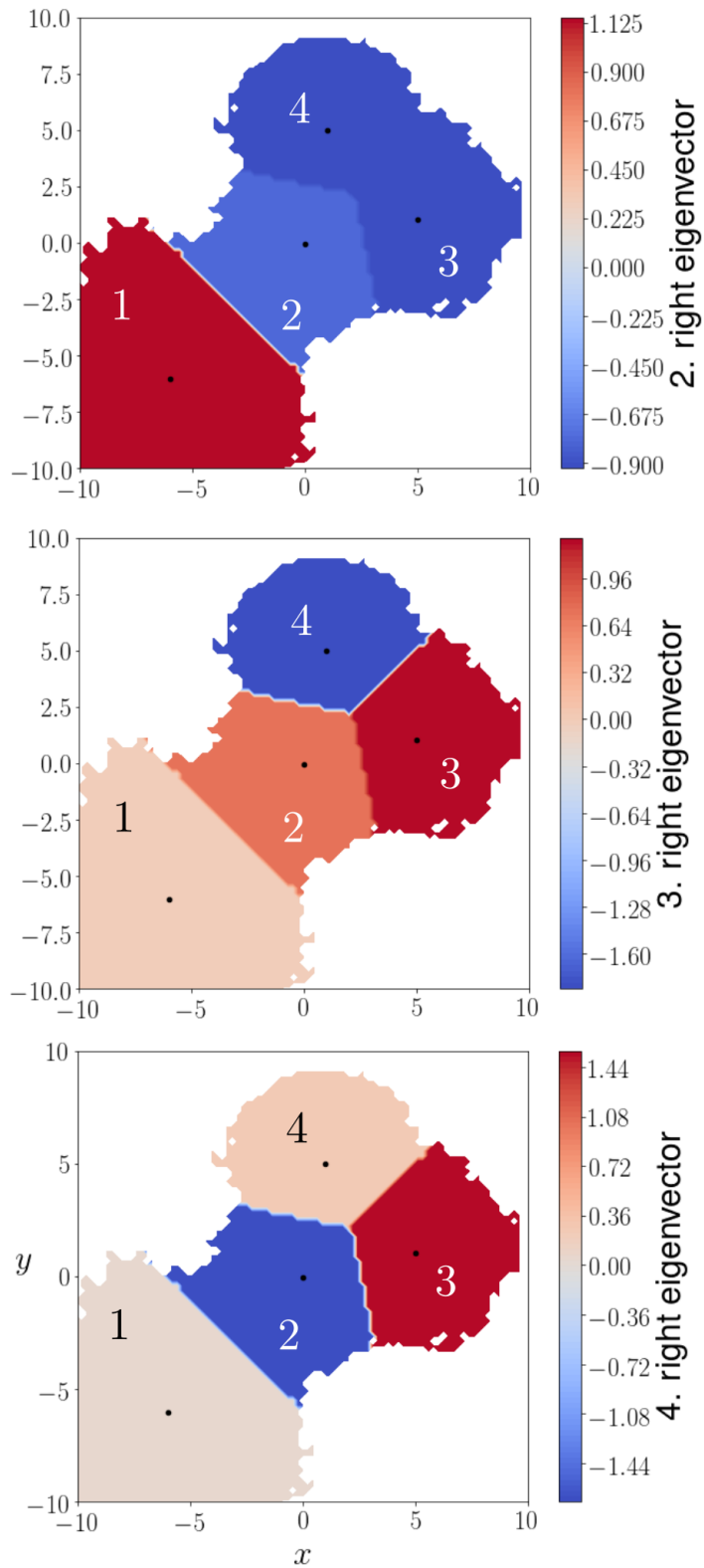


Figure 4.17.: Second, third and fourth eigenvectors resulting from the transition matrix $T(\tau = 75)$. Each eigenvector represents a transfer of probability density from its lower values (blue) to its higher values (red), and vice-versa.

4. Results and Discussion

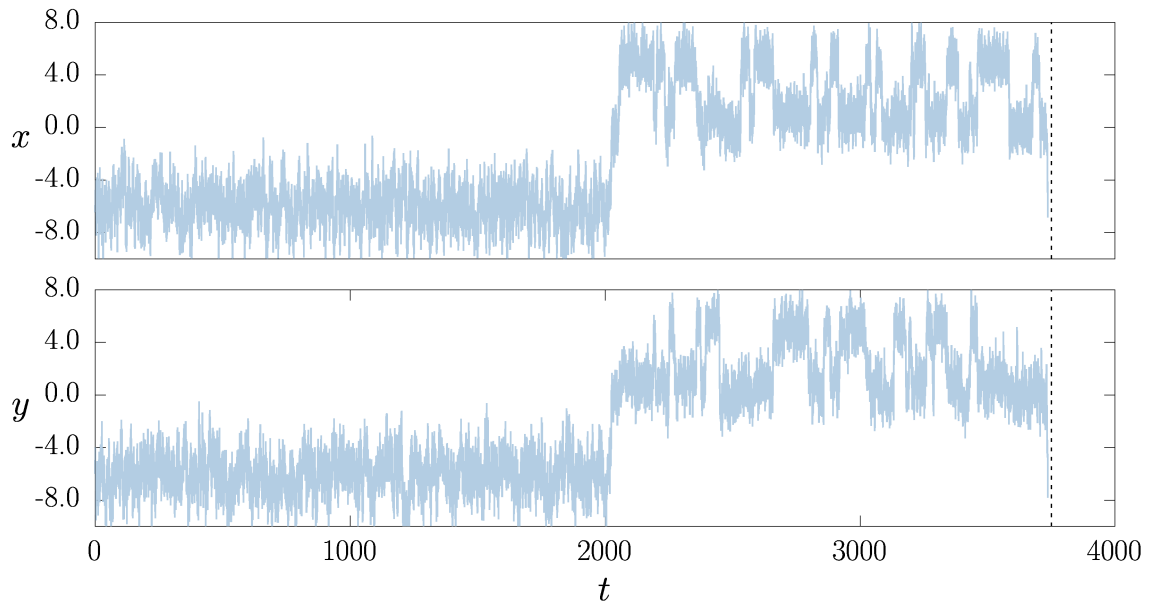


Figure 4.19.: The trajectory seen in Figure 4.14 cut so that there is one transition from state 1 to state 2, and one transition back.

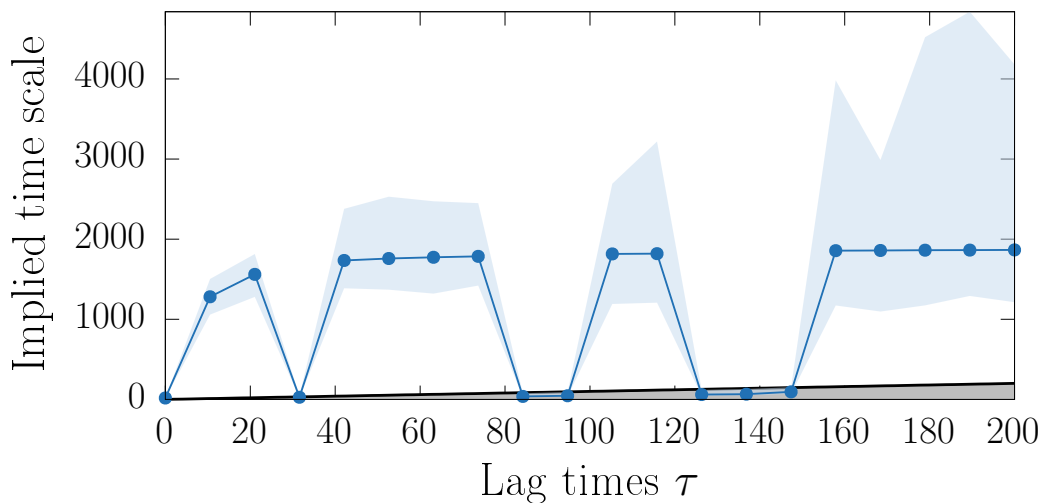


Figure 4.20.: Implied time scales as a function of the lag time τ for the slowest transition process when the trajectory is cut so that it contains only two such transitions: one forward and one back. The light blue area represents a 95% confidence interval for the values of the implied time scales. The gray area represents implied time scales smaller than the lag time. The lag time $\tau = 75$ was chosen for estimation of the transition matrix $T(\tau)$.

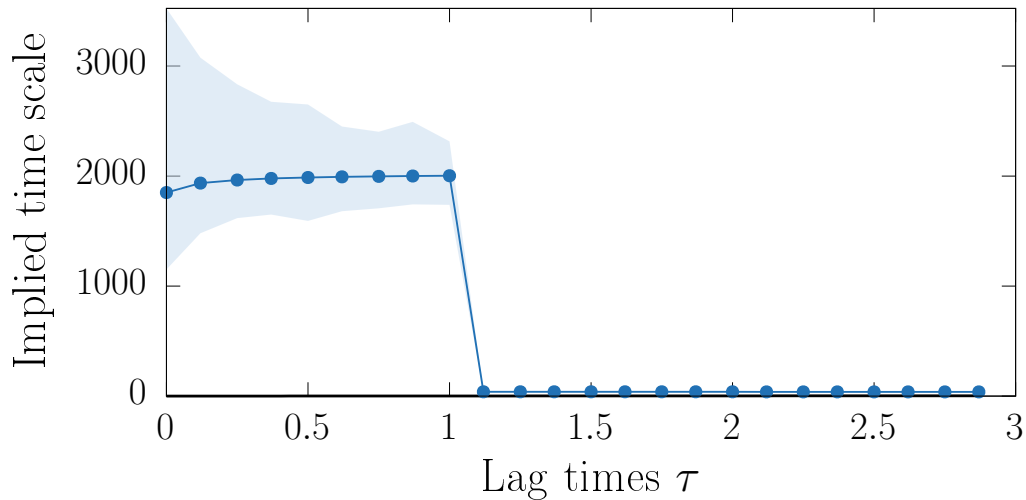


Figure 4.22.: Implied time scales of the slowest transition process as function of the lag time τ . Although the value of the implied time scales converged to around $t_1 = 2000$, at $\tau = 1.15$ there is a sudden drop in value. The light blue area represents a 95% confidence interval for the values of the implied time scales.

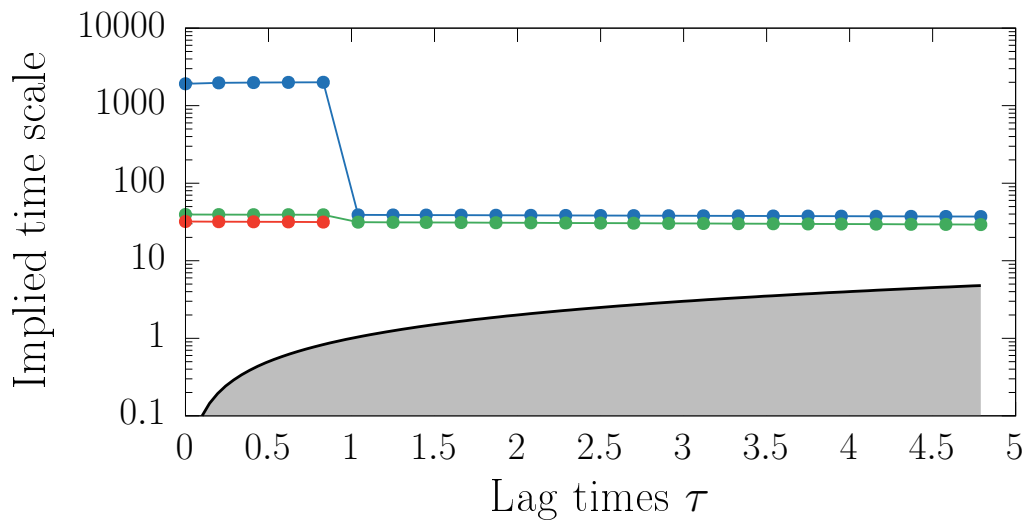


Figure 4.23.: Implied time scales for all three transition processes as a function of lag time τ . The time scales are on a logarithmic scale. Starting at $\tau = 1.15$ the analysis finds only two processes, thus the labeling switches so that the previous second-slowest process (initially colored in green) becomes the slowest process, and is thus colored in blue. The third slowest process (colored in red) becomes the second slowest process, and accordingly it is colored in green. The gray area represents implied time scales smaller than the lag time.

4. *Results and Discussion*

5. Conclusions and Outlook

The validation tests show that the Brownian dynamics simulation behaves according to theory, in both the thermodynamic and kinetic senses. It generates trajectories in a space that can be interpreted as the state space of protein dynamics, and run through the Markov state model generation method used for the analysis of protein dynamics.

An example was shown how the simulation can be used to generate a benchmark trajectory which yielded known and expected results when subjected to the aforementioned Markov state model generation method, such as number of states, their relative population, and the set of transition processes between them.

It was also shown how the program can be used to detect one possible problem which might arise when clustering a trajectory by the commonly used k-means clustering method: short "intrusions" of the trajectory into an area that is clustered as a different state might be counted as short transitions, and thus lower the overall estimated value of the transition time scale.

Utilizing this simulation further can be done by systematically testing each step in the process of generating a Markov state model from protein dynamic (Figure 3.3). For example, generating trajectories which have extremely small number of transitions for some processes, and trying to optimize the estimation of the Markov state models in these situations. This is in fact a problem that was encountered in the course of the *Dynasome* project in our group.

5. *Conclusions and Outlook*

6. Acknowledgements

I would like to thank Prof. Helmut Grubmüller, my supervisor at the MPI-BPC, for giving me the chance to work on this project, guiding me throughout it and providing clarifications.

I would also like to thank all members of the Department for Theoretical and Computational Biophysics at the Max Planck Institute for Biophysical Chemistry, for providing a great and enlighten work environment at the time period in which I was a member of the department. Especially, the support (both in the technical and conceptual sense) of Andreas Volkhardt, Kristian Blom, Malte Schäffner and Nicolai Kozlowski is greatly appreciated.

The feedback given to me at the oral presentation of this project, both by Prof. Bettina Keller, my supervisor in the FU-Berlin, and Dr. Dirk Andrae, is also greatly appreciated.

And lastly, I would like to thank my wife, Liubov Zakharova*, for all her love and support.

*At the time of writing this not yet a PhD, but that is scheduled to change soon.

6. Acknowledgements

Bibliography

1. Marth, J. D. A unified vision of the building blocks of life. *Nature cell biology* **10**, 1015 (2008).
2. Erdin, S., Lisewski, A. M. & Lichtarge, O. Protein function prediction: towards integration of similarity metrics. *Current opinion in structural biology* **21**, 180–188 (2011).
3. Hensen, U. *et al.* Exploring protein dynamics space: the dynasome as the missing link between protein structure and function. *PloS one* **7**, e33931 (2012).
4. Quetschlich, D. *Markov State Models for Comparing Protein Dynamics* MA thesis (Max-Planck-Institute for Biophysical Chemistry, Göttingen, 2016).
5. Bowman, G. R., Pande, V. S. & Noé, F. *An introduction to Markov state models and their application to long timescale molecular simulation* (Springer Science & Business Media, 2013).
6. Prinz, J.-H. *et al.* Markov models of molecular kinetics: Generation and validation. *The Journal of chemical physics* **134**, 174105 (2011).
7. Jiang, R., Zhang, X. & Zhang, M. Q. *Basics of Bioinformatics* (Springer, 2016).
8. Celebi, M. E., Kingravi, H. A. & Vela, P. A. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications* **40**, 200–210 (2013).
9. Van Gunsteren, W. & Berendsen, H. Algorithms for Brownian dynamics. *Molecular Physics* **45**, 637–647 (1982).
10. Landau, L. & Lifshitz, E. *Statistical Physics (Course of Theoretical Physics vol 5)* (Pergamon Oxford, 1958).
11. Schulten, K. & Kosztin, I. *Lecture notes: 'Non-Equilibrium Statistical Mechanics' (PHYCS 498NSM), University of Illinois 2000.*
12. Briane, V., Vimond, M. & Kervrann, C. An overview of diffusion models for intracellular dynamics analysis (2018).

Bibliography

13. Scherer, M. K. *et al.* PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation* **11**, 5525–5542. ISSN: 1549-9618 (Oct. 2015).
14. Dhont, J. K. *An introduction to dynamics of colloids* (Elsevier, 1996).
15. Song, Y. *et al.* Finite element solution of the steady-state Smoluchowski equation for rate constant calculations. *Biophysical Journal* **86**, 2017–2029 (2004).
16. Rosen, M. I. Niels Hendrik Abel and equations of the fifth degree. *The American mathematical monthly* **102**, 495–505 (1995).
17. Kloeden, P. E. & Platen, E. *Numerical solution of stochastic differential equations* chap. 9.1 (Springer Science & Business Media, 2013).
18. Sander, C. & Schneider, R. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics* **9**, 56–68 (1991).
19. Dukka, B. K. Structure-based methods for computational protein functional site prediction. *Computational and structural biotechnology journal* **8**, e201308005 (2013).

Appendices

A. Simulation Code

```
# NumPy and SciPy
import numpy as np
from numpy import exp, sqrt, pi, log
from scipy.special import erf

# For progress bar
from tqdm import tqdm_notebook, tqdm

import sys

# For saving and loading potentials
import pickle

# For multi-dimension integration
import itertools

# Constants
sqrt2pi = sqrt(pi)
sqrt2 = sqrt(2)
s2p = sqrt(2*pi)
s2p_1 = 1.0/s2p

class gaussian():
    """
    Represents a Gaussian function.
    Initialized with N dimensions by setting up
```

A. Simulation Code

```
the values of A (amplitude), M (center) and S (width).
"""
def __init__(self, A, M, S):
    if not A.shape == M.shape == S.shape:
        raise ValueError('Amplitude, Mean and Variance \
                           must have the same shape.')
        return
    self.A = A
    self.M = M
    self.S = S
    self.dim = A.shape[0]

def get_1d_value(self, x, dim):
    """
    Returns the function's value at position x along
    the dimension dim.
    """
    a = self.A[dim]
    m = self.M[dim]
    s = self.S[dim]
    return a * exp(-(x-m)**2/(2*s**2))

def get_partial_derivative(self, pos, dim):
    """
    Returns the function's partial derivative with
    respect to dimension dim, at position pos.
    """
    a = self.A[dim]
    m = self.M[dim]
    s = self.S[dim]
    return (m-pos[dim])/s**2 * self.get_value(pos)

def get_value(self, pos):
    """
    Returns the function's value at position pos.
    """
```

```

    """
    return np.prod([self.get_1d_value(x, i)
                    for i, x in enumerate(pos)])

def integral_1D(self, dim, a, b):
    """
    Integrates the function in the closed interval [a,b]
    along the dimension dim.
    """
    erf_a = erf((a-self.M[[dim]])/(sqrt2*self.S[[dim]]))
    erf_b = erf((b-self.M[[dim]])/(sqrt2*self.S[[dim]]))
    return s2p_1 * self.A[[dim]]*self.S[[dim]] * (erf_b - erf_a)

def integral(self, I):
    """
    Integrates the function over the multi-dimensional
    closed interval I = [a1,b1]X[a2,b2]X...X[an,bn].
    """
    return np.prod([self.integral_1D(d, a, b)
                    for d, (a, b) in enumerate(I)])

class potential:
    """
    Represents a log-Gaussian potential composed
    of m Gaussian functions.
    Initialized with a list of Gaussian functions
    and kBT value.
    """
    def __init__(self, gaussians=None, kBT=1):
        self.gaussians = gaussians
        if gaussians is not None:
            self.num_dims = np.max([g.dim
                                    for g in self.gaussians])
            self.num_gaussians = len(gaussians)

```

A. Simulation Code

```
    else:
        self.num_gaussians = 0
    self.kBT = kBT

def get_value(self, pos):
    """
    Returns the value of the potential at
    position pos.
    """
    val = np.sum([g.get_value(pos)
                  for g in self.gaussians])
    return -self.kBT * np.log(val)

def get_force(self, pos):
    """
    Returns the value of the force derived
    from the potential, at position pos.
    """
    norm_factor = self.kBT / np.sum([g.get_value(pos)
                                     for g in self.gaussians])
    force = np.zeros(self.num_dims)
    for d in range(self.num_dims):
        force[d] = np.sum([g.get_partial_derivative(pos, d)
                          for g in self.gaussians])
    return norm_factor * force

def integral(self, intervals):
    """
    Integrates the potential's Gaussian functions
    over the N-dimensional closed interval
     $I = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n]$ .
    """
    if len(intervals) != self.num_dims:
        raise ValueError('Number of intervals ({}), is different than\
                           number of dimensions ({}).'\

```

```

        .format(len(intervals), self.num_dims))
    return np.sum([ig.integral(intervals)
                   for ig in self.gaussians])

def get_equilibrium_histogram_1D(self, bins, num_steps=1000, dim=0):
    """
    Returns a histogram (using argument bins) representing
    the 1-dimensional equilibrium distribution of the potential,
    calculated by the Boltzmann distribution.
    """
    I = np.array([self.integral([[bins[i], bins[i+1]]])
                  for i, b in enumerate(bins[:-1])])
    return I / np.sum(I) * num_steps

def get_equilibrium_histogram_2D(self, bins_x, bins_y, num_steps=1000):
    """
    Returns a histogram (using argument bins) representing
    the 2-dimensional equilibrium distribution of the potential,
    calculated by the Boltzmann distribution.
    """
    # Create intervals from bins
    intervals_x = [[bins_x[i], bins_x[i+1]]
                   for i, _ in enumerate(bins_x[:-1])]
    intervals_y = np.array([[bins_y[i], bins_y[i+1]]
                             for i, _ in enumerate(bins_y[:-1])])
    interval_list = [intervals_x, intervals_y]

    # Create an array of all possible 2D intervals
    intervals = np.array(list(product(*interval_list)))

    shape = (bins_x.shape[0]-1, bins_y.shape[0]-1)
    I = np.array([self.integral(interval)
                  for interval in intervals]).reshape(shape)
    return I/np.sum(I) * num_steps

```

A. Simulation Code

```
def save(self, filename):
    """
    Saves the potentials to the file filename.
    """
    with open(filename, 'wb') as file:
        pickle.dump(self, file, pickle.HIGHEST_PROTOCOL)

def load(self, filename):
    """
    Loads the content of filename
    (must be a saved potential).
    """
    with open(filename, 'rb') as file:
        new = pickle.load(file)
        self.gaussians = new.gaussians
        self.num_dims = np.max([g.dim
                                for g in self.gaussians])
        self.kBT = new.kBT

class zero_potential(potential):
    """
    A potential  $U=0$ .
    """
    def __init__(self):
        pass

    def get_force(self, pos):
        return pos * 0.0

def simulate(params, notebook=False):
    """
    Simulation. Integrates the Brownian dynamics
    equation of motion, using the potential and
```

```

parameters given in params.
"""
num_steps = params['num_steps']
num_dim = params['num_dim']
num_dim_sqrt = np.sqrt(num_dim)
num_particles = params['num_particles']

A = params['Ddt'] / params['kBT']
B = np.sqrt(2*params['Ddt'])
U = params['potential']

xs = np.zeros(shape=(num_steps, num_dim, num_particles))
xs[0, :, :] = params['x0']

if notebook: # For jupyter notebook
    sim = tqdm_notebook(range(1, num_steps))
else:
    sim = tqdm(range(1, num_steps))

for t in sim:
    drift = np.zeros(shape=(num_dim, num_particles))
    for i in range(num_particles):
        drift[:, i] = A * U.get_force(xs[t-1, :, i])
    noise = B * np.random.normal(size=(num_dim, num_particles))
    xs[t, :, :] = xs[t-1, :, :] + drift + noise
return xs

```

A. Simulation Code

B. Derivation of Transition Rates via Kramer's Approximation for Symmetric Double Log-Gaussian Potential Wells

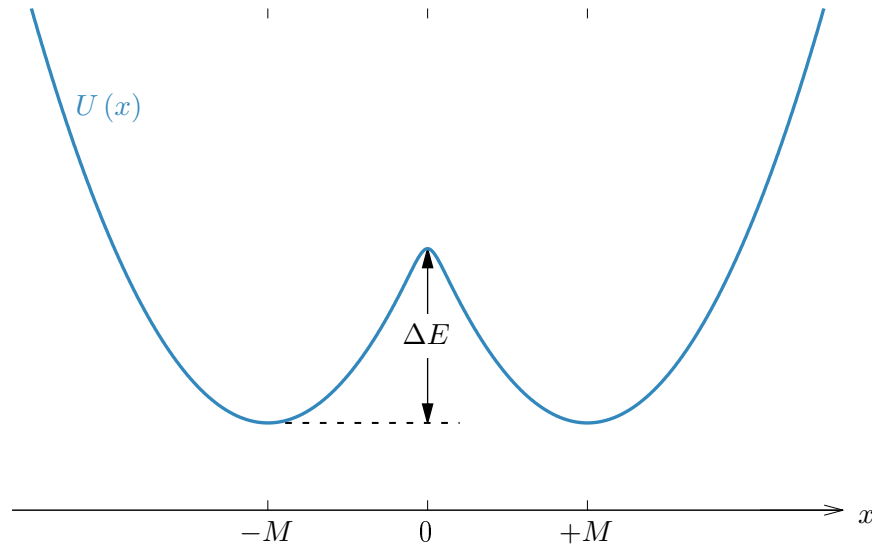


Figure B.1.: Symmetric double potential well.

A potential $U(x) = -\log\left(e^{-\frac{(\mu+x)^2}{2}} + e^{-\frac{(-\mu+x)^2}{2}}\right)$ represents two identical wells at $x = \pm\mu$ (Figure B.1). The height of the barrier at $x = 0$ (labeled B) is therefore

$$\begin{aligned}
 \Delta E &= U(x=0) - U(x=\pm\mu) \\
 &= \log\left(1 + e^{-2\mu^2}\right) - \log\left(2e^{-\frac{\mu^2}{2}}\right) \\
 &= \frac{\mu^2}{2} - \log(2) + \log\left(1 + e^{-2\mu^2}\right).
 \end{aligned} \tag{B.1}$$

The second derivative of $U(x)$ is

$$U''(x) = \frac{-4\mu^2 + 2 \cosh(2\mu x) + 2}{2 \cosh(2\mu x) + 2}, \quad (\text{B.2})$$

and thus

$$\begin{aligned} U''(x=0) &= 1 - \mu^2 \\ U''(x=\mu) &= 1 - \left(\frac{\mu}{\cosh(\mu^2)} \right)^2, \end{aligned} \quad (\text{B.3})$$

meaning

$$\omega_1 = \sqrt{\frac{1 - \left(\frac{\mu}{\cosh(\mu^2)} \right)^2}{m}}, \quad \omega_2 = \sqrt{\left| \frac{1 - \mu^2}{m} \right|}. \quad (\text{B.4})$$

All together, by Kramer's theorem we get (for $D = 1, k_B T = 1$ and thus $\gamma = 1$)

$$k_{A \rightarrow C} = \frac{\sqrt{\left[1 - \left(\frac{\mu}{\cosh(\mu^2)} \right)^2 \right] |1 - \mu^2|}}{2\pi} \exp \left[\frac{-\mu^2}{2} + \log(2) - \log(1 + e^{-2\mu^2}) \right]. \quad (\text{B.5})$$

C. Derivation of Transition Rates Between Two States

C.1. Single Transition

Let $R(t)$ be a trajectory of a system, which at times $0 \leq t < t_1$ is found at a state labeled 1, and at times $t = t_1$ is found at a different state labeled 2 (Figure C.1).

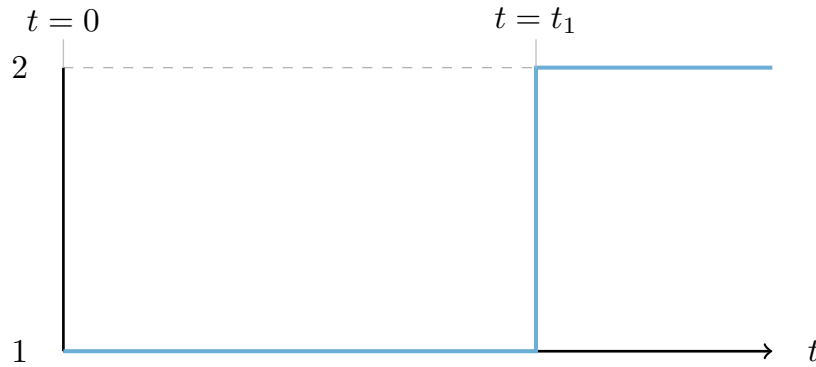


Figure C.1.: A system transitioning from state 1 to state 2 at time $t = t_1$.

The distribution of the transition rate $k = k_{1 \rightarrow 2}$ is given by $p(k | t_1)$. According to Bayes' theorem

$$p(k | t_1) \propto p(t_1 | k) p(k). \quad (\text{C.1})$$

The likelihood $p(t_1 | k)$ can be derived by time discretization of the trajectory $R(t)$ with time step Δt (Figure C.2). For each time step the probability of staying in state 1 is $p = 1 - k\Delta t$, and, thus, the total probability of staying in state 1 for n steps is

$$\begin{aligned} p &= (1 - k\Delta t)^n \\ &= \left(1 - \frac{kt_1}{n}\right)^n, \end{aligned} \quad (\text{C.2})$$

C. Derivation of Transition Rates Between Two States

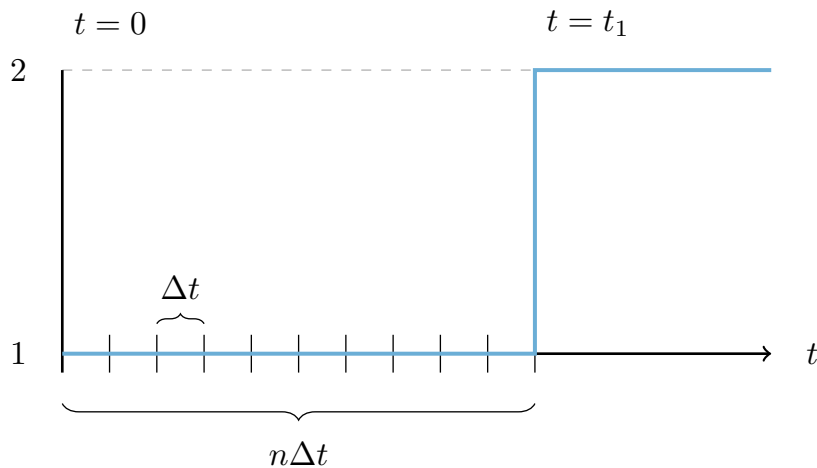


Figure C.2.: A discretization of the system.

since $n\Delta t = t_1$. The probability of transitioning once is k , and therefore the total discrete probability of transitioning at exactly $t_1 = n\Delta t$ is

$$p(t_1 | k) = k \left(1 - \frac{kt_1}{n}\right)^n, \quad (\text{C.3})$$

which in the limit $\Delta t \rightarrow 0$ (i.e. $n \rightarrow \infty$) converges to

$$p(t_1 | k) = ke^{-kt_1}. \quad (\text{C.4})$$

Setting the prior to

$$p(\log(k)) = 1 \quad (\text{C.5})$$

yields (since $p(x) dx = p(y) dy$)

$$p(k) = \frac{1}{k}. \quad (\text{C.6})$$

Plugging equations C.4 and C.6 into equation C.1 yields

$$p(k | t_1) \propto e^{-kt_1}. \quad (\text{C.7})$$

Let us now set $k = e^\tau$, and observe the change to the posterior:

$$\begin{aligned} p(\tau | t_1) &\propto p(k | t_1) \frac{dk}{d\tau} \\ &= e^{-e^\tau t_1} e^\tau \\ &= e^{-e^\tau t_1 + \tau}. \end{aligned} \tag{C.8}$$

The most likely τ is found where $\frac{d}{d\tau} p(k | t_1) = 0$. The derivative of $p(\tau | t_1)$ is

$$\frac{d}{d\tau} p(\tau | t_1) \propto (1 - t_1 e^\tau) e^{-e^\tau t_1 + \tau}, \tag{C.9}$$

and solving for $p(\tau | t_1) = 0$ yields

$$\tau = \log\left(\frac{1}{t_1}\right) \Leftrightarrow k = \frac{1}{t_1}, \tag{C.10}$$

which is the expected result, as rate is the inverse of timescale. Figure C.3 shows a graphical representation of $p(k | t_1)$.

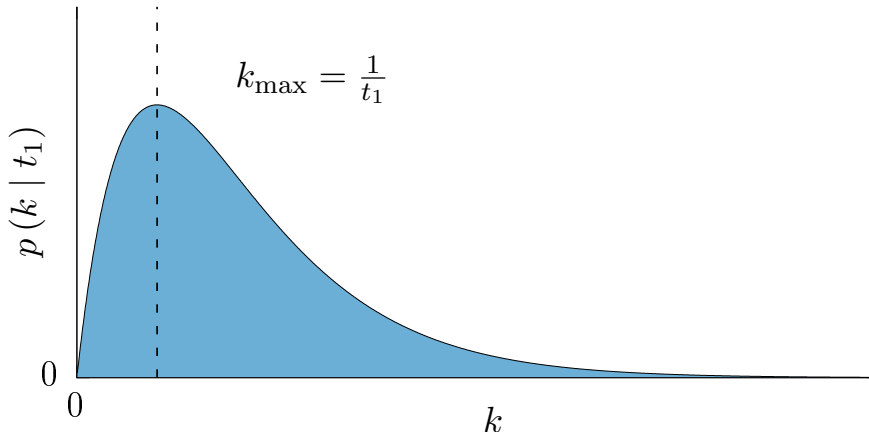


Figure C.3.: Distribution of $p(k | t_1)$ for a single transition observed at $t = t_1$, derived using a Bayesian method.

C.2. Many Transitions

For a sequence of n transitions $1 \rightarrow 2$ at times $\{t_1, t_2, \dots, t_n\}$ and $n - 1$ transitions $2 \rightarrow 1$ at times $\{t'_1, t'_2, \dots, t'_{n-1}\}$, we define the set of time periods Δt_i in which the system was in state 1 before transitioning to state 2:

$$\Delta t_i = \begin{cases} t_1 & \text{if } i = 1 \\ t'_i - t_{i-1} & \text{if } i = 2, 3, \dots, n \end{cases}. \quad (\text{C.11})$$

We then set the likelihood as the product of the separate likelihoods from each of the time periods $\{\Delta t_i\}$ (here $k = k_{1 \rightarrow 2}$):

$$p(\{\Delta t_i\} | k) = \prod_{i=1}^n p(\Delta t_1, \Delta t_2, \dots, \Delta t_n | k) \quad (\text{C.12})$$

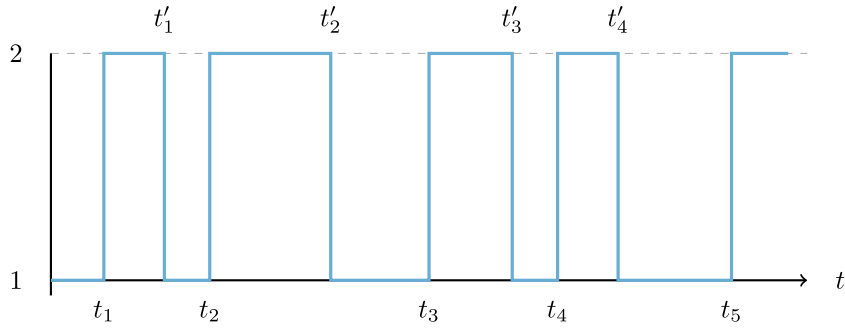


Figure C.4.: Five transitions $1 \rightarrow 2$ at times t_i , and four transitions $2 \rightarrow 1$ at times t'_i .

As with equation C.7, we can now calculate the probability distribution $p(k | \{\Delta t_i\})$ using Bayes' theorem, and utilizing the same prior:

$$\begin{aligned} p(k | \Delta t_1, \Delta t_2, \dots, \Delta t_n) &\propto \prod_{i=1}^n p(\Delta t_i | k) \cdot p(k) \\ &= \prod_{i=1}^n k e^{-\Delta t_i k} \\ &= k^n e^{-k \sum_{i=1}^n \Delta t_i} \\ &= k^n e^{-nk \langle \{\Delta t_i\}_{i=1}^n \rangle}. \end{aligned} \quad (\text{C.13})$$

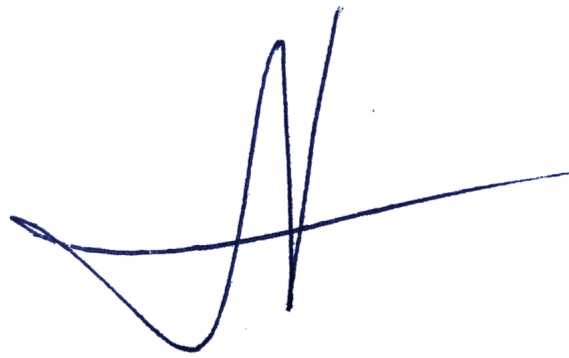
The most likely value of k is then given by

$$k_{\max} = \frac{1}{\langle \Delta t_i \rangle}. \quad (\text{C.14})$$

Declaration

I hereby declare that I have written the present thesis independently, without enlisting any external source/resources other than those specified, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Göttingen, December 2, 2019

A handwritten signature in blue ink, consisting of a series of loops and a long horizontal stroke extending to the right.

.....
Peleg Bar Sapir