Methods for Modeling Metastable Conformational Dynamics from Trajectory Data

Diploma thesis
Department of Mathematics and Computer Science
Freie Universität Berlin

submitted by
Martin Fischbach

Berlin, 2008 - November

This work is dedicated to Julia.

Abstract

Molecular dynamics is characterized by a multitude of processes that occur on different timescales. At a given timescale of interest, these dynamics are often metastable, i.e. there are fast processes in which the molecule vibrates and oscillates within a metastable set of structures, while there are slow processes in which the molecule switches between metastable sets. Considerable effort has been devoted to developing methods which automatically characterize these metastable states and quantify the dynamics within and between them.

The Perron Cluster Cluster Analysis (PCCA) partitions the state space into metastable sets based on a transition matrix. In order to set up this transition matrix, state space must first be partitioned in some way, e.g. by geometrical clustering. For computational efficiency, this clustering often needs to be done on some subspace of "interesting" or "relevant" coordinates, which bears the danger of adding memory to the system by creating overlaps between metastable states.

Hidden Markov Based models with continuous output (HMM-Gaussian and HMM-SDE and in general HMM-VAR) explicitly avoid to work on the full-dimensional state space and allow metastable states to overlap in the few coordinates worked with. The potential drawback here is that a particular model is assumed for the dynamics within metastable states, which may be wrong for a practical system. Additionally, HMMs suffer from local optima, i.e. determining the globally optimal HMM for a given set of observations is a difficult task.

In this work, we compare PCCA and HMM-based methods as for their ability to determine the correct metastable states and the true timescales of model systems and a real molecular system. To avoid the problem of local optimizers in HMMs, a Genetic Algorithm is developed and implemented such a way that the system may asymptotically converge to its globally optimal HMM model, at the cost of additional computational effort. Moreover, the HMM code was extended to work with multiple trajectories and long lagtimes.

Lumping microstates of the state space into metastable sets creates memory due to the finite time necessary to equilibrate within each metastable set. In order to account for this, we find it is essential to conduct the modeling process (either PCCA or HMM) for different lagtimes, and to verify that the rates/timescales of the implied rate matrix are converged. We observe that the reliability of HMMs strongly depends on the projection of the dynamics onto the observed coordinates. Thus a reliable and automatic way to use HMMs on molecular dynamics data is still elusive.

The clustering methods PCCA and HMM-VAR are applied to microsecond MD simulations of an artificial pentapeptide. A Markov model with converged timescales is constructed based on a fine k-means clustering of state space coordinates. A PCCA-based definition of metastable states shows a convergence to these timescales for larger lagtimes τ. HMM-VAR is applied to low-dimensional projections onto the principal components of state

space coordinates. For 1 and 2 dimensions, HMM-VAR provides a reasonable agreement to the true timescales, while the agreement becomes worse with an increased number of dimensions. Furthermore, the direct comparisons of the assignment of metastable states in PCCA and HMM-VAR show significant discrepancies. The observed discrepancies may be due to the approximation of conformational states via Gaussians and due to the increasing number of fit parameters in higher dimensions.

Contents

1 Introduction

Molecular dynamics (MD) simulation generate large amounts of data. Models are required to analyze these data [4] and to extract the essential dynamical features and to provide a physical interpretation. A common approach is to describe the system dynamics in terms of transitions between coarse partitions of conformational space. One basic question which immediately arises is how best to partition state space into discrete states.

Several clustering methods, that partition space, are available. Two major groups of clustering methods are available: these based on geometric proximity and those clustering methods based on kinetics. However, when aiming at modeling the system dynamics an essential requirement to clustering methods is, that they partition space without destroying the essential dynamical behavior of the system.

The biomolecular function often depends on the existence of dynamically metastable states, states in which a molecular system stays for long periods of time. The transitions between metastable states are rare events. On the longest time scales, dynamics is characterized by flipping processes between metastable conformations. On shorter time scales, the dynamical behavior is dominated by flexibility within these conformations.

A construction of a reduced model, which reflects the "effective" dynamics of a biomolecular systems, is desired. One approach to dynamical data-based kinetic clustering is to decompose the dynamics of macromolecules undergoing conformational changes. The approach uses the improved Perron cluster cluster analysis (PCCA) method [3, 23, 22, 17]. Small partitions of conformational space, so called microstates, are merged to clusters. PCCA uses the transition matrix T which models the switching between the microstates to merge the microstates under the constraint of maximal metastability of the clustered dynamics. The obtained clusters then represent approximations of the ideal metastable states. The initial microstates can be defined by geometrical proximity, nevertheless, it is essential not to choose microstates, such that they merge kinetically separated regions of state space.

Another approach avoiding the initial geometric clustering is given by Hidden Markov Models (HMM). The effective dynamics is described by a Markov chain with discrete states, which represent switching between the metastable conformations via a transition matrix T. The dynamics of the system within a metastable state is modeled by a stochastic differential equation (SDE) in a (quadratic) model potential. This description by an SDE also accounts the relaxation from one metastable conformation to another. The combination of HMM and SDE yields the HMM-SDE approach [16, 7]. A generalization of HMM-SDE using autoregressive models is HMM-VAR [13].

A comparison of these two different clustering approaches is performed here. It is evaluated how PCCA and HMM-VAR perform in terms of recovering the true physical timescales of the underlying dynamical system (measured by the dominant eigenvalues of the microstate transition matrix) and how well they agree in terms of assigning microstates to metastable

states. Both methods are first applied to different kinds of low-dimensional model trajectories in order to understand their performance. The simple structure of these trajectories allows general functional tests and validation of the clustering. Due to a small number of essential parameters, diagnostics can easily be performed. To support this evaluation approach the low dimension of the model trajectory provides the possibility to visually interpret clustering results.

The application of both clustering methods to an artificial pentapeptide reveals the capabilities when dealing with real molecular systems. The analysis and evaluation of the clustering results focuses on the comparison of timescales and macrostate assignment between both clustering methods.

The memory that is introduced by merging of microstates of the state space into metastable sets can be compensated by increasing lagtime. This memory effect is induced by the fact, that finite time is required to equilibrate within a metastable state. For increasing lagtimes the implied timescales, which indicate the speed of switching processes between metastable states, converge towards the true timescales. Due to the importance of the lagtime, a rigorous evaluation of the influence of different lagtimes is performed for all of the above trajectory data and a verification of the converged implied timescales is given. For this the HMM-VAR method is adapted to operate with different lagtimes.

The Baum-Welch algorithm [1] estimates the unknown parameters of a HMM under given observation. Internally the algorithm retrieves a maximum-likelihood by the alternating calculation of the forward and backward probabilities, thus improving the likelihood continuously with each iteration step. However, this procedure will only converge to a local maximum that may, due to the raggedness of the likelihood landscape, represent a very bad model for the observed data. Thus finding the global optimum on the likelihood landscape is a relevant problem. Several approaches like simulated annealing [9], Monte Carlo methods [2] and Genetic Algorithms [5] are known to solve that kind of problem. Here, a genetic algorithm in combination with the Baum-Welch algorithm is developed that allows HMM-VAR to find the global optimum and thus yields an improved quality of the clustering results by HMM-VAR, at the expense of computational effort.

1.1 Outline

The thesis is structured as follows: Section 2 covers the theory which is essential for the understanding of the two clustering methods HMM-VAR and PCCA. Furthermore, a theoretical approach to the simultaneous parameter estimation by HMM-VAR for multiple input sequences is given. In Section 3 three elementary two-dimensional model trajectories are clustered with PCCA and HMM-VAR and results are shown. Section 4 presents one possible resort to the convergence to local optima through the use of a genetic algorithm. The application to a model trajectory of Section 3 is presented. Within Section 5 the clustering method HMM-VAR and PCCA are applied to a pentapeptide glycine-serine-glycine-serine-tryptophan to whose glycine end the dye MR121 is attached. The work closes in Section 6 with a conclusion of the obtained results. Open problems which arose during the realization of this thesis are summarized. A final outlook presents ideas which seem reasonable wrt. the experience gained during this work.

2 Theory

2.1 Markov Chains

A Markov chain[1] is a special class of stochastic processes ([12], Chapter 1). So a Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with Markov property, meaning, that given a present state, future states are independent of past states. Accordingly one could define, that future states only depend on the current state, but not on past states, so the system is memoryless. Mathematically, the Markov property is

$$P(X_{n+1} = x | X_n = x_n, \ldots X_1 = x_1) = P(X_{n+1} = x | X_n = x_n).$$

Intuitively speaking, the probability for the realization x of the random variable X_{n+1} and knowledge of all previous realizations x_n to x_1 is equal to the probability for the realization x of random variable X_{n+1} only having knowledge of the realization x_n of variable X_n.

Time-homogeneous Markov chains additionally have the property

$$P(X_{n+1} = x | X_n = y) = P(X_n = x | X_{n-1} = y),$$

where the transition probability from state y to state x does not depend on time.

A definition for the transition probability of going from state i to state j in n time steps is given as

$$p_{ij}^{(n)} = P(X_n = j \mid X_0 = i)$$

and the single-step transition as

$$p_{ij} = P(X_1 = j \mid X_0 = i).$$

A probability vector ν is a vector with $\nu_i \geq 0$ for all i and $\sum_{i=1}^{m} \nu_i = 1$. An initial distribution (vector) π is the vector defined as

$$\pi_i = P(X_0 = i).$$

If the Markov chain is a time-homogeneous Markov chain and the state space of X is finite, a stochastic matrix can be used to describe the transitions of a Markov chain. Therefore,

[1] Often, the term Markov chain is used to mean a discrete-time Markov process.

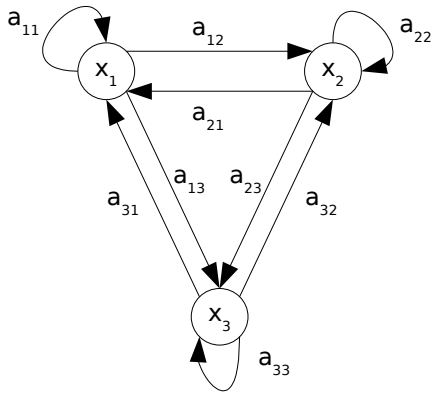

Figure 2.1: Example of a Markov model with three states x_1, x_2 and x_3 which represents a time-homogeneous Markov chain.

this matrix is often called transition matrix. Within a transition matrix T the entry t_{ij} denotes the transition probability to go from state i to state j. The row sum fulfills

$$\sum_j t_{i,j} = 1.$$

The transition for the Markov model given in Figure 2.1 is

$$T = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

2.2 Inverse Problem, Likelihood Principle

Let Θ be a stochastic dynamical system and $O(t)$ be an observable trajectory that can be generated by the system.

A forward problem is, given a model and an initial condition $O(0)$, to generate a trajectory $O(t)$. This problem can be straightforwardly solved with standard numerical techniques. Since Θ is a stochastic system, repeating this process for a given initial condition will not always generate the same trajectory, but rather a distribution of trajectories.

In practical scenarios, one is often faced with the inverse problem. Given an observed trajectory $O(t)$, or an ensemble thereof, reconstruct the underlying dynamical system Θ. Due to the stochasticity of Θ, this problem is ill-posed, i.e. there are many possible models that could have generated the observation. The inverse modeling problem is to make an assertion about the most likely model(s) corresponding to the observation. For an in-depth discussion of the theory of inverse problems see [18].

Within the context of inverse problems the term likelihood is often appropriate. Informally speaking, the likelihood gives a measure of how good a model Θ is in explaining the data O. More formally, likelihood is the hypothetical probability that an event that has already occurred would yield a specific outcome. The concept differs from that of a probability

in that a probability refers to the occurrence of future events, while a likelihood refers to past events with known outcomes. Using this intuition of likelihood our problem can be formulated as

$$L(\Theta) = \alpha P(O|\Theta),$$

that is asking for the probability of O under a given observation Θ. Here, α is a arbitrary constant. Using Bayes theorem, the probability $P(\Theta|O)$ can be expressed as

$$P(\Theta|O) = \frac{P(O|\Theta)P(\Theta)}{P(O)}. \tag{2.1}$$

Thus inserting equation 2.1 into the expression of the likelihood one obtains

$$L(\Theta) = \alpha P(O|\Theta) = \alpha \frac{P(\Theta|O)P(O)}{P(\Theta)}.$$

We make the assumption that all parameters Θ are initially equally distributed, thus that these parameters are uniform prior. Therefore, we can write $P(\Theta) = \gamma$ with the constant $\gamma \in \mathbb{R}$. We introduce $\beta = \frac{\alpha}{\psi}$. Secondly, since the observation O is given, $P(O) = 1$. This allows us to write the likelihood L as a function of the probability $P(\Theta|O)$.

$$L(\Theta) = \frac{\alpha}{\gamma} P(\Theta|O) = \beta P(\Theta|O).$$

So, by maximization of the likelihood L we are able to estimate an optimal model Θ'. The notion of a maximized likelihood is that likelihood, which delivers the most probable parameters under given observation O. It is noted as

$$\Theta' = \arg\max_{\Theta} L(\Theta).$$

Further explanations about the notion of maximum likelihood are given in Section 2.3 and Section 2.5.2.

2.3 Transition Matrix Estimation and Maximum Likelihood Transition Matrix

Usually, $T(\tau)$ is not readily given but needs to be estimated from a set of trajectories. Here, these trajectories are typically from molecular dynamics simulations, but the estimation of Markov model transition matrices is ubiquitous in statistics and has applications in other disciplines, such as finance.

Consider one trajectory with n observations at time resolution τ given:

$$Y = \{y_1 = s(0), y_2 = s(\tau), \ldots, y_n = s((n-1)\tau)\}$$

(The generalization to multiple trajectories is straightforward). Let the frequency matrix $C = (c_{ij})$ count the number of observed transitions between states, *i.e.* c_{ij} is the number of observed transitions from state i at time t to state j at time $t + \tau$, summed over all times t:

$$c_{ij}(\tau) = |\{y_k = i, y_{k+1} = j \mid k = 1...n-1\}|,$$

and, as a shorthand notation we define:

$$c_i := \sum_{k=1}^{m} c_{ik},$$

which is the total number of observed transitions leaving state i. It is intuitively clear that in the limit of an infinitely long trajectory, the elements of the true transition matrix are given by the trivial estimator:

$$\hat{T}_{ij}(\tau) = \frac{c_{ij}}{\sum_k c_{ik}} = \frac{c_{ij}}{c_i}. \tag{2.2}$$

For a trajectory of limited length, the underlying transition matrix $T(\tau)$ cannot be unambiguously computed. The probability that a particular $T(\tau)$ would generate the observed trajectory is given by:

$$p(Y|T) = \prod_{k=1}^{n-1} T_{y_k, y_{k+1}} = p(C|T) = \prod_{i,j=1}^{m} T_{ij}^{c_{ij}}$$

Vice versa, the probability that the observed data was generated by a particular transition matrix $T(\tau)$ is

$$p(T|C) \propto p(T)p(C|T) = p(T) \prod_{i,j \in S} T_{ij}^{c_{ij}}, \tag{2.3}$$

where $p(T)$ is the prior probability of transition matrices before observing any data. $p(C|T)$ is called likelihood and the goal of the transition matrix estimation is usually to identify the maximum of $p(C|T)$, i.e., the maximum likelihood estimator. For the case of a uniform prior, this is identical to the transition matrix with maximum posterior probability.

We will now derive the Maximum Likelihood Estimator by finding the transition matrix that maximizes $p(C|T)$:

$$L = p(C|T) = \prod_{i,j \in S} T_{ij}^{c_{ij}}.$$

The likelihood is difficult to work with due to the product. For optimization purposes it is therefore a common "trick" to instead work with the logarithm of the likelihood (log-likelihood):

$$\log L = \log p(C|T) = \sum_{i,j \in S} c_{ij} \log T_{ij}.$$

This is meaningful since the logarithm is a monotonic function: as a result, the maximum of $\log L$ is also the maximum of L. However, this function is not bounded from above, since for $T_{ij} \to \infty$, $\log L \to \infty$. Of course, we somehow need to restrict ourselves to sets of variables which actually form transition matrices, i.e., they satisfy the constraint:

$$\sum_j T_{ij} = 1.$$

When optimizing with equality constraints, one uses Lagrangian multipliers. The Lagrangian for $\log L$ is given by:

$$F = \log L + \lambda_1(\sum_j T_{1j} - 1) + ... + +\lambda_m(\sum_j T_{mj} - 1).$$

This function is maximized by the maximum likelihood transition matrix. It turns out that F only has a single stationary point, which can be easily found by setting the partial derivatives to zero. Those are given by

$$\frac{\partial \log F}{\partial T_{ij}} = \frac{c_{ij}}{T_{ij}} + \lambda_i.$$

Set to 0:

$$\frac{c_{ij}}{\hat{T}_{ij}} + \lambda_i = 0$$
$$\lambda_i \hat{T}_{ij} = -c_{ij}.$$

We now make use of the transition matrix property:

$$\lambda_i \sum_{j=1}^{m} \hat{T}_{ij} = \lambda_i = -\sum_{j=1}^{m} c_{ij} = -c_i$$

and thus:

$$\frac{c_{ij}}{\hat{T}_{ij}} - c_i = 0$$
$$\hat{T}_{ij} = \frac{c_{ij}}{c_i}.$$

It turns out that $\hat{T}(\tau)$, as provided by Eq. (2.2), is the maximum of $p(C|T)$ and thus also of $p(T|C)$ when transition matrices are assumed to be uniformly distributed *a priori*. In the limit of infinite sampling, $p(T|C)$ converges towards a delta distribution with its peak at $\hat{T}(\tau)$.

2.4 Implied Timescales

The eigenvalue/eigenvector pairs of a transition matrix indicate the elementary processes of the system. Here, an eigenvalue yields the implied rate or implied timescale of the corresponding process and the eigenvector bears the information between which states the corresponding process switches.

We will see later, how this information is used by PCCA to identify metastable sets. The physically interesting information provided by an eigenvalue, namely the implied rate or timescale is obtained by considering following relation:

$$T(\tau) = \exp(\tau K),$$

where K is the generator of the system. For strictly positive definite transition matrices this relation can be inverted providing the rate matrix that is implied by $T(\tau)$

$$K = \frac{1}{\tau} \cdot \log T(\tau) = \frac{1}{\tau} \cdot Q^{-1} \cdot \log(\Lambda) \cdot Q,$$

with $\log(\Lambda) = diag(\log(\lambda_1), \ldots, \log(\lambda_m))$ and Q the matrix of eigenvectors $Q = [q_1 \ \ldots \ q_m]$, being the eigenvalues of K and

$$k_i = \frac{-\log(\lambda_i)}{\tau}$$

being the i^{th} implied rate, or vice versa,

$$\tau_i^* = \frac{-\tau}{\log(\lambda_i)}$$

being the i^{th} implied timescale.

Since the implied timescales are a physical property of the system they should be invariant with respect to numerical modeling. We can thus use the implied timescales as an indicator for Markovianity. For a Markov process, the population of the states at some time $t + n\tau$, can be expressed by applying the Transfer matrix n times to a population at a previous time t, which leads to the Chapman-Kolmogorov equation [21]:

$$p(t + n\tau) = p(t)\,[T(\tau)]^n = p(t)\,S(n\tau)$$

where we have defined $S(n\tau) := [T(\tau)]^n$. Using the eigenvalue-eigenvector equations:

$$\begin{aligned} \Lambda\, Q &= Q^T\, T(\tau) \\ \Gamma(n)\, Q &= Q^T\, S(n\tau), \end{aligned}$$

where Λ and $\Gamma(n)$ are diagonal matrices with eigenvalues $\lambda_1, \ldots, \lambda_m$ and $\gamma_1(n), \ldots, \gamma_m(n)$ on the diagonal, respectively. We can write:

$$[T(\tau)]^n = Q^{-1}\,\Lambda^n\,Q^T = Q^{-1}\,\Gamma(n)\,Q^T = S(n\tau),$$

and hence the eigenvalues λ_i of $T(\tau)$ are related to the eigenvalues $\gamma_i(n)$ of $S(n\tau)$ by:

$$\gamma_i(n) = \lambda_i^n.$$

This relation can equivalently be transformed to

$$\tau_i^* := -\frac{n\tau}{\ln \gamma_i(n)} = -\frac{\tau}{\ln \lambda_i}, \tag{2.4}$$

where τ_i^* is the characteristic timescale corresponding to the decay of eigenmode i. According to this equation, if the process associated to eigenmode i is Markovian, then τ_i^* is constant and thus independent of n. For many processes, this is true only for some minimum timescale $n\tau > \tau_{\text{mem}}$, after which the memory pertaining to the inter-state dynamics has disappeared. In this case, the Markov model of the system must employ the Markov matrix $S(n\tau)$ instead of $T(\tau)$. The time $n\tau$ which is used to set up the Markov matrix is referred to as lagtime. It is desirable to find a definition of states such that the lagtime is minimal, *i.e.* such that there is minimal memory in the system.

Note that choosing the number or size of microstates for a given system is not trivial if the statistics is poor. If we have lots of statistics, it is always best to use as many microstates as possible, in order to avoid merging kinetically separated states. In practice, statistics are always limited and using too many states could mean that we overestimate the timescales of the system, since different transitions to the same metastable state are not counted correctly due to too fine partitioning.

2.5 Hidden-Markov Models (HMM)

2.5.1 General Properties

A hidden Markov model is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters. An example of a simple hidden Markov is given in Figure 2.2. Hidden parameters are determined from the observable parameters. These extracted model parameters are used to perform further analysis.

In contrast to a Markov model, where each state is directly visible to the observer and transition probabilities are the only parameters, a hidden Markov model consists of states which are not directly visible (hidden states). But instead variables influenced by these states are visible. Each state has a probability distribution over the possible outputs. Thus the sequence of observed states, generated by the HMM, gives information about the sequence of hidden states.

A Hidden Markov model is a probabilistic of the joint probability of a collection of random variables $\{O_1, \ldots, O_T, Q_1, \ldots, Q_T\}$. The sequence $Q_1, \ldots, Q_T$ is the hidden process, the Q_t variables are hidden and discrete and denote the state of the system at time t. The sequence $O_1, \ldots, O_T$ is the sequence of observed states, O_t variables are either continuous or discrete observations.

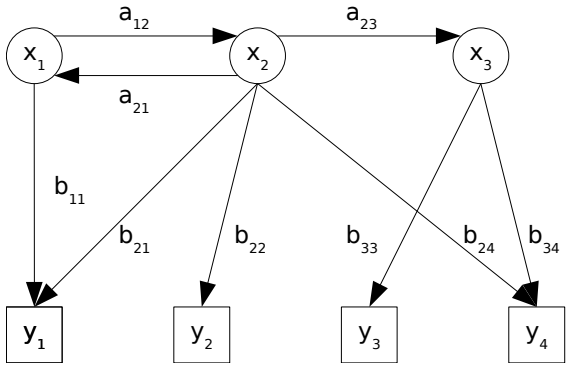

Figure 2.2: Schema of a basic Hidden-Markov model. The hidden states have the labels x_1, x_2 and x_3. The transitions probabilities between the hidden states are labeled a_{ij}, f.e. a_{12} is the transition probability to go from state x_1 to state x_2. Each hidden state x_i produces some output or observation y_j with a certain probability b_{ij}, thus b_{ij} is the output probability for observation y_j if the hidden state is x_i.

In the HMM context the sequence of hidden states is assumed to be a Markov process. Thus this process has the Markov property: a state Q_{t+1}, given the history $Q_1, \ldots, Q_t$, depends only on the previous state Q_t:

$$P(Q_{t+1} = s_i | Q_1, \ldots, Q_T) = P(Q_{t+1} = s_i | Q_t).$$

Furthermore we assume that the underlying hidden Markov chain defined by $P(Q_t|Q_{t-1})$ is time homogeneous, thus independent of the time t. Therefore, we can represent $P(Q_t|Q_{t-1})$ as a time-independent stochastic transition matrix. Since the HMM state space in general is finite, we thus are concerned with a Markov chain, which is characterized by the so-called transition matrix $A = (a_{ij})$. An entry a_{ij} corresponds to the conditional probability of switching from the hidden state s_i to s_j. The sum over j has to be one for each i, thus the transition matrix A is a row-stochastic matrix.

Each hidden state causes a specific output that might be either discrete or continuous. This output is distributed according to a certain conditional distribution. Thus realizations of HMMs are concerned with two sequences, an observation sequence and a sequence of hidden states.

An HMM is formally defined as a tuple $\lambda = (S, V, A, B, \pi)$:

- $S = (s_1, \ldots, s_N)$ is the set of hidden states,
- V is the observation or output space, thus $O_t \in V$,
- $A = (a_{ij})$ is the transition matrix, where a_{ij} is the probability of switching from the hidden state s_i to s_j,
- B as a vector of probability density functions (pdf) in the observation space

- $\pi = (\pi_{1,}, \ldots, \pi_N)$ is a stochastic vector, that describes the initial state distribution $\pi_i = P(Q_i = s_i)$.

According to the HMM a complete set of HMM parameters for a given model is: $\lambda = (A, B, \pi)$. There are three basic problems associated with HMMs:

- Determine $P(O|\lambda)$ for some $O = (o_1, \ldots, o_T)$, that means to determine the probability of observe a certain output sequence $O = (o_1, \ldots, o_T)$ under given parametrization λ.
- Given $O = (o_1, \ldots, o_T)$ and some λ, find the best state sequence $q = (q_1, \ldots, q_T)$ that explains O. This is the search for the most probable hidden path (Viterbi path) under given O and λ. To determine this sequence, the Viterbi algorithm is used.
- Find $\lambda^* = \arg\max_{\lambda} p(O|\lambda)$. The Baum-Welch algorithm solves this problem.

2.5.2 Maximum-likelihood

We have a density function $p(x|\lambda)$ that is governed by the set of parameters λ. Furthermore we have a data set of size N, drawn from that distribution, $X = \{x_1, \ldots, x_n\}$. We assume that these data is independent and identically distributed with distribution p. The resulting density for the samples is

$$p(X|\lambda) = \prod_{i=1}^{N}(x_i|\lambda) = L(\lambda|X)$$

The function $\mathcal{L}(\lambda|X)$ is called the likelihood of the parameters λ given the data. The likelihood is thought of as a function of the parameters λ where the data X is fixed. In the maximum likelihood problem, our goal is to find the λ that maximizes $\mathcal{L}$. So we want to find

$$\lambda^* = \arg\max_{\lambda} \mathcal{L}(\lambda|X).$$

Often one uses the log-likelihood $\log L(\lambda|X)$ instead, because it is analytically easier.

The difficulty of our problem depends on the form how $p(x|\lambda)$ changes. If $p(x|\lambda)$ is a single Gaussian distribution, where $\lambda = (\mu, \sigma^2)$ then we can set the derivative of $\log \mathcal{L}(\lambda|X)$ to zero and solve directly. However for many problems it is impossible to find such analytical expressions.

2.6 Expectation-Maximization (EM) Algorithm

The Expectation-Maximization algorithm is a general method of finding the maximum-likelihood parameter estimation of an underlying distribution from a given data set. An excellent introduction to this field is given in [1].

The EM-algorithm repeats the two steps below until convergence is obtained:

- Expectation-step: this step evaluates the expectation value Q based on the given parameter set λ_k.
- Maximization-step: this step determines the refined parameter set λ_{k+1} by maximizing the expectation

$$\lambda_{k+1} = \arg\max_{\lambda} Q(\lambda, \lambda_k),$$

the maximization guarantees $\mathcal{L}(\lambda_{k+1}) \geq \mathcal{L}(\lambda_k)$.

2.6.1 Baum-Welch Algorithm - EM algorithm for HMMs

In this section, we derive the EM algorithm for finding the maximum-likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors. This algorithm is called Baum-Welch algorithm. It is a special instance of the EM algorithm based on HMMs.

The probability of seeing the partial sequence $o_1, \ldots, o_t$ and ending up in state i at time t is given by:

$$\alpha_i(t) = P(O_1 = o_1, \ldots, O_t = o_t, Q_t = i|\lambda).$$

The variable $\alpha_i(t)$ is called the forward variable. We can efficiently define $\alpha_i(t)$ recursively as:

- $\alpha_i(1) = \pi b_i(o_1)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} (\alpha_i(t) a_{ij}) b_j(o_{t+1})$
- $P(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T)$. That is the probability of the observation of the sequence $o_1, \ldots, o_T$ under given parameter λ. It is given by summation over all α_i at fixed time T.

The backward procedure is similar. The probability of the ending partial sequence $o_{t+1}, \ldots, o_T$ given that we started at state i at time t is:

$$\beta_i(t) = P(O_{t+1} = o_{t+1}, \ldots, O_T = o_T|Q_t = i, \lambda).$$

Accordingly the variable $\beta_i(t)$ is called the backward variable. Analogously we can define $\beta_i(t)$:

- $\beta_i(T) = 1$
- $\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1)$
- $P(O|\lambda) = \sum_{i=1}^{N} \beta_i(1) \pi_i b_i(o_1)$

We proceed and compute the probability $\gamma_i(t)$ of being in state i at time t for the state sequence O

$$\gamma_i(t) = P(Q_t = i|O, \lambda)$$

with respect to the forward- and backward-variables. Note:

$$P(Q_t = i|O, \lambda) = \frac{P(O, Q_t = i|\lambda)}{P(O|\lambda)} = \frac{P(O, Q_t = i|\lambda)}{\sum_{j=1}^{N} P(Q, Q_t = j|\lambda)}.$$

Furthermore because of the conditional independence it is:

$$\alpha_i(t)\beta_i(t) = P(o_1, \ldots, o_t, Q_t = i|\lambda)P(o_{t+1}, \ldots, o_T|Q_t = i, \lambda) = P(O, Q_t = i|\lambda)$$

and thus the definition of $\gamma_i(t)$ in terms of $\alpha_i(t)$ and $\beta_i(t)$ is:

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}.$$

The probability of being in state i at time t and being in state j at time $t+1$ is given as

$$\xi_{ij}(t) = \frac{P(Q_t = i, Q_{t+1} = j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_i(t) \cdot a_{ij}b_j(o_{t+1})\beta(t+1)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i(t) \cdot a_{ij}b_j(o_{t+1})\beta(t+1)}.$$

Through the summation over all time steps t one obtains the expected total number of transitions away from state i for O :

$$\sum_{t=1}^{T} \gamma_i(t).$$

With the same assumptions one obtains for the expected number of transitions from state i to state j for O :

$$\sum_{t=1}^{T-1} \xi_{ij}(t).$$

For the estimation of HMM we obtain for the relative frequency spent in state i at time 1:

$$\pi_i = \gamma_i(1).$$

The quantity a_{ij}, which is an entry of the transition matrix, is the expected number of transition from state i to state j relative to the expected total number of transition away from state i:

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}.$$

And for discrete distributions, the quantity

$$b_i(k) = \frac{\sum_{t=1}^{T-1} \delta_{o_t,,v_k}\gamma_i(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

is the expected number of times the output observations have been equal to v_k while in state i relative to the expected total number of times in state i.

2.7 HMM-Gaussian - HMM with Continuous Output

In standard HMM the observation probability is discrete, in HMM Gaussian a Gaussian probability distribution function is assigned to each hidden state. Thus for a hidden state j, the pdf is

$$b_j(o_t) = \mathcal{N}(o_t|\mu_j\Sigma_j).$$

The parameter estimation for HMM-Gaussian changes little. The calculation of a_{ij} and $b_i(k)$ stays identical, but additionally it is necessary to calculate the parameters describing the parameters of each Gaussian distribution as follows during the maximization-step of the EM-algorithm:

$$\mu_i = \frac{\sum_{t=1}^{T-1} \gamma_i(t) o_t}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

and

$$\Sigma_i = \frac{\sum_{t=1}^{T-1} \gamma_i(t)(o_t - \mu_i)(o_t - \mu_i)^T}{\sum_{t=1}^{T-1} \gamma_i(t)}.$$

2.8 HMM-SDE

The HMM-SDE is a general approach to model the dynamics of a system. Each hidden state of the HMM has a stochastic differential equation assigned to it, thus generating the output probability of that hidden state. In full length this approach is presented in [7].

The stochastic differential equation (SDE) to approximate the effective dynamics is of the following type:

$$dx(t) = -\frac{\partial}{\partial x} V^{(q(t))} \cdot x(t) + \sigma^{(q(t))} dW(t).$$

Here

- $q(t)$: a Markov jump process with states $1, \ldots, M$,
- $W(t)$: standard Brownian motion,
- $\Sigma = (\sigma^{(1)}, \ldots, \sigma^{(M)})$ noise intensities and
- $V = (V^{(1)}, \ldots, V^{(M)}$ interaction potentials.

Within the SDE the variables V and σ are parameters wrt. the hidden state q.

Thus each SDE approximates the dynamics within a single metastable state. The setup of the potential V is a polynomial potential of the form

$$V^{(q)}(x) = \frac{1}{2} D^{(q)} (x - \mu^{(q)})^2 + V_0^{(q)}.$$

2.8.1 Propagation of Probability Density

In order to construct $\mathcal{L}$ appropriately one has to know the probability of output of state $x(t_j)$ under the condition of being in metastable state q_{t_j} for given parameters λ. One determines the probability by considering the propagation of probability densities by the SDE associated with metastable state q_{t_j}.

We consider a fixed state $q(t) = q$ for the time intervals t considered. For different realizations of the stochastic process W we consider a statistical density function $\rho(x,t)$ of an ensemble of SDE solutions. So we get an identical representation of the dynamics, the so called Fokker-Planck operator:

$$\partial_t \rho = \triangle_x V^{(q)}(x)\rho + \nabla_x V^{(q)}(x) \cdot \nabla_x \rho + B^{(q)} \triangle_x \rho,$$

where $B^{(q)}$denotes the variance of the white noise.

The ansatz for the solution is based on the superposition of Gaussian distributions:

$$\rho(x,t) = A(t) \exp(-(x - x(t))^T \Sigma(t)(x - x(t)). \tag{2.5}$$

This leads us to a system of three ordinary differential equations:

$$\dot{x} = -D^{(q)}(x - \mu^{(q)}),$$

$$\dot{\Sigma} = -2B^{(q)}\Sigma^2 + 2D^{(q)}\Sigma,$$

$$\dot{A} = (D^{(q)} - B^{(q)}\Sigma)A,$$

for the time-dependent parameters $\{x, \Sigma, A\}$. This system can be solved analytically. We consider the probability of output $O(t_{j+1})$ in metastable state $q_{t_{j+1}}$. Therefore, the output probability distribution results to be

$$\rho(O_{t_{j+1}}|q_{t_j}, O_{t_j}) = A(t_{j+1}) \exp(-(O_{t_{j+1}} - x(t_{j+1}))\Sigma(t_{j+1})(O_{t_{j+1}} - x(t_{j+1}))^T),$$

with

$$x(t_{j+1}) = \mu^q + \exp(-D^{(q)}\tau)(O(t_j) - \mu^q),$$

$$\Sigma(t_{j+1}) = (1 - \exp(-2D^{(q)}\tau))^{-1} D^{(q)} B^{(q)^{-1}},$$

$$A(t_{j+1}) = \frac{1}{\sqrt{\pi}} \Sigma(t_{j+1})^{\frac{1}{2}}$$

for metastable state $q = q_{t_{j+1}}$ and with $\tau = t_{j+1} - t_j$.

2.8.2 Likelihood

In order to be able to calculate the maximum-likelihood, which is required for the parameter estimation of the EM-Algorithm, we need the likelihood function.

First referring to the rate matrix R a transition matrix T can be obtained by

$$T = \exp(\tau R).$$

For the given model $\lambda = (\pi, T, x, \Sigma, A)$ with π the initial distribution of hidden states of the HMM, T the transition matrix and x, Σ and A as used as parameters in the ordinary differential equations. Thus the joint probability distribution for the observation and hidden state sequences is:

$$P(O, q|\lambda) = v(q_o)\rho(O_0|q_0)\prod_{t=1}^{T} T(q_{t-1}, q_t)\rho(O_t|q_t, O_{t-1}), \tag{2.6}$$

wherein ρ is the probability distribution:

$$\rho(O_t|q_t, O_{t-1}) = A^{(q_t)}(t)\exp(-(O_t - x^{(q_t)}(t))\Sigma^{(q_t)}(t)(O_t - x^{(q_t)}(t))^T). \tag{2.7}$$

The joint likelihood for the complete data is

$$\mathcal{L}(\lambda) = \mathcal{L}(\lambda|O, q) = P(O, q|\lambda).$$

2.8.3 Parameter Estimation

The key objective of the EM-algorithm is the expectation

$$Q(\lambda, \lambda_k) = E(\log P(O, q|\lambda)|O, \lambda_k), \tag{2.8}$$

where λ_k is the current parameter set. The expectation and maximization step of the EM-algorithm are performed as described above, the Q-function delivers the parameter estimates accordingly.

The calculation to obtain the Q-function is quite lengthy and will not be done here in full detail. Instead a short overview of the necessary steps will be given.

In the beginning some simplification is done through the use of Euler discretization:

$$x(t + \tau) = O_t - D^{(q)}(O_t - \mu^{(q)})\tau,$$

$$\Sigma(t + \tau) = \frac{1}{2\tau}B^{(q)^{-1}},$$

$$A(t + \tau) = \frac{1}{\sqrt{\pi}}\Sigma(t + \tau)^{\frac{1}{2}}.$$

Inserting these expressions into equation 2.7 one yields

$$\rho(O_t|q_t, O_{t-1}) = \frac{1}{(4\pi\tau^2)^{\frac{1}{4}}} B^{(q_t)^{-1/2}} \exp(-(O_t - x^{(q_t)})\frac{1}{2\tau} B^{(q_t)^{-1}}(O_t - x^{(q_t)})^T) \tag{2.9}$$

with $x^{(q_t)} = (O_{t-1} - D^{(q_t)}(O_{t-1} - \mu^{(q_t)})\tau)$.

Furthermore we proceed by inserting equation 2.6 into equation 2.9. This delivers the likelihood function with respect to the parameters μ, D and B. Finally to obtain the parameter estimates one has to insert this likelihood into equation 2.8 and calculate the partial derivatives $\frac{\partial Q}{\partial \mu}$, $\frac{\partial Q}{\partial D}$ and $\frac{\partial Q}{\partial B}$. Setting these partial derivatives to zero gives the final analytical expressions for the parameter estimates:

$$\mu^{(i)} = \frac{X_1 X_2 - X_3 X_4}{X_1 X_4 - X_3 X_5},$$

$$D^{(i)} = \frac{\sum_{t=2}^{T} \alpha_t(i)\beta_t(i)(O_t - O_{t-1})(O_{t-1} - \mu^{(i)})}{-\tau \sum_{t=2}^{T} \alpha_t(i)\beta_t(i)(O_{t-1} - \mu^{(i)})^2},$$

$$B^{(i)} = \frac{\sum_{t=2}^{T} \alpha_t(i)\beta_t(i)(-O_t + O_{t-1} - D^{(i)}(O_{t-1} - \mu^{(i)})\tau)^2}{\tau \sum_{t=0}^{T} \alpha_t(i)\beta_t(i)}$$

and with the abbreviations

$$\begin{aligned}
X_1 &= \sum_{t=1}^{T} \alpha_t(i)\beta_t(i)(O_t - O_{t-1}), \\
X_2 &= \sum_{t=1}^{T} \alpha_t(i)\beta_t(i)O_{t-1}^2, \\
X_3 &= \sum_{t=1}^{T} \alpha_t(i)\beta_t(i)(O_t - O_{t-1})O_{t-1}, \\
X_4 &= \sum_{t=1}^{T} \alpha_t(i)\beta_t(i)O_{t-1}, \\
X_5 &= \sum_{t=1}^{T} \alpha_t(i)\beta_t(i).
\end{aligned}$$

2.9 HMM-VAR as Generalization of HMM-SDE

The parameter estimation of a one dimensional SDE has been described in detail in the previous section. In this section the generalization of the concept behind HMM-SDE is investigated. Extending the HMM-SDE approach, based on a one-dimensional SDE, to a d-dimensional SDE of the form

$$\dot{z} = F(z - \mu) + \Sigma\dot{W}$$

the parameter estimation is obtained by investigation of an appropriate likelihood function, this extension is performed in [16]. A more generalized approach is based on VAR processes ([13]), here the interpretation of a SDE as VAR process is given and a comprehension is presented. An detailed analysis of the difficulties of parameter estimation of generalized multidimensional Langevin processes is given in [8].

The trajectory z_t with $t \in \{1, \ldots, T'\}$ is given at discrete points in time. According to the above equation and given an observation z_t, the conditional probability density of z_{t+1} is a Gaussian with density function

$$f_\lambda(z_{t+1}|z_t) = \frac{1}{\sqrt{|2\pi R(\tau)|}} \cdot \exp(-\frac{1}{2}(z_{t+1} - \mu_t)^T R(\tau)^{-1}(z_{t+1} - \mu_t)) \tag{2.10}$$

with the variance of the distribution given as

$$\mu_t := \mu + \exp(\tau F)(z_t - \mu)$$

and

$$R(\tau) := \int \exp(sF)\Sigma\Sigma^T \exp(sF^T)\, ds.$$

A likelihood function with parameter set $\lambda = (\mu, F, \Sigma)$ can be constructed as

$$L(\lambda|Z) = \prod_{t=1}^{T'-1} f_\lambda(z_{t+1}|z_t).$$

But unfortunately this likelihood function can not be solved analytically. Therefore, we express Equation 2.10 by using the definition of μ_t as

$$\begin{aligned} z_{t+1} &= \mathcal{N}(\mu + \exp(\tau F)(z_t - \mu), R) \\ &= (I - \exp(\tau F))\mu + \exp(\tau F) z_t + \mathcal{N}(0, R), \end{aligned}$$

where $\mathcal{N}$ is a multivariate normal distribution and I is an identify matrix. Using the definitions

$$\begin{aligned} \Phi &:= \begin{pmatrix} (I - \exp(\tau F))\mu & \exp(\tau F) \end{pmatrix} \in \mathbb{R}^{d \times (d+1)} \\ X &:= \begin{pmatrix} 1 & \cdots & 1 \\ z_1 & \cdots & z_{T-1} \end{pmatrix} \in \mathbb{R}^{(d+1) \times (T'-1)} \\ Y &:= \begin{pmatrix} z_2 & \cdots & z_T \end{pmatrix} \in \mathbb{R}^{d \times (T'-1)} \\ \epsilon &:= \begin{pmatrix} \mathcal{N}(0, R) & \cdots & \mathcal{N}(0, R) \end{pmatrix} \in \mathbb{R}^{d \times (T'-1)} \end{aligned}$$

allows to write

$$Y = \Phi X + \epsilon.$$

A transformation of the parameter set λ to $\lambda' = (\Phi, R)$ leads to the likelihood function

$$L(\lambda'|Z) = \left(\frac{1}{\sqrt{2\pi R}}\right)^{T'-1} \exp(-\frac{1}{2} tr(Y - \Phi X)(Y - \Phi X)^T R^{-1})$$

for which maximum likelihood estimators are available:

$$\Phi = YX^T(XX^T)^{-1} \tag{2.11}$$

and

$$R = \frac{(Y - \Phi X)(Y - \Phi X)^T}{T' - 1}. \tag{2.12}$$

2.9.1 Estimator Calculation

The analytic estimators given in Equation 2.11 and 2.12 are in general not used for the computation of the parameters, as the required matrix inversion can be numerically unstable. Instead a moment matrix is constructed

$$\begin{aligned} M(Z) &= \sum_{i=1}^{T} \begin{pmatrix} 1 \\ z_i \\ \vdots \\ z_{i+p} \end{pmatrix} \cdot \begin{pmatrix} 1 & z_i^T & \dots & z_{i+p}^T \end{pmatrix} \\ &= \begin{pmatrix} XX^T & XY^T \\ YX^T & YY^T \end{pmatrix} =: \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}. \end{aligned}$$

The moment matrix is an important object, since it contains all statistical relevant information about the observed process. By rewriting the likelihood in terms of M this fact becomes evident:

$$\begin{aligned} L(\Phi, R|Z) &= L(\Phi, R|M) \\ &= \left(\frac{1}{\sqrt{2\pi R}}\right)^{m} \cdot \exp\left(-\frac{1}{2} tr(\,(M_{22} - M_{21}\Phi^T - \Phi M_{12} + \Phi M_{22}\Phi^T)R^{-1}\,)\right). \end{aligned}$$

A Cholesky decomposition of the moment matrix M yields

$$\begin{aligned} M &= \begin{pmatrix} XX^T & XY^T \\ YX^T & YY^T \end{pmatrix} \\ &= RR^T = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \cdot \begin{pmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{pmatrix} = \begin{pmatrix} R_{11}^T R_{11} & R_{11}^T R_{12} \\ R_{12}^T R_{11} & (R_{12}^T R_{12} + R_{22}^T R_{22}) \end{pmatrix}. \end{aligned}$$

Thus the estimators Φ and R can be expressed as

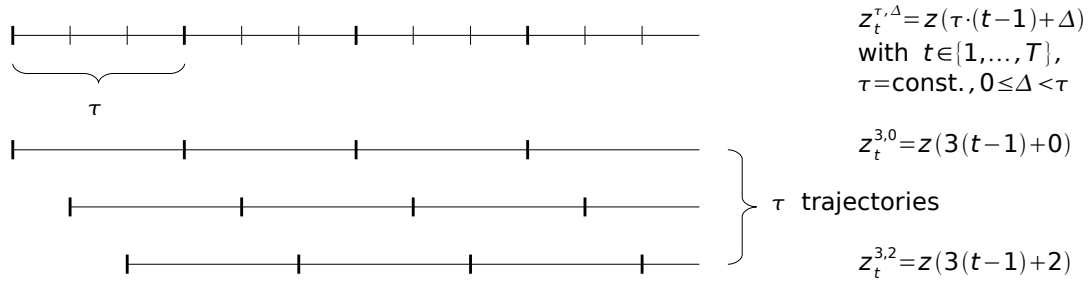

Figure 2.3: Original trajectory and τ shifted trajectories.

$$\Phi = \left(R_{11}^T R_{12}\right)^T$$

and

$$R = \frac{1}{m} \cdot R_{22}^T R_{22}.$$

The derivation of HMM-VAR based on VAR processes presented here is based on personal communication with Eike Meerbach.

2.10 Using HMM-VAR for Trajectories with Different Lagtimes

Given a trajectory z_t, where the index t indicates the discrete character of the trajectory, the requirement is often to analyze the trajectory for different lagtimes τ. This requirement stems from the fact that only for sufficiently large lagtimes τ the Markov model obtained from clustering fulfills the Markovian property. So far, taking different lagtimes into account, the resulting trajectory for lagtime τ is given as $z_t^\tau := z(\tau(t-1))$. Intuitively spoken, only each multiple of τ value is taken from the trajectory, e.g. with $\tau = 100$ and $t \in \{1, \ldots, 1000\}$ only $z_1^{100} = z(0)$, $z_2^{100} = z(100), \ldots, z(900)$, thus the obtained trajectory is sparse.

Without discarding information the only opportunity is also to take the shifted trajectories $z_t^{\tau,\Delta} := z(\tau(t-1)+\Delta)$ for all $0 \leq \Delta < \tau$ into account. In total for a certain lagtime τ there exist τ such shifted trajectories (see Figure 2.3).

But one problem which still remains is to prepare a suitable input for HMM-VAR from the shifted trajectories. Two solutions are reasonable:

- Concatenation of all shifted trajectory pieces $z_t^{\tau,0}, \ldots z_t^{\tau,\tau-1}$. A simple concatenation will not suffice. Oriented on the example above suppose $z_t^{100,0}$ and $z_t^{100,1}$ are the first two trajectory pieces, thus simple concatenation yields

$$\begin{aligned} & z_1^{100,0}, z_2^{100,0}, \ldots, z_9^{100,0}, z_1^{100,1}, z_2^{100,1}, \ldots, z_9^{100,1} \\ = \; & z(0), z(100), \ldots, z(900), z(1), z(101), \ldots, z(901) \end{aligned}$$

Thus at the glueing point $z(900)$, $z(1)$ an additional transition might have been introduced for the underlying Markov model. Therefore, imagine $z(0)$, $z(1)$ live in the conformational state 1, while $z(900)$ lives in a conformational state, which is not state 1. Thus a transition is induced, which especially in case of statistically rare events is inacceptable.
Therefore, the only way is to adapt the HMM-VAR accordingly, so that special care is taking at the glueing points. This involves the neglect of these transitions for the observation as well as the neglect within the Markov model of the HMM. Without going into much detail here, this step requires serious bookkeeping. Due to the error-prone character this approach is not chosen.

- The second solution takes multiple trajectories as input for HMM-VAR. At this point the set of trajectories consists of τ shifted trajectories. Instead of a concatenation of trajectory pieces, HMM-VAR is adapted to simultaneously calculate model parameters λ for all trajectory pieces. This approach is presented below.

2.10.1 Extension of HMM-VAR to Multiple Trajectories

The necessity to perform HMM-VAR simultaneously on multiple trajectories requires some modifications to the HMM-VAR, which operates on a single trajectory. Here, we adapt our notation. The set of m trajectories as input for HMM-VAR is $\{z^k | 0 \leq k < m\}$.

Within this approach, the trajectories z^i are treated independently. The calculation of the forward variables $\alpha_i(t)$, where i is the index for the hidden state and t is the time step, is performed independently for each trajectory z^k. So, we get forward variables $\alpha_i^k(t)$, which depend on the trajectory index k as well. Accordingly the backward variables $\beta_i^k(t)$ are calculated. The forward and backward variables are used to calculate $\gamma_i^k(t)$, the probability to be in state i at time t for trajectory k. The transition probabilities $\xi_{ij}^k(t)$ are calculated appropriately.

Given $\alpha_i^k(t)$, $\beta_i^k(t)$, $\gamma_i^k(t)$ and $\xi_{ij}^k(t)$ as above the calculation of the transition matrix, the initial distribution and the moment matrix is possible. Here, for the calculation of these entities we sum over the transition probabilities for all k. The transition matrix is obtained by

$$a_{ij} = \frac{\sum_{k=0}^{m-1} \sum_{t=1}^{T-1} \xi_{ij}^k(t)}{\sum_{k=0}^{m-1} \sum_{t=1}^{T-1} \gamma_i^k(t)},$$

and the initial distribution by

$$\pi_i = \sum_{k=0}^{m-1} \gamma_i^k(1)$$

and appropriate normalization. At this point, it needs to be mentioned, that a proper justification for a common initial distribution π is required.

For all trajectories z^k a common parameter estimation is performed by the following two steps:

- Calculate moment matrix M with appropriate weighting γ. All statistical relevant information is contained within M. The information coming from different points in time is weighted by γ.

- Calculation of estimators from moment matrix M.

The approach presented here does not conform to mathematical strictness. A clean derivation is required and will be part of future work.

2.11 Perron-Cluster-Cluster-Analysis (PCCA)

PCCA is a method for the determination of metastable states based on a transition matrix ([3, 23, 22, 17]). Therefore, PCCA transforms that transition matrix into a block-diagonal form by permutation. The transition matrix is obtained by counting the transitions between microstates (see Section 2.3). By PCCA each of the microstates is assigned to one of 1,..,C clusters or metastable states. The assignment of macrostates to microstates is called the membership-assignment.

2.11.1 Mathematical Idea

Given a transition matrix T with a real-valued spectrum with sorted Eigenvalues $\lambda_1 \geq \ldots \geq \lambda_s$ and an Eigenvector matrix X, where the i^{th}column of X is an Eigenvector corresponding to λ_i, the clustering can be computed in terms of the basis X as

$$\chi_{dis} = XA,$$

with $A \in \mathcal{R}^{s \times n_c}$.

2.11.2 Derivation

For the derivation of PCCA we need a special kind of functions ξ_j and a disjoint decomposition χ_i of our space Ω. The functions $\{\xi_1, \ldots, \xi_s\} : \Omega \to \mathbb{R}$ are a partition of unity and non-negative. They form a so called membership basis of Ω and are usually characteristic functions that define microstates

$$\sum_{i=1}^{s} \xi_i(q) = 1, \quad \forall q \in \Omega.$$

The $\{\chi_1, \ldots, \chi_c\}$ are almost characteristic functions and also are a partition of unity, that decompose the space Ω into c (almost) disjoint subsets, that can be referred to as macrostates. To discretize the macrostate basis χ_i the microstate basis ξ_i is used

$$\chi_l(q) = \sum_{i=1}^{s} \xi_i(q) \chi_{disc}(i, l).$$

The goal will be to find for a given set of microstates the optimal discretization matrix χ_{disc}. So we can give two transition matrices. One for the microstates, one for the macrostates:

$$T_{ij}^{macro} = \frac{\langle \chi_i, P^\tau \chi_j \rangle_\pi}{\langle \chi_i \rangle_\pi},$$

$$P = T_{ij}^{micro} = \frac{\langle \xi_i, P^\tau \xi_j \rangle_\pi}{\langle \xi_i \rangle_\pi}.$$

If we have a metastable subset, that means

$$\chi_l \approx \mathcal{P}\chi_l,$$

it follows easily that

$$P\chi_{disc} \approx \chi_{disc}$$

has also to be true. If the χ_l should still be non negative and be a partition of unity, it follows for the discretization χ_{disc}, that it has to be a stochastic matrix. For PCCA it is reasonable to express the discretization in terms of the first c eigenvectors of the microstate transition matrix

$$\chi_{disc} = X \cdot \mathcal{A}.$$

In this case to keep the properties of χ_{disc} these transfer similar to $\mathcal{A}$. After all this transformations the question still remains, which discretization or which $\mathcal{A}$ out of the feasible set of allowed matrices to choose. For this reason we choose an objective function $I(\mathcal{A})$, which is to be maximized to find $\mathcal{A}$. There are two prominent choices:

1. For each cluster / microstate there should be a point $q \in \Omega$ with maximal degree of membership $\chi_i(q) \approx 1$:

$$I_1(\mathcal{A}) = \sum_{j=1}^{c} \max_{l=1,\ldots,s} \chi_{disc}(l,j) = \sum_{j=1}^{c} \max_{l=1,\ldots,s} \sum_{i=1}^{c} X(l,i)\mathcal{A}(i,j) \leq c.$$

2. Maximize the metastability of the conformations $\chi_1, \ldots, \chi_c$, which equivalent to maximize the trace of the macrostate transition matrix

$$I_2(\mathcal{A}) = trace(P^{macro}) = trace(\widetilde{D}^{-1}\mathcal{A}^T\Lambda\mathcal{A}) \leq \sum_{i=1}^{c} \lambda_i.$$

Once a solution is found it is an optimal clustering. The solution can be found using Linear Programming, but it is not always feasible. In general any optimization technique can be used.

Another more intuitive approach to PCCA is given by the idea, that each stochastic matrix inhibits a Perron Eigenvalue of one with a constant right eigenvector. If we have a transition matrix with c completely decoupled subset, then this is true for each subset. If we have

a first order approximation to this, in other terms a transition matrix with metastability, then, to first order approximation. The eigenvectors will be a linear combination of the characteristic functions for each metastable subset and the eigenvalues will be close to one. This means that the number of eigenvalues close to one are in indication for the number of metastable sets in the system.

Looking at the transition matrix as a generator of a Markov chain, we can decompose the transition matrix in s matrices of rank one, which describe each a movement of probability in the system, where the eigenvalues give the timescale of the process. In detail this is minus the inverse of the log of the rate. So eigenvalues close to one indicate nearly infinite lifetimes of processes.

2.12 Principal-Component Analysis (PCA)

Principal-component analysis (PCA) (see [11], Chapter 2) is a vector space transform used to reduce multidimensional data sets to lower dimension. Mathematically PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

2.12.1 Covariance Matrix

Consider two sets of data, written as row vectors,

$$\vec{a}' = [a'_1\ a'_2\ \ldots\ a'_n]; \quad \vec{b}' = [b'_1\ \ldots\ b'_n].$$

Performing a mean correction (zero mean correction) by the mean of $\bar{a} = \sum_{i=1}^{n} a_i$ and with $\Delta a_i = \bar{a} - a_i$, we obtain vectors $\vec{a}$ and $\vec{b}$

$$\vec{a} = [\Delta a_1\ \ldots\ \Delta a_n]; \quad \vec{b} = [\Delta b_1\ \ldots\ \Delta b_n].$$

The variance of $\vec{a}$ is defined as

$$\sigma_a^2 = < a_i \cdot a_i > = \frac{1}{n-1} \sum_{i=1}^{n} a_i \cdot a_i = \frac{1}{n-1}\, \vec{a} \cdot \vec{a}^T.$$

In the same manner the covariance between $\vec{a}$ and $\vec{b}$ is defined as

$$\sigma_{ab}^2 = < a_i \cdot b_i > = \frac{1}{n-1} \sum_{i=1}^{n} a_i \cdot b_i = \frac{1}{n-1}\, \vec{a} \cdot \vec{b}^T.$$

Extending these definitions to matrix calculus and setting up a matrix X, whose rows consists of some data vectors (for the sake of simplicity we take $\vec{a}$, $\vec{b}$ and some further

vectors up to $\vec{z}$):

$$X = \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vdots \\ \vec{z} \end{bmatrix}.$$

One obtains the covariance matrix as

$$C_X = \frac{1}{n-1} XX^T = \begin{pmatrix} \sigma_a^2 & \sigma_{ab}^2 & \cdots & \sigma_{az}^2 \\ \sigma_{ab}^2 & \sigma_b^2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{az}^2 & \cdots & \cdots & \sigma_z^2 \end{pmatrix}. \tag{2.13}$$

The following properties can be easily derived from Equation 2.13:

- C_X is a square symmetric matrix,
- the diagonal terms are the variances of the data vectors,
- the off-diagonal terms are the covariances of two different data vectors.

Thus, the covariance matrix C_X captures the correlations between all possible pairs of data vectors. The correlation values reflect the noise and redundancy of the data. In the diagonal terms, large (small) values correspond to interesting dynamics (noise). In the off-diagonal terms large (small) values correspond to high (low) redundancy.

The main purpose of PCA is now to

- minimize redundancy, measured by covariance and
- maximize the signal, measured by variance

of C_X by using some linear orthogonal transform. So the new "optimized" and transformed matrix C_Y has off-diagonal terms, which are 0. Thus C_Y has to be a diagonal matrix.

2.12.2 Diagonalization of Covariance Matrix

Mathematically speaking, our purpose is to find an orthonormal matrix P, where $Y = PX$ such that

$$C_Y = \frac{1}{n-1} YY^T$$

is diagonal. Inserting PX for Y yields

$$\begin{aligned} C_Y &= \frac{1}{n-1} (PX)(PX)^T \\ &= \frac{1}{n-1} (PX)(X^T P^T) \\ &= \frac{1}{n-1} P(XX^T)P^T \\ &= \frac{1}{n-1} PAP^T. \end{aligned} \tag{2.14}$$

Note that the introduced matrix $A = XX^T$ is symmetric. Since A is symmetric, it is orthogonally diagonalizable. Furthermore it is diagonalized by a matrix of its orthonormal eigenvectors (no prove given here). Thus the matrix A can be written as

$$A = EDE^T, \tag{2.15}$$

where E is a matrix of Eigenvectors arranged as columns and D is a diagonal matrix.

We choose the transformation matrix P in Equation 2.14 to be a matrix, where each row of P is an Eigenvector of $A = XX^T$. Thus $P = E^T$ and substituting P into Equation 2.15 yields

$$A = P^T DP. \tag{2.16}$$

Finally inserting A into Equation 2.14 one obtains

$$\begin{aligned} C_Y &= \frac{1}{n-1} PAP^T \\ &= \frac{1}{n-1} P(P^T DP)P^T \\ &= \frac{1}{n-1} (PP^T)D(PP^T) \\ &= \frac{1}{n-1} (PP^{-1})D(PP^{-1}) \\ C_Y &= \frac{1}{n-1} D. \end{aligned}$$

With this choice of matrix P, whose rows are the Eigenvectors of XX^T at the same time being the principal components of X, we have found the transformation, which diagonalizes C_Y. Furthermore, the i^{th}diagonal value of C_Y is the variance of X.

2.13 K-means Clustering

The k-means algorithm (Lloyd 1957) belongs to the group of model-free method for object classifications [6]. The algorithm is also known as Linde-Buzo-Gray (LBG) or Lloyd algorithm. The basic idea is to group the data points in such a way to m clusters that the sum of distances between each data point and its respective cluster center is minimized.

Following assumptions are made:

- Input data has to have a naturally way of clustering (clusters have to be available in the data).
- The total number of clusters n is known (sometimes the shortcut k is used).
- A function c is known to determine the center (centroid) for a given set of data points.

The following heuristic is used to solve the k-means problem:

1. Given: data set $C = \{x_1, \ldots, x_m\}$, total number of clusters is n and function c
2. Initialization: partition input points into n initial sets (either random or using some prior knowledge)
3. Calculate: Use function c to calculate the center for each set $\Rightarrow$ Set the cluster centroid to this point
4. Assignment: Data points x_i are associated with the nearest centroid $\Rightarrow$ Data is arranged into new partitions
5. Recalculate: Positions of centroids are recalculated for the new partitions
6. Iterate: Go to (4) until convergence (points do no longer switch clusters or centroids are no longer changed)
7. END

An example of the mode of operation of the k-means algorithm is given in Figure 2.4.

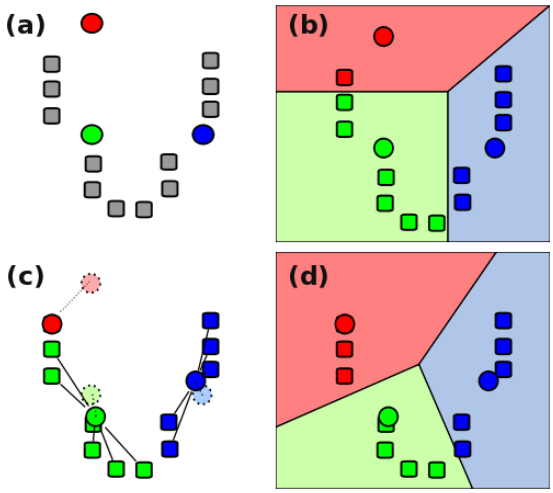

Figure 2.4: Example for k-means algorithm: (a) Shows the initial randomized centroids and a number of points. (b) Points are associated with the nearest centroid. (c) Now the centroids are moved to the center of their respective clusters. (d) Steps 4 and 5 are repeated until a suitable level of convergence has been reached. (Picture taken from `http://en.wikipedia.org/wiki/K-means_algorithm`).

The k-means algorithm has following advantages:

- Algorithm is extremely fast,
- algorithm is simple.

But the k-means algorithm has the following disadvantages:

- The algorithm can get stuck in local minima, it is not guaranteed to return the global optimum solution.
- Different initial sets of clusters (random component in algorithm) for same data set C can lead to different clustering results.
- The quality of final solution may depends strongly on the initial set of clusters (step 2). Because the algorithm is extremely fast, a common method is to run the algorithm with different settings and return the best clustering found.
- The main drawback is that the number of cluster m has to be known beforehand. A simple solution would be to compare the results of multiple runs with different m.
- If the data set C has no natural clusters, one obtains strange results.

3 PCCA and HMM on Model Examples

This chapter is dedicated to the analysis of a model trajectory and to the comparison of the clustering results of PCCA and HMM-VAR. A model trajectory based on a 3-basin potential is chosen to obtain an overview of the performance as well as to get an impression of the pitfalls which can occur during the application of PCCA and HMM-VAR.

3.1 Approach

An overview of the general approach for the evaluation and the comparison of the clustering results is given in Figure 3.1. Based on different potentials U, different model trajectories $z(t)$ are created. These trajectories are clustered with PCCA and HMM-VAR.

First and according to the above approach, a model trajectory is investigated where clustering is successful for both PCCA and HMM-VAR. It is shown, that the correct initialization of HMM-VAR has an serious impact on the convergence of HMM-VAR towards an optimal solution.

Second, the investigation of a model trajectory is performed where HMM-VAR fails. The type and shape of the potential U and thereby the kinetics of metastable states have much influence on the clustering results.

3.2 Model Trajectory Generation and Potential

For the analysis a model trajectory is created, which is based on an overdamped Langevin process

$$\gamma \dot{z} = -\nabla U(z) + \sigma \dot{W}.$$

Using Euler discretization the Langevin process can be expressed as

$$\begin{aligned} \Delta z = z(t+\Delta t) - z(t) &= \frac{1}{\gamma}(-\nabla U(z) + \sigma \dot{W}) \cdot \Delta t \\ z(t+\Delta t) &= z(t) + \frac{1}{\gamma}(-\nabla U(z) + \sigma \dot{W}) \cdot \Delta t. \end{aligned} \tag{3.1}$$

Here, z is a position vector depending on time t, $\dot{z}$ is the time derivative wrt. z, $U(z)$ is the potential at location z, $\dot{W}$ denotes some random noise and σ is the appropriate noise intensity.

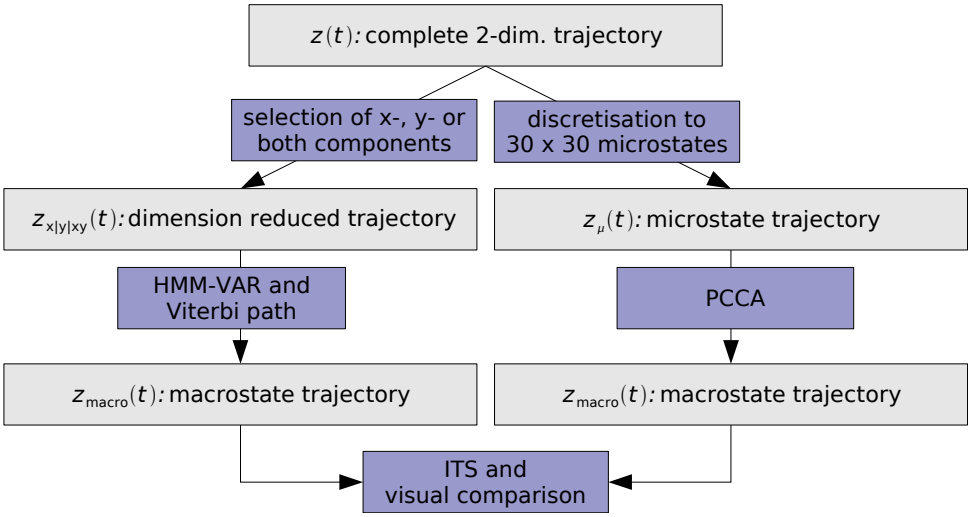

Figure 3.1: General approach for the evaluation and the comparison of clustering results from PCCA and HMM-VAR for model trajectories. The left branch shows the HMM-VAR part, the right branch shows the PCCA part.

The analysis is performed with a 2-dimensional trajectory $z(t)$, thus $z(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$. The potential $U(z)$ is a function $U(z) : \mathbb{R}^2 \to \mathbb{R}$. For the analysis a 3-well potential chosen. Basically the potential is defined as a sum of Gaussian potential basins as

$$U(z) = U(x, y) = \sum_{i=1}^{p} -I_i \cdot \exp\left(-\left(\left(\frac{\mu_{x,i} - x}{\delta_{x,i}}\right)^2 + \left(\frac{\mu_{y,i} - y}{\delta_{y,i}}\right)^2\right)\right),$$

where p is the number of parameter sets. Each parameter set contains I, μ_x, μ_y, δ_x, δ_y. The parameters for the potentials are given in Figure 3.3.

3.3 3-Well Potential

3.3.1 Parameters for Trajectory Generation

According to the potential $U_{\text{type}}(z)$, given in Figure 3.2, top left, a two dimensional trajectory $z(t)$ is generated based on the discretized Langevin process given in equation 3.1.

The following parameters are used for the generation of the trajectory:

- The starting point $z_0 = z(t_o)$ is chosen to be at $z_0 = (20, 10)^T$, thus the trajectory starts in the bottom right minimum.
- The time step for the discretization is 0.6.
- The noise intensity σ is 1.7, the random noise $\dot{W}$ is setup as a two dimensional Gaussian normal distribution $\mathcal{N}^2(0, 1)$.

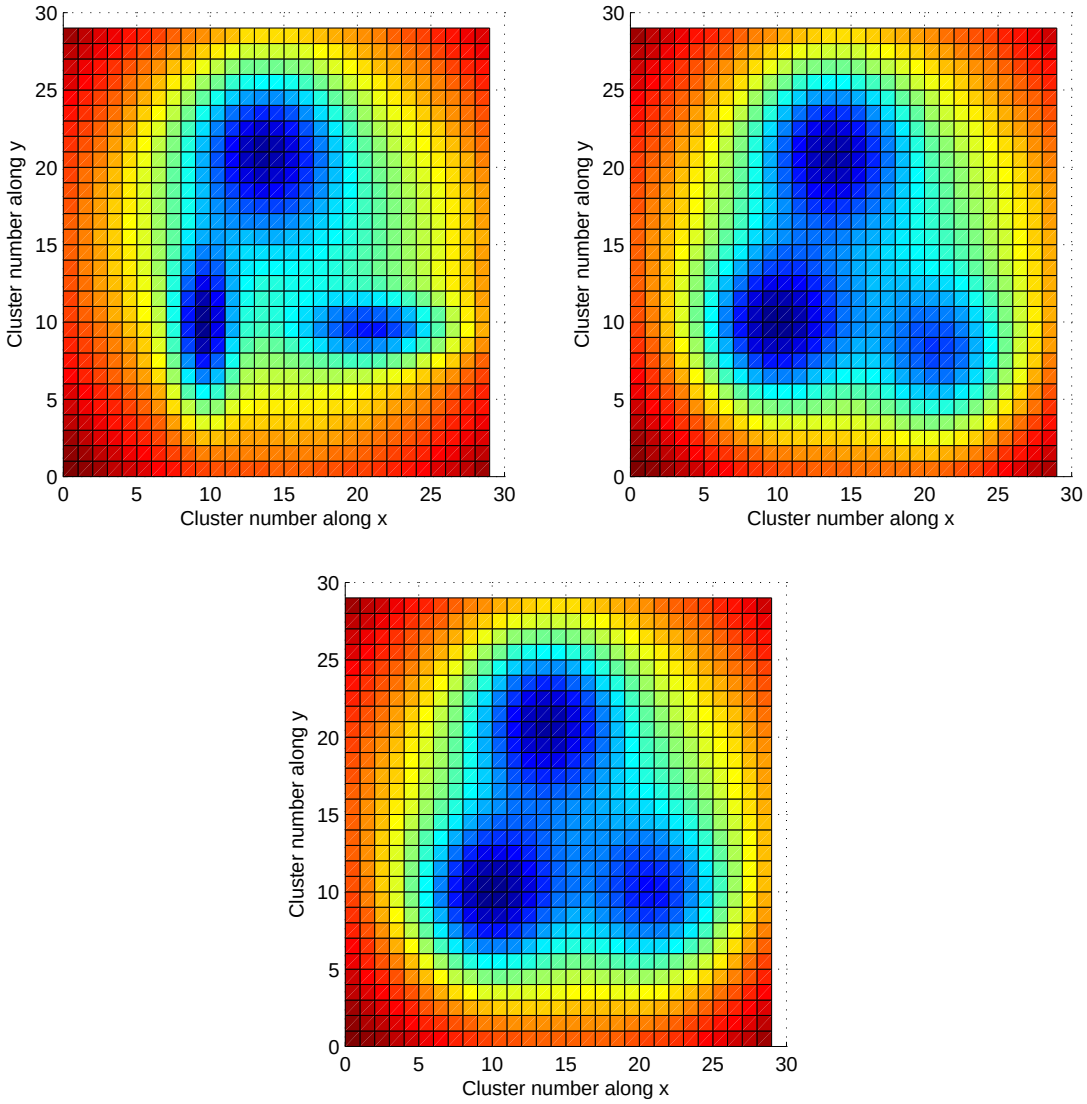

Figure 3.2: Different 3-well potentials. Top-left: 3-well potential $U_{\mathrm{type}}(z)$ with two potentials basins at $y = 9$, which differ in shape. Top-right: 3-well potential $U_{\mathrm{shifted}}(z)$, where the two bottom potentials are slightly shifted to each other. Bottom: 3-well potential $U_{\mathrm{std}}(z)$.

	I	μ_x	μ_y	δ_x	δ_y	Remarks
U_{std}	2	15	15	20	20	global basin
	1.2	9	9	5	5	bottom left basin
	0.9	21	9	5	5	bottom right basin
	1.0	13	21	5	5	top basin
U_{type}						all other parameters like U_{std}
				2	5	different potential type
				5	2	different potential type
U_{shifted}						all other parameters like U_{std}
			7			bottom right basin is shifted in y

Figure 3.3: Parameters I, μ_x, δ_x, μ_y, δ_y in parameter sets for the different potentials.

- The trajectory has $5 \cdot 10^6$ time steps.

The generated trajectory is displayed in Figure 3.4.

3.3.2 Microstate-Clustering and True Timescales

The two dimensional continuous trajectory $z(t)$ is clustered into a total of 900 microstates, 30 states along x- and y-axis respectively, resulting in a microstate trajectory $z_\mu(t)$. Therefore, a function $f : \mathcal{R}^2 \to A$, with $A = \{0, \ldots, 899\}$ and $m(t) \in A$ is used, which maps each point $z(t)$ into its appropriate microstate. Thus the continuous trajectory is discretized along the coordinate axis.

A transition matrix $T(\tau)$ is constructed from the microstate trajectory as described in Section 2.3: the total number of transitions from microstate i to microstate j is counted in

$$c_{ij} = \mid \{i = m(t), j = m(t+\tau) | t = 0, \ldots, t_{max}\} \mid$$

and the total number of transitions leaving microstate i is

$$c_i := \sum_{k=1}^{m} c_{ik}.$$

Thus the transition matrix entry t_{ij} is given by

$$t_{ij} = \frac{c_{ij}}{c_i}.$$

In the case of 900 microstates this results in a transition matrix $T(\tau)$ being in $\mathbb{R}^{900\times900}$.

For different lagtimes τ the implied timescales have been calculated. By calculating the first eigenvalues of $T(\tau)$ the implied timescale (ITS) is obtained as

$$ITS_i(\tau) = -\frac{\tau}{\log \lambda_i}.$$

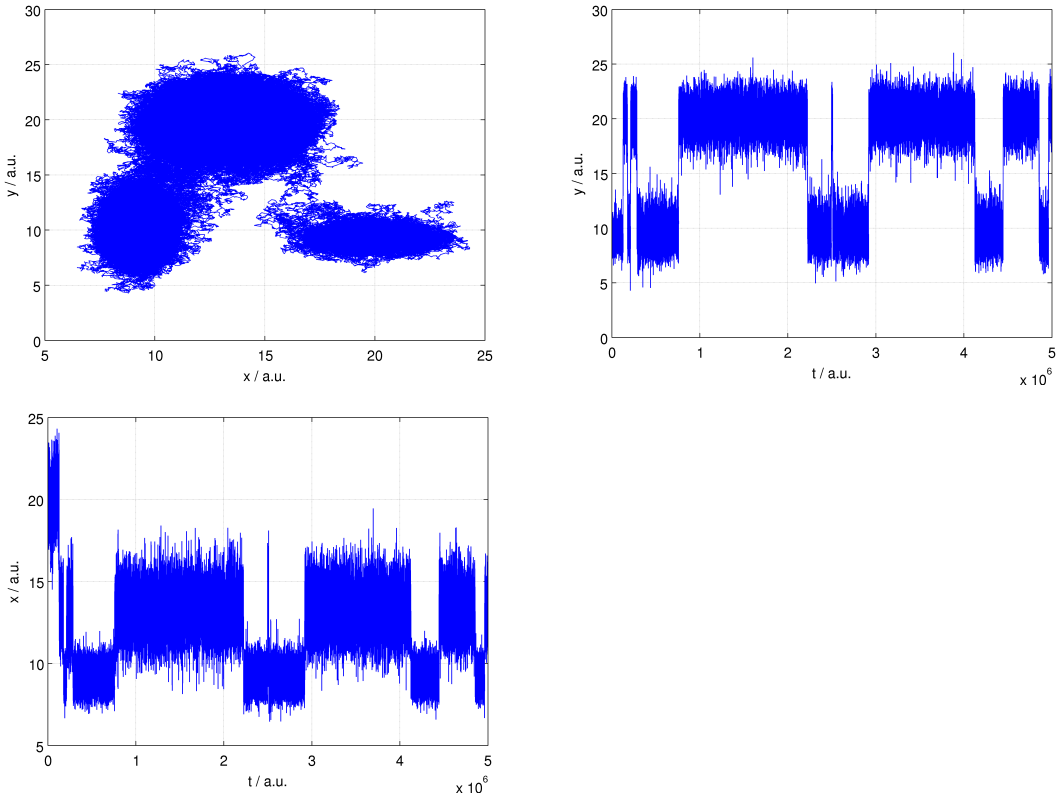

Figure 3.4: A plot of the trajectory $z(t)$ for the potential $U_{\text{type}}(z)$ (see Figure 3.2). Top-left: Visited points of $z(t)$ for $t = 0$ to $t = 5 \cdot 10^6$. Top-right: Projection of the trajectory $z(t)$ onto the y-axis. Bottom: Projection of the trajectory $z(t)$ onto the x-axis.

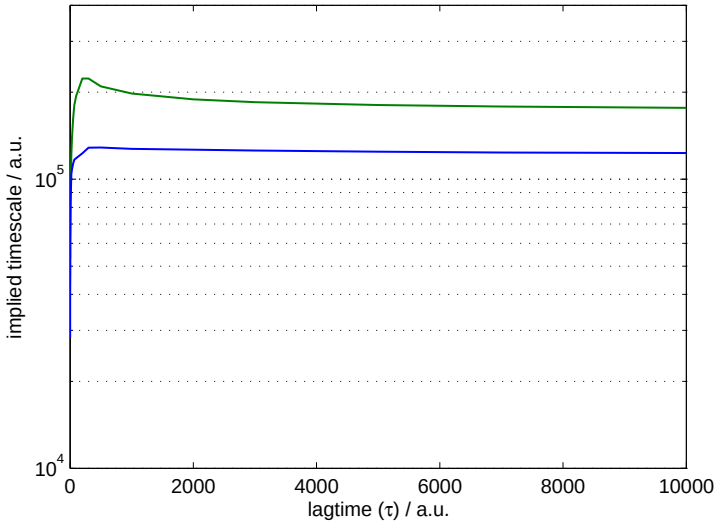

Figure 3.5: True timescales determined by the eigenvalues of the transition matrix $T(\tau)$. The green curve displays the ITS calculated by the 2^{nd} eigenvalue, the blue curve display the ITS calculated by the 3^{rd} eigenvalue. The slower a switching process is, the higher is its ITS.

The resulting ITS are determined for different lagtimes and are plotted in Figure 3.5.

3.3.3 PCCA: Clustering Results and Timescales.

The transition matrix $T(\tau)$ is built from the microstate or discretized trajectory. To sum up the insights given in Section 2.11, PCCA identifies the metastable states. A metastable state is a set of microstates, in whom the system stays a long time before leaving that state. Thus, given a discrete trajectory $z_\mu(t) \in A = \{\text{microstates } a\}$, PCCA determines a partition of microstates into $|C|$ metastable sets, such that the number of transitions within these sets is maximized and the number of transitions to different sets is minimized. In the following C is the set of macrostates, each $c \in C$ is a set of microstates $\{a_i, \ldots, a_j\}$. To determine the partition of microstates in macrostates PCCA transforms the transition matrix $T(\tau)$ into block-diagonal form by permutation without changing the eigenvalues. These blocks of the transition matrix "build" macrostates. Each of the microstates is assigned to one of the metastable sets, thus PCCA gives an assignment $A \to C$.

In case of the given model trajectory a total number of three macrostates is evident, since the potential surface consists of three minima. The clustering results of PCCA are given in Figure 3.6 and 3.7.

With PCCA delivering the assignment of microstates to macrostates a macrostate trajectory $z_{\text{macro}}(t)$ is constructed. This trajectory is taken to build a transition matrix $T(\tau) \in \mathcal{R}^{3\times 3}$ following the same procedure of counting transitions, but this time between macrostates. By calculating the eigenvalues of the transition matrix for different lagtimes $\tau \in \{1, 10, 100, 1000, 2000, 5000, 7000, 10000\}$, the implied timescales are computed, see Figure 3.8.

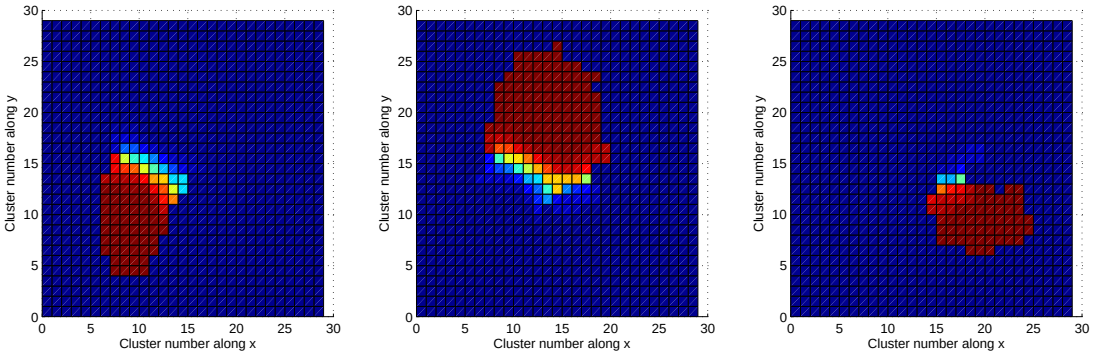

Figure 3.6: PCCA clustering results of model trajectory for lagtime 1.

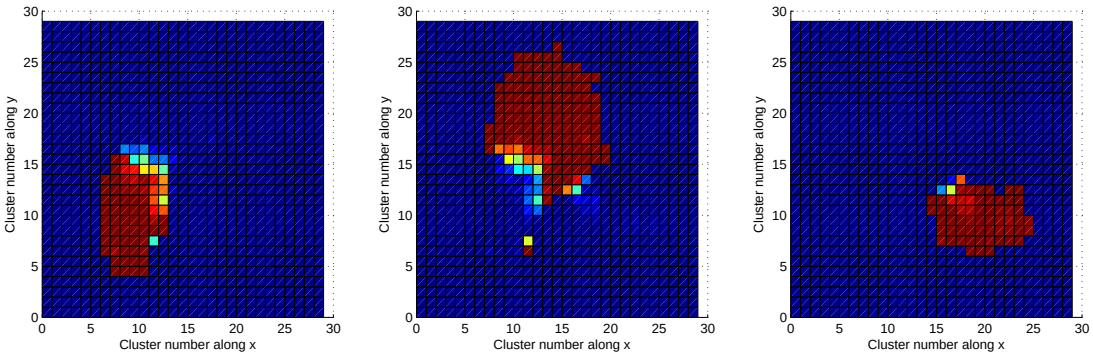

Figure 3.7: PCCA clustering results of model trajectory for lagtime 1000.

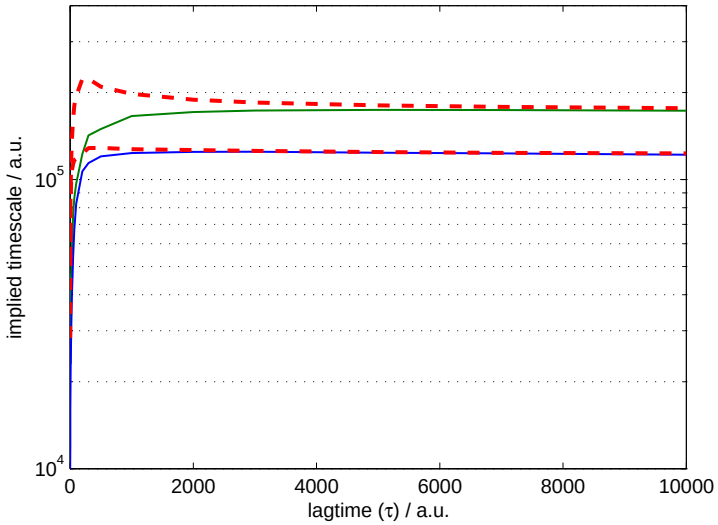

Figure 3.8: Implied timescales for PCCA clustering of the 2^{nd} and the 3^{rd} eigenvalue. The red dashed curves indicate the true timescales. Throughout the whole presentation of timescales, we will indicated reference true timescales in red.

3.3.4 HMM-VAR starting from "correct" Assignment of Viterbi Path

The model trajectory $z(t)$ is taken as input for HMM-VAR. The memory p is set to one, thus HMM-VAR is identical to HMM-SDE. This setting is kept for all clusterings made by HMM-VAR. One of the benefits of HMM-VAR is its ability to work on a reduced state space. Thus the two-dimensional trajectory $z(t)$ is projected onto a subspace so HMM-VAR can reveal its capability. For the sake of simplicity this subspace is either the x-component or the y-component of the trajectory $z(t)$. Nevertheless and for reasons of completeness, HMM-VAR is applied to the (full) two-dimension trajectory $z(t)$.

Initialization of HMM-VAR is either performed by an initial path, from which the HMM parameters are estimated, or directly by HMM-VAR parameters. These parameters are a transition matrix, an initial distribution, and for each hidden state a covariance matrix of noise intensities and a regression matrix.

Here, we start the initialization with an initial path. The initial path, which serves as initialization input for the HMM, is taken from the results of the PCCA clustering. PCCA delivers a macrostate trajectory $z_{\text{macro}}(t)$. This trajectory assigns each time step to one of the C macrostate clusters, detected by PCCA. Since we expect that this macrostate trajectory exhibits a proper assignment to macrostates, it is taken as initial path for HMM initialization.

This expectation is confirmed by the following facts:

- Taking the macrostate trajectory $z_{\text{macro}}(t)$ as initialization of HMM the obtained likelihood after the first calculation step is higher than the likelihood obtained from an equally distributed random sequence. The generated random sequence has for each time step an integer random number $0 \leq r < C$, with C the number of clusters.
- Taking the macrostate trajectory as initialization of HMM the obtained likelihood after the first calculation step is higher than the likelihood obtained from an initialization with a random Markov chain based on a transition matrix T.

After convergence of HMM-VAR the Viterbi path has been calculated. The Viterbi path contains the same information as the macrostate trajectory $z_{\text{macro}}(t)$ does for PCCA. Thus the Viterbi path contains the most likely assignment to hidden states of the HMM-VAR model, which represent metastable states, under the given observation $z(t)$.

In order to obtain appropriate visualization results of the clustering by HMM-VAR we count how often the microstate is assigned to cluster c for each microstate, which is one of the 30×30 microstates along x and y respectively. We denote by $|A|$ the total number of microstates and by $|C|$ the number of clusters. To express the membership a matrix $M^{|A|\times|C|}$ is used. So, we determine the metastable state c for time step t from the Viterbi path, retrieve the according microstate a for t from the microstate trajectory $z_\mu(t)$ and add one for the appropriate entry in the membership matrix $m_{a,c}$. Finally, the entries $m_{a,c}$ are rescaled such that $\sum_i m_{a,i} = 1$ for each a.

A visual representation of the clustering results of HMM-VAR is given in Figure 3.9 and 3.10. The clustering results show a clean distinction of the three metastable states.

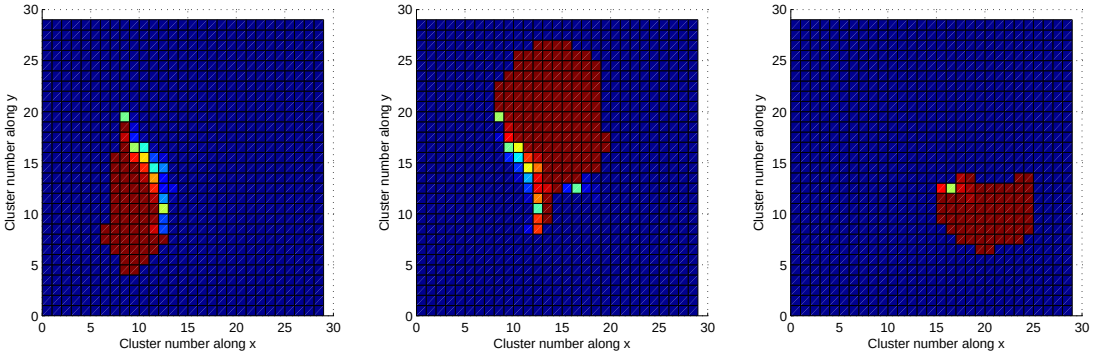

Figure 3.9: HMM-VAR clustering results of model trajectory for projection along x.

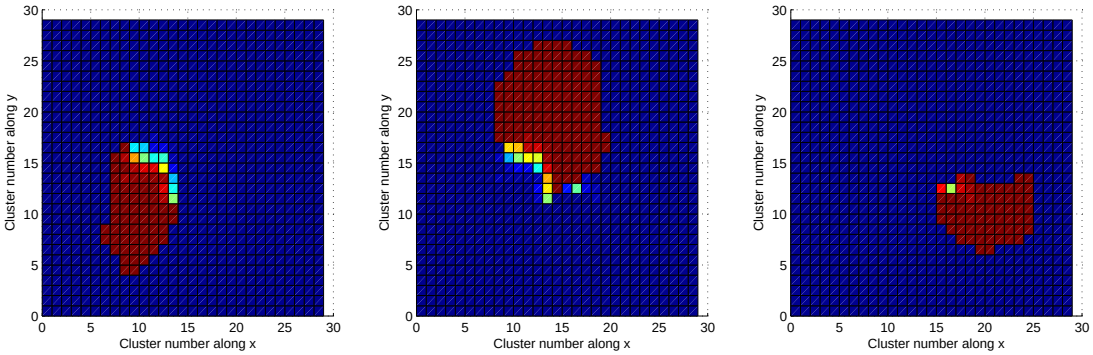

Figure 3.10: HMM-VAR clustering results of model trajectory for projection along y.

The above clustering is executed for lagtimes $\tau \in \{1, 10, 100, 1000, 2000, 5000, 7000, 10000\}$. Here, we reference to the theory of this approach given in Section 2.10.1. With an implementation of that method the clustering could be performed for the above lagtimes. A comparison of the ITS is given in Figure 3.11, which shows proper agreement between the true timescales and those obtained by HMM-VAR.

3.3.5 Convergence of HMM-VAR from Wrong Assignment

In Section 3.3.4 we have initialized HMM-VAR from a path, which stems from PCCA. Here, we perform the initialization by a Markov chain, which is based on a transition matrix T. When the initialization is performed that way, the clustering results are wrong. The Baum-Welch algorithm has converged, but the local minimum, which is found, is not the true global minimum (see Figure 3.12). The wrong clustering results using that kind of initialization are nearly identical for all lagtimes. An illustration of the likelihoods due to different initialization is given in Figure 3.13.

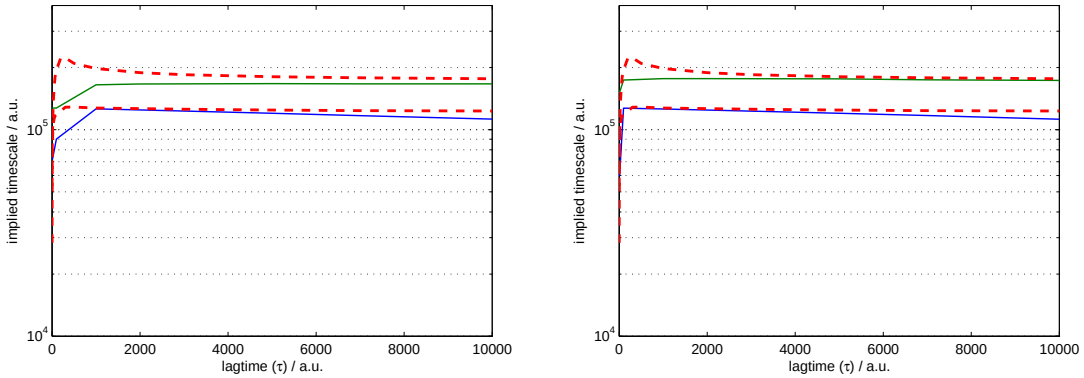

Figure 3.11: Implied timescales obtained by HMM-VAR. Left: ITS obtained for clustering the X-component of trajectory. Right: ITS obtained for clustering the Y-component of trajectory.

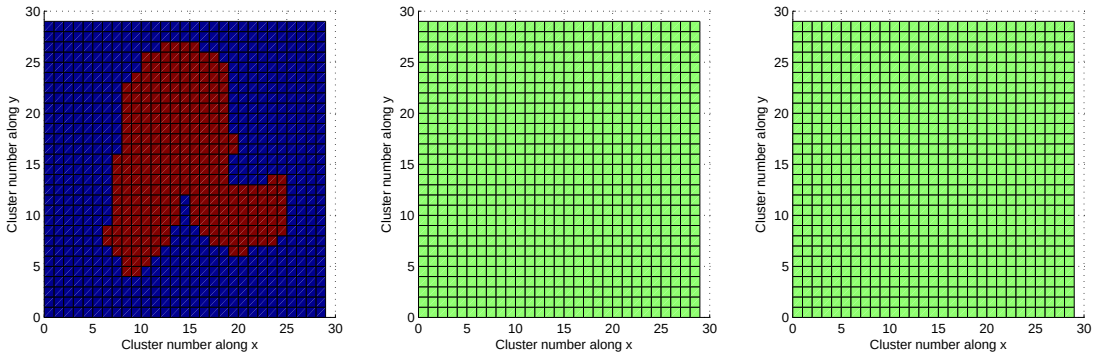

Figure 3.12: HMM-VAR clustering results of model trajectory for initialization from random Markov process. The likelihood has converged to a local minimum, but this minimum does not yield a proper clustering.

	likelihood (initialization)	likelihood (converged)
Initialization from PCCA macrostate trajectory	4317577.21	4317584.91
Initialization by random Markov process	4307543.24	4307543.30

Figure 3.13: Likelihoods obtained by different initializations of HMM-VAR. If the initialization is done by random Markov process the likelihood converges to a local minimum with lower likelihood. Furthermore, the initialization by the PCCA macrostate trajectory is nearly optimal as can be seen in the first row.

3.4 3-Well Potential where HMM-VAR Clustering fails

In this section the analysis is identical to that in the previous section. But instead we investigate here a model trajectory which is based on the potential $U_{\mathrm{std}}(t)$ (see Figure 3.2). Clustering is done by PCCA and HMM-VAR as beforehand.

The potential $U_{\mathrm{std}}(t)$ is chosen such, that the two bottom-most basins have the same y-coordinate of the potential basin and additionally have an identical shape. Here, identical shape means, that the slopes of the potentials are identical. When doing PCCA, clustering results are expected to deliver reasonable results. The transition matrix $T(\tau)$ which is obtained by counting microstate transitions will hardly be influenced by different potential shapes, so we expect PCCA to cluster reliably. But when executing HMM-VAR on the projection onto the y-coordinate, we expect HMM-VAR to miserably fail. This is due to the fact that the kinetics of the two bottom-most potentials do not differ. In contrast, the clustering of the trajectory projected onto the x-axis is expected to deliver reasonable results, since the mean values of the potential minima differ significantly. The geometrical distinction of potential basins allows HMM-VAR to separate the according states.

Further analysis will be performed to study the effect on the obtained ITS. A plot of the true timescales is given in Figure 3.17. A comparison with the ITS obtained by PCCA and HMM-VAR reveals the ability of the clustering algorithms to properly identify the time scales of switching processes between metastable states. As well as we expect the visual clustering results of HMM-VAR not to be reasonable for the y-coordinate of the trajectory, we also expect the comparison of true timescales with timescales obtained by HMM-VAR to detect a discrepancy in the ITS.

3.4.1 PCCA

The clustering results obtained by PCCA are given in Figures 3.15 and 3.16. The three different clusters according to the potential U_{std} are cleanly resolved. With increased lagtime τ a slight fuzziness in clustering can be observed. Especially this seems to be apparent on the "rims" of the clustering regions.

3.4.2 HMM-VAR

Clustering of the model trajectory based on the potential U_{std} by HMM-VAR is performed in the same way as in Section 3.3.4. Therefore and from the results of the previous section, the initialization of HMM-VAR is done by the macrostate trajectory obtained by PCCA. The lagtimes τ are chosen identically.

The clustering results of HMM-VAR for this model trajectory are displayed in Figure 3.19 and 3.20. The clustering results for the y-projection reveal the incorrect clustering. The two bottom most metastable can not be distinguished by their kinetics and are mangled. A flat third state is detected, which lies geometrically in between the two bottom most states and the top most state.

The influence of the wrong clustering has serious impact on the ITS as well (see Figure 3.21). Especially the implied timescales for the y-component clustering show this behavior.

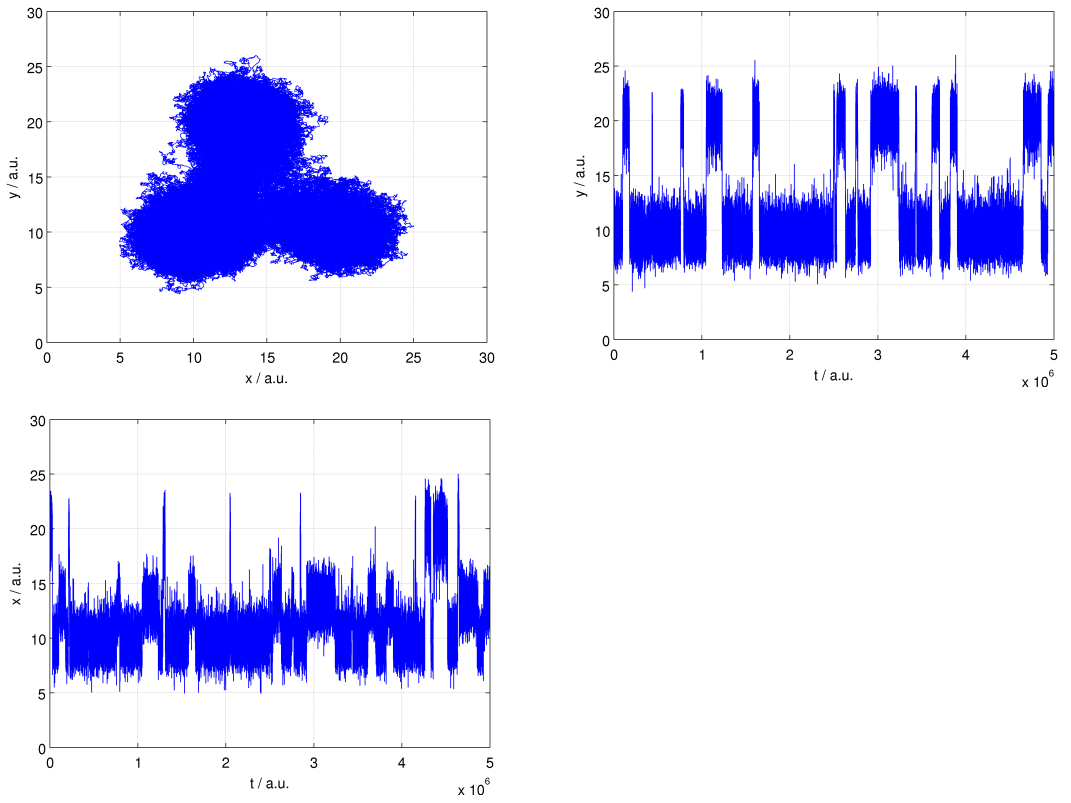

Figure 3.14: A plot of the trajectory $z(t)$ for the potential $U_{\mathrm{std}}(z)$ (see Figure 3.2). Top-left: Visited points of $z(t)$ for $t = 0$ to $t = 5 \cdot 10^6$. Top-right: Projection of the trajectory $z(t)$ onto the y-axis. Bottom: Projection of the trajectory $z(t)$ onto the x-axis.

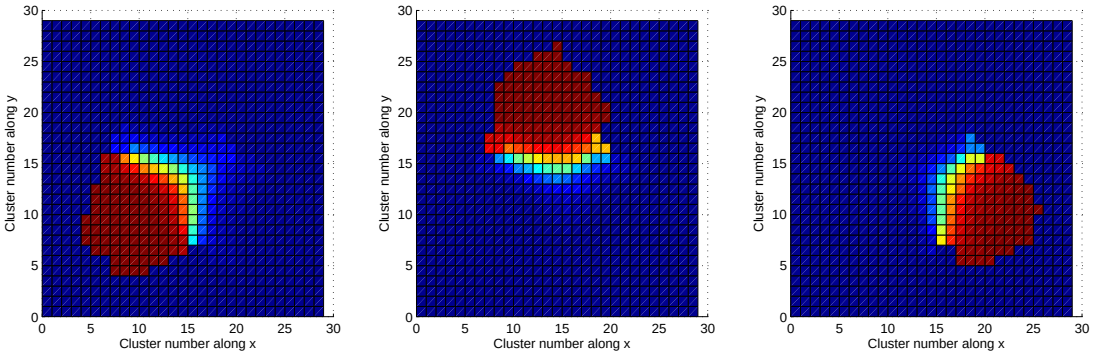

Figure 3.15: PCCA clustering results of model trajectory for lagtime 1.

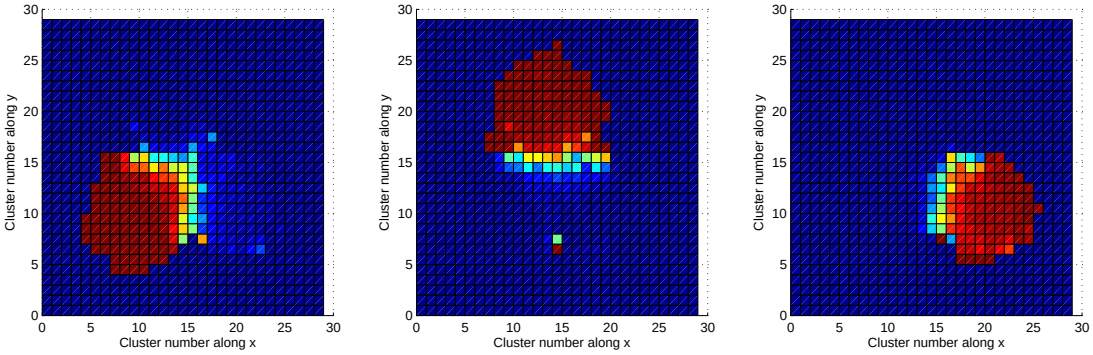

Figure 3.16: PCCA clustering results of model trajectory for lagtime 1000.

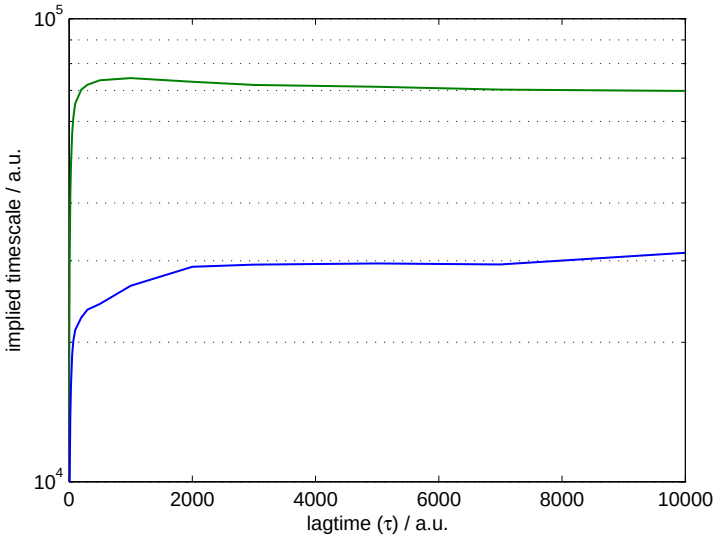

Figure 3.17: True timescales determined by the eigenvalues of the transition matrix $T(\tau)$. The green curve displays the ITS calculated by the 2^{nd} eigenvalue, the blue curve display the ITS calculated by the 3^{rd} eigenvalue.

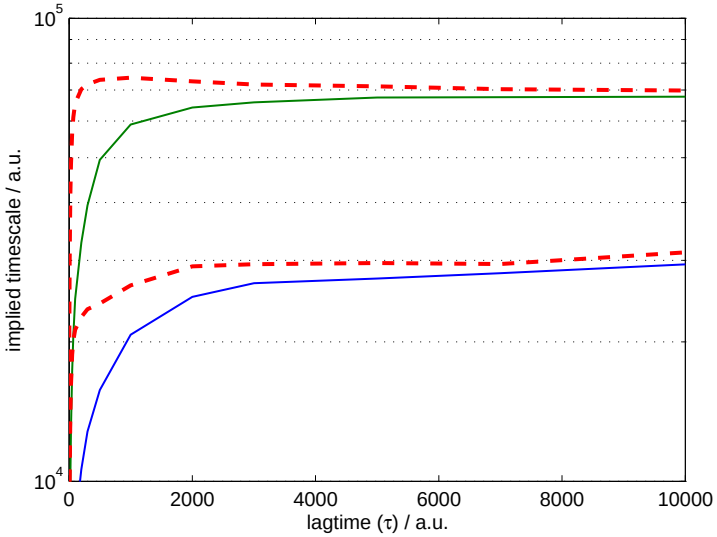

Figure 3.18: Implied timescales for PCCA clustering of the 2nd and the 3rd eigenvalue. The red dashed curves indicate the true timescales.

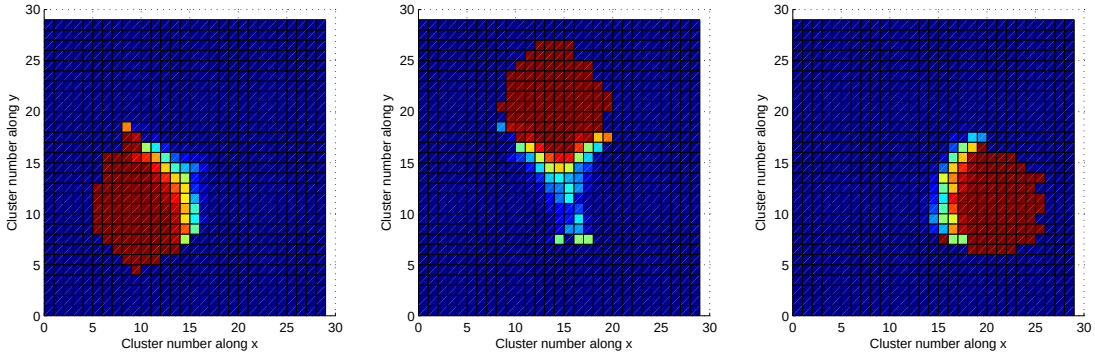

Figure 3.19: HMM-VAR clustering results of model trajectory for projection along x.

The ITS related to the third eigenvalue are two decades out of range in comparison to the true timescales. This is due to the mangling of the two bottom most states.

3.5 Comprehension of Results

By the analysis of simple model trajectories with HMM-VAR and PCCA and the evaluation of the clustering results and ITS a first impression of the performance is obtained and several insights into possible problems are provided. From the clustering results based on HMM-VAR the following conclusion of the essential results is drawn:

- HMM-VAR is sensitive to the type of the potential. Overlapping potentials (as in the case of U_{type}) need to differ substantially in order to allow a proper distinction of metastable states.

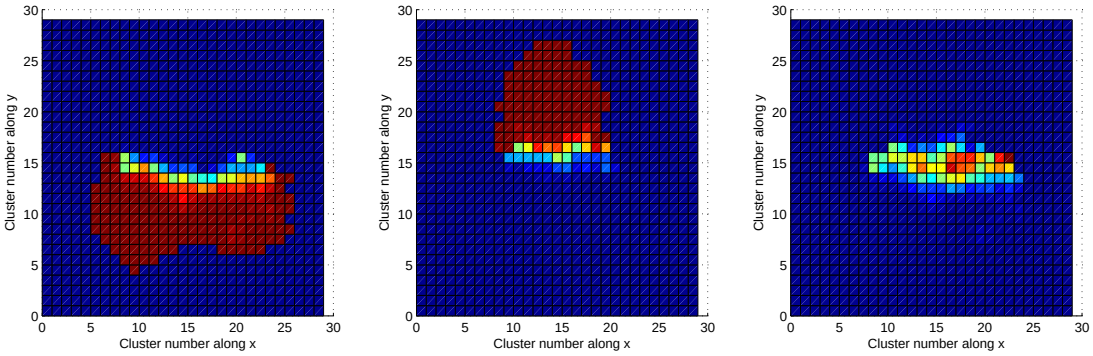

Figure 3.20: HMM-VAR clustering results of model trajectory for projection along y. The clustering fails and does not determine the true clusters. On the left the two potentials have been mangled.

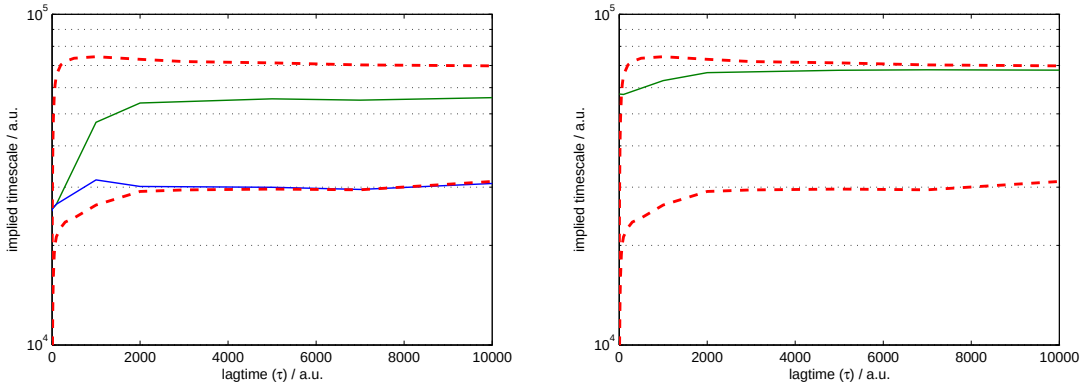

Figure 3.21: Implied timescales obtained by HMM-VAR. Left: X-component of trajectory. Right: Y-component of trajectory. The ITS obtained by the 3$^{\text{rd}}$ eigenvalue for different lagtimes τ are two decades below the expected results (bottom dotted red line). The improper ITS for that process is a serious indication that the clustering has failed.

- The determination of the model parameters via the Baum-Welch algorithm, concrete HMM-VAR, is susceptible to the initial conditions. Even though the Baum-Welch algorithm always converges, the obtained minimum can be local. Thus to overcome the problem of local convergence a genetic algorithm is attached to the Baum-Welch algorithm.

4 A Genetic Algorithm to escape Local Likelihood Maximizers in HMMs

As presented in Section 3.5 the convergence of the Baum-Welch algorithm towards local minima is a serious drawback. In order to overcome this problem a new approach is required. One possible approach is based on the combination of a genetic algorithm (GA) and the Baum-Welch algorithm. This approach will be investigated here.

First, a brief introduction to genetic algorithms is delivered. The introduction contains the definition of terms, the basic idea of a genetic algorithm, a description of genetic operations and a discussion of possible termination criteria of a genetic algorithm.

Afterwards, the formulation of HMM-VAR problems within the context of GA is provided.

The application of the GA to the model trajectories of the last chapter is performed and thus a possibility to escape local minima is presented.

Implementation details of the GA are illustrated in the last section of this chapter.

4.1 Genetic Algorithm

Genetic algorithms belong to the category of global search heuristics. Mathematically speaking, genetic algorithms provide a possible solution to determine the global optimum of the optimization problem $f : S \to \mathbb{R}$, where S denotes some search space.

Inspired by evolutionary biology and by the use biological principles such as inheritance, mutation, crossover and selection, a variety of solution candidates within the search space are created by the genetic algorithm. Thus solution candidates with a high quality or goodness of the solution are obtained[5].

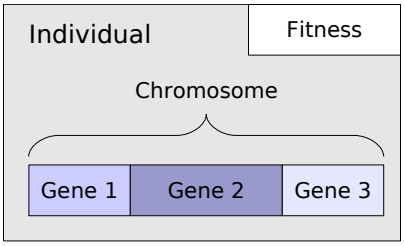

Figure 4.1: Diagram of an individual, which is the basic entity of a genetic algorithm. The individual consists of a chromosome, which itself contains genes. Each individual has a fitness, describing the ability of the individual to survive.

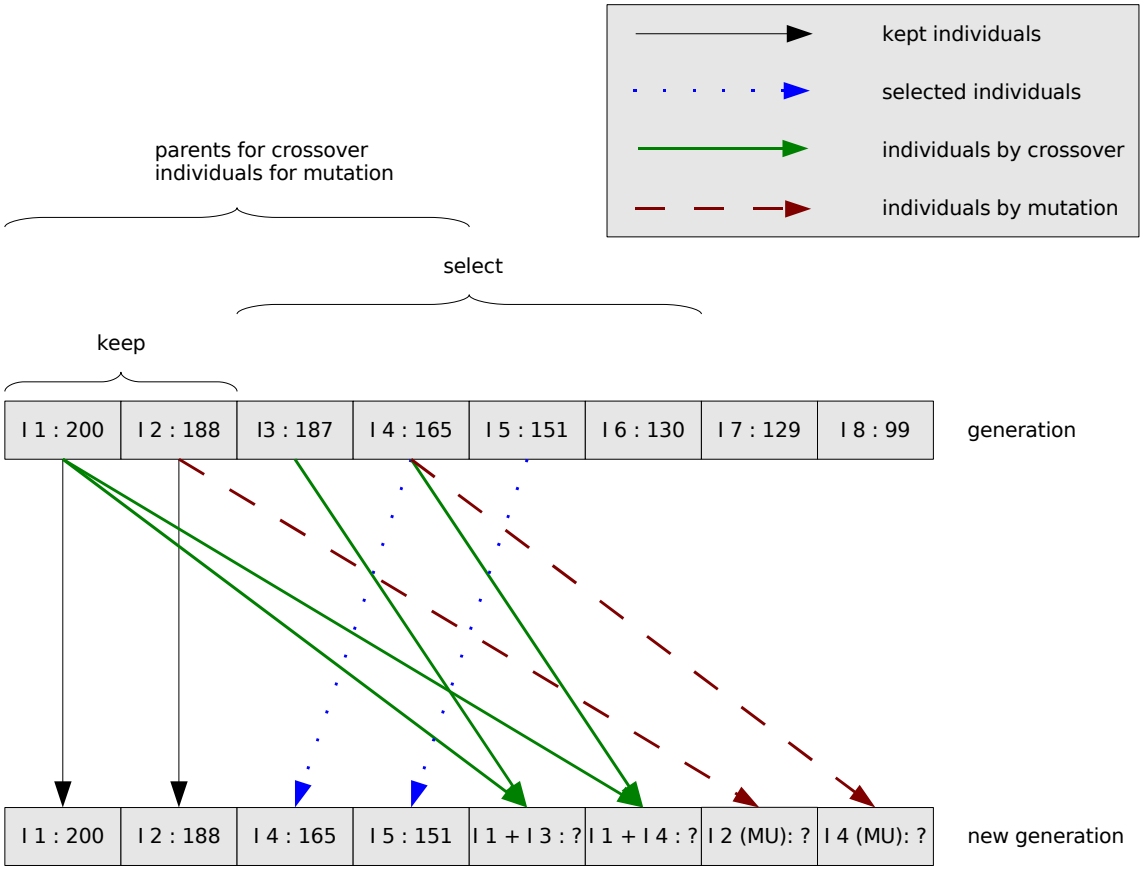

Figure 4.2: Illustration of the creation process of a new generation based on the genetic operations selection, crossover and mutation within a genetic algorithm.

Within the context of genetic algorithms a solution candidate is called individual (see Figure 4.1). Several individuals form a generation. Each individual has a chromosome, which consists of a set of genes. The chromosome is decoded such that it represents one element within the search space. Staying in the context of GA the optimization problem f delivers a measure for the quality of an individual. This function f is called the fitness of an individual, the higher the fitness, the greater the chance of an individual to survive.

Implementation centric, genetic algorithms simulate the evolution of generations of individuals. The evolution usually starts from a population of randomly generated individuals and takes places in newly created generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm.

4.1.1 Basic Algorithm

Basically, a genetic algorithm is based on the following steps

1. Generate / choose initial generation consisting of n individuals.
2. Repeat
   a) Calculate the fitness of each individual of the current generation
   b) Create new generation (see Figure 4.2) by application of genetic operations selection, crossover and mutation.
   c) Make the new generation current.
3. Until termination criterion is satisfied.

The initial generation of individuals is direct input of the genetic algorithm. An alternative is to input just one individual and to create the remaining individuals of the initial generation by mutation of that individual.

4.1.2 Genetic Operations

Genetic operation are the foundation of a genetic algorithm. Like in evolutionary biology offsprings or new individuals are replicated by combination of parental genes, the so called crossover and suffer from selection and mutation through the environment. A genetic algorithm now simulates the evolution of individuals based on these three genetic operations.

In order to mimic the biology **selection** refers to the survival of the fittest. With each individual having its fitness function, which reflects the survivability of this individual, the selection is based on exactly that fitness function. Thus a higher fitness function yields a higher chance of an individual to survive. Algorithmically, selection can be performed by several approaches. The selection of one individual is usually performed by:

- Random selection from the first n fittest individuals of one generation.
- Roulette-wheel selection based on a selection probability $p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$ of an individual i, which depends on the individuals fitness f_i.
- Tournament based selection
  - Set tournament size k, selection probability p and select k individuals.
  - Sort k individuals according to their fitness.
  - Choose individual i with probability $(1-p)^{i-1} \cdot p$. If chosen, return that individual, otherwise continue with next individual $i+1$. The returned individual represents the "winner" of the tournament.

The **crossover** operation generates offsprings by combination of parental genes. Algorithmically, individuals of the next generation are created by combining genes of parental individuals of the current generation (see Figure 4.2, green arrows). For the decision which genes to take from the parental generation, two strategies are normally used:

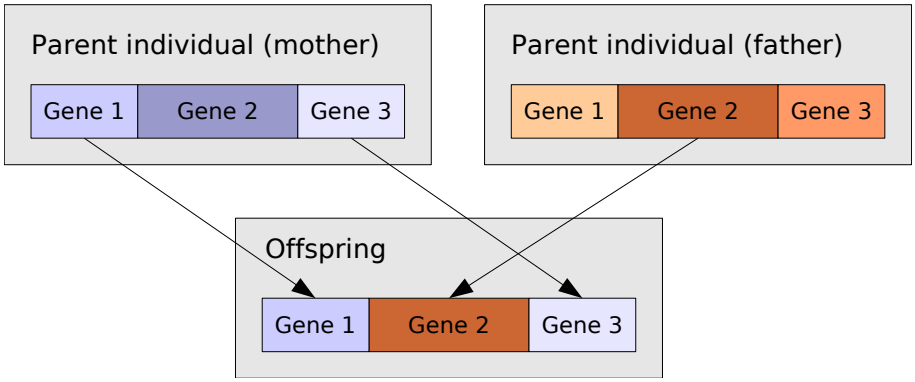

Figure 4.3: Crossover of individuals. Two genes of the offspring are taken from the mother, while one gene is taken from the father.

- The genes of the offspring are taken alternately from the mother and father, thus the first gene stems from the mother, second from the father, third from the mother and so on.
- For each gene it is randomly chosen, whether it is taken from the mother or from the father.

The choice, which parental individuals to use for offspring generation, is usually performed by a tournament based selection.

Mutation is the third of the genetic operations (see Figure 4.4). Like in a real environment individuals suffer from mutation. Mutation in the context of genetic algorithms means the altering of the chromosome by mutation of one or several genes.

Within the creation process of a new generation some individuals are mutated in some or all of their genes. These mutated individuals are added to the new generation, this process is illustrated in Figure 4.2, see brown dashed arrows.

The mutation operation can be performed by two approaches.

- If the data of the gene is coded as a string of bits, mutation is done by randomly flipping a certain amount of bits, depending on the amount of mutation.
- If the data of the gene can be interpreted as a number, mutation is done by adding some random value. The strength of mutation depends on the range, the random value is picked from.

When performing mutation operations, care has to be taken not to create invalid genes. Here, invalid gene means either that the search space of the problem has been left, since the whole chromosome has become invalid. Or it means, that the gene is no longer a valid representation of the problem.

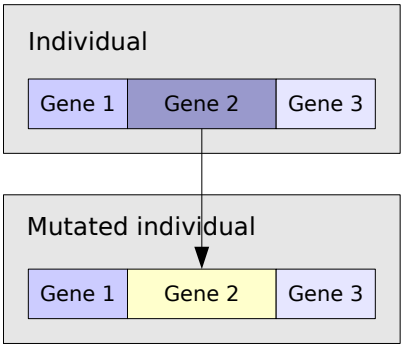

Figure 4.4: Mutation of an individual.

4.1.3 Termination Criteria

Basically, step 2 of the genetic algorithm (see 4.1.1) can be repeated as long as necessary. With each new generation individuals can be created, which provide better solutions according to their fitness. Unfortunately, there exists no guarantee, that this is going to happen. Instead it is also possible, that either no individuals are created which provide a better fitness, or that a certain set of chromosomes has established, which does not deliver a better solution. It is even imaginable, that by mutation and crossover only individuals are created which are worse then individuals of previous generations. Nevertheless, a reasonable termination criterion has to be set.

Several different termination criteria of a genetic algorithm are possible. Here, we just list some. Depending on the type of the problem a reasonable termination criterion has to be chosen.

- The GA can be terminated after a certain number of generations has been reached. This is by far the most simple abort criterion.
- Furthermore, the genetic algorithm can be terminated after a certain fitness threshold has been reached. This criterion makes sense, when a concrete fitness is sufficient for the problem solution.
- A further abort criterion can be triggered, if no improvement has been achieved during the last m generations. Explicitly, this is calculated as

$$\frac{1}{m} \sum_{i=n-m}^{n} f_{\text{best}}(i) = f_{\text{best}}(n),$$

  if n is the current generation and $f_{\text{best}}(i)$ is the best fitness of all individuals in generation i.
- A further abort criterion can be defined as follows: the average improvement of the current generation compared to the previous generation is below a certain threshold.

If the termination criterion is fulfilled, the genetic algorithm returns the individual with the best fitness of the current generation.

4.2 Example of Genetic Algorithm with HMM-VAR

According to the proposal given in Section 3.5, a genetic algorithm will be applied to HMM-VAR. This approach delivers a possibility to overcome the problem of getting stuck in local minima during convergence of the EM algorithm.

4.2.1 Prerequisites

Before a genetic algorithm can be applied to HMM-VAR it has to be clarified

- which HMM-VAR parameters form the chromosome of an individual,
- how the coding of chromosomes and genes is chosen to represent HMM-VAR parameters and
- how the fitness of an individual corresponds to the likelihood obtained by HMM-VAR.

The last aspect is answered directly. Each individual has a fitness, which is the higher, the "better" the individual is. The chromosome of an individual represents a point in the search space and serves as initialization for HMM-VAR. Here, we have not said how the chromosome represents the initialization input of HMM-VAR. But the maximum-likelihood of HMM-VAR, or to be more precise of the EM algorithm, can be directly used as the fitness of an individual.

HMM-VAR can be either initialized by an initial path or by model parameters. The initial path has the same length as the length of the input trajectory. If initialization is done by an initial path, a chromosome represents such a path and each individual has such a chromosome. First, mutation and crossover operations on paths are tedious and need to be carefully designed. Second, a path consisting of T time steps assigning to one of C states has a variety of C^T different representations. This requires even more elaborated genetic operations. Additionally, the memory consumption is high, since each individual has to keep one path, which requires a clever encoding. Nevertheless, the use of an initial path as initialization can have the benefit, that HMM-VAR is more unlikely to run into regions of bad model parameters. Bad means here, that these parameters cause numerical problems due to bad conditioning.

The model parameters of HMM-VAR are a transition matrix T, an initial distribution π and covariance matrix Σ_i and a regression matrix Φ_i, the last two each per hidden state. If the initialization is performed by model parameters, these parameters have to be encoded in the chromosome of the individual. Here, we choose an obvious approach, which exposes a relatively high granularity of genes. Each of the parameters like transition matrix, initial distribution, etc. is put into a single gene. High granularity here means, that one single gene is chosen to represent a matrix. A low granularity would mean, that each single entry of a matrix is represented as gene. Due to several constraints which are present for the HMM-VAR parameters the approach of high granularity is taken.

This choice of genes results in a total of $2 + 2n$ genes, if n is the number of hidden states for the hidden Markov model. As illustration of the total number of single parameters

for an one-dimensional trajectory and using HMM-VAR with a memory of $p = 1$ (which is equivalent to HMM-SDE) one obtains: transition matrix has $n \cdot n$ parameters, the initial distribution has n parameters and per hidden state we have 2 parameters for the regression matrix and 1 parameter for the noise intensity. Adding up we obtain a total of $n \cdot n + n + 3n = n^2 + 4n$ parameters. For 3 hidden states, that results in 21 parameters.

4.2.2 Mutation of Genes representing HMM-VAR Parameters

Special care has to be taken during the application of genetic operations on genes, which represent HMM-VAR parameters. The crossover operation is not critical here due to the high granularity of genes. By the crossover operation only genes as a whole are combined to setup the chromosome of an offspring. So, the genes are not modified internally.

But the mutation operation is critical. The mutation operation has to applied such, that the mutated gene is a valid gene.

Transition matrices are stochastic matrices, so each entry t_{ij} of such a matrix T has to be $0 \leq t_{ij} \leq 1$, and the row sum $\sum_i t_{ij}$ equals 1. These constraints have to be taken into account, when applying the mutation operation. Exactly, we have a mutation by a random number r within the range s, so $-s \leq r \leq s$. After the mutation of an entry t_{ij} the mutated entry is $t_{ij} + r$. The first constraint is $0 \leq t_{ij} + r \leq 1$, while the row sum requires to modify a different entry $t_{i'j}$ within the same row of the matrix, whose value is set to $t_{i'j} - r$, such that the row sum equals 1. Algorithmically, first an entry of the matrix is randomly selected. Then, a different entry within the same row is randomly selected. After generation of a random number r it is checked if all constraints can be fulfilled. If so, the mutation is applied. Otherwise, a new random r is generated. This is repeated, until an appropriate random r is found.

When mutating the noise intensity covariance matrix and the regression matrix different constraints have to be considered, f.e. the noise intensity matrix has to be positive definite, that is $z^T \Sigma z > 0$ for all $z \in \mathbb{R}^n$ or equivalently all eigenvectors or Σ have to be real and positive.

We have performed the genetic algorithm with HMM-VAR and a memory of $p = 1$ on a one-dimensional trajectory. Therefore, the noise intensity matrix Σ_i reduces to a scalar and the regression matrix Φ is in $\mathbb{R}^{2\times 1}$. This is a serious simplification.

If the trajectory has higher dimension a possible violation of these constraints by mutation has to be avoided. More importantly, an elaboration of a valid parameter space for the regression matrix has to be performed.

4.2.3 Simple Test Cases

A simple test of the GA has been performed with the Rosenbrock function

$$f_{\text{rosen}}(x, y) = (1 - x)^2 + 105 \cdot (y - x^2)^2,$$

which has a global minimum at $(1, 1)$. The chromosome of the individuals here is chosen to consist of two genes, one gene for x and one gene for y. Mutation of genes is simply

achieved by addition of a random number r. Crossover is performed as described in Section 4.1.2. A generation size of 100 individuals is chosen, termination criterion is "no change of best individual" within the last 10 generations. The starting point for the genetic algorithm is set to $(-5, 5)$. With that setup of the GA, the global minimum at $(1, 1)$ is found after 23 generations.

In order to get an impression of the dependence of the parameter/gene number on genetic algorithms another basic test function $f_{\text{harm}} : \mathbb{R}^{21} \to \mathbb{R}$ is used:

$$f_{\text{harm}}(\vec{x}) = \vec{x} \cdot \vec{x},$$

which has a global minimum at $(0, \ldots, 0)$. Here, for each component of the vector $\vec{x}$ a single gene is setup. The setup of mutation and crossover is like in the test case above. Termination criterion is "no change of best individual" within the last 100 generations. The generation size is set to 1000 individuals. After approximately 250 generations the minimum is found.

For both examples the fitness function of the individuals was chosen to be $f_{\text{fitness}} = -f_{\text{rosen}}$ and accordingly $f_{\text{fitness}} = -f_{\text{harm}}$.

4.2.4 Application to HMM-VAR on Model Trajectory

In Section 3.3.5 it has been shown that HMM-VAR does not converge to a global optimum. Instead it converges to a local optimum, which does not deliver reasonable clustering results. We focus our analysis of the genetic algorithm to that case.

The x-component of the trajectory z_{type} is the observation sequence for the HMM-VAR. For the application to this model trajectory, an initial individual and thus a set of HMM-VAR parameters needs to be provided. A pure guess of the model parameters seems impossible. Not only does the estimation of the parameters require some previous knowledge, but also is HMM-VAR sensitive to parameters which are too far off from sensible value. Instead a different approach is taken: the parameters of the initial individual are obtained by a single initial HMM-VAR run. This HMM-VAR run is initialized by a random Markov chain based on a transition matrix T. The transition matrix has 0.98 on its main diagonal, all off-diagonal value are 0.01. As a remark here: the estimated transition matrix obtained by HMM-VAR for proper clustering (see Section 3.3.4) has 0.9991 on its main diagonal for lagtime 1. The estimated parameters after the convergence of HMM-VAR run are taken as parameters for the initial individual. The remaining individuals of the first generation are obtained by mutation of that initial individual.

In order to calculate the fitness of an individual the likelihood of HMM-VAR is used. Three different strategies for the application of an genetic algorithm on top of HMM-VAR are possible:

1. The fitness calculation is directly based on the likelihood for given model parameters λ. This strategy completely leaves out the Baum-Welch algorithm. Given the parameters λ from the chromosome of an individual the likelihood $L(O|\lambda)$ can be computed and is taken as fitness.

2. The fitness calculation is based on the Baum-Welch algorithm. Instead of the sole computation of the likelihood like before, here, the Baum-Welch algorithm is used. The Baum-Welch algorithm is an instance of the Expectation-Maximization algorithm. Starting with the given model parameters by the individual, the maximum likelihood obtained by the EM algorithm is taken as fitness.
3. The third strategy is identical to the second one expect for the fact, that here the estimated model parameters obtained by the EM algorithm are written back to the genes of the individual.

4.2.5 Results and Performance

With the above retrieval of HMM-VAR parameters, through an initial run of HMM-VAR, an individual is setup. First generations individuals are created through mutation of that individual.

The mutation amount is selected empirically. Initial tests have shown, that a high mutation amount easily leads to parameter regimes, where numerical problems occur in HMM-VAR calculations or where parameters are far off from the optimal parameters. A low mutation amount on the other hand has the consequence not to sample the search space exhaustively enough. This results in final solutions of the genetic algorithm which are not optimal.

In general it is difficult to set the parameters which influence the genetic algorithm. These parameters are individuals per generation and parameters for genetic operations, like mutation amount, crossbreed ratio, selection ratio etc. These values for genetic operation are mainly untouched, we have used parameters for the genetic algorithm which have worked for the test cases above. We have focused the variation to mutation amount and number of individuals per generation and have used parameters which seem reasonable after initial tests.

The application of the genetic algorithm on top of HMM-VAR is performed for two of the above strategies.

The sole computation by strategy 1 is detected to need too much computation time. Compared to the test case for the function f_{harm}, a size of 1000 individuals per generation seems necessary. Analogously a total number of 250 generations is estimated to get to the optimum solution. A simple likelihood calculation takes around 10 seconds. Thus $2.5 \cdot 10^7$ secs $\sim$ 29 days are required. We have performed the analysis with a reduced generation size, but have not reached to optimal solution thereby.

Second, we have analyzed strategy 2. The genetic algorithm is adjusted to have 20 individuals per generation. Termination criterion is "a non-changing best individual" for 10 generations. With that setup we have found the optimal solution after a computation time of $\sim$ 3 days. The results of the convergence towards the optimal likelihood are plotted in Figure 4.5.

4.3 Comprehension

Genetic algorithms provide one possibility to locate a global optimum within some landscape. The successful application of the genetic algorithm on top of HMM-VAR has been

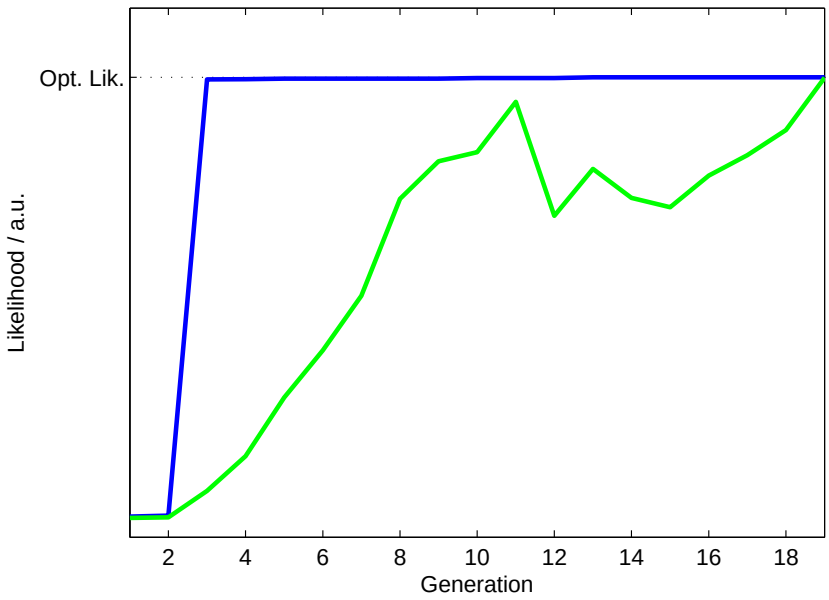

Figure 4.5: Optimization results of the genetic algorithm combined with the Baum-Welch algorithm on HMM-VAR. The blue curve displays the fitness of the best individual per generation. The green curve displays the generations fitness mean. After 20 generations the x-axis is cut, since there is no improvement in the fitness of the best individual.

shown to obtain an optimal clustering, where the sole Baum-Welch has failed and got stuck in a local maximum. The obtained clustering result is here optimal in that sense, that no better likelihood has been found.

By far, this does not prove the general reliability and performance of genetic algorithms, but the results have provided a first step into the successful combination of genetic algorithms and HMM-VAR.

Even though genetic algorithms seem to include a high power in the determination of global optima, they suffer from certain facts. First, no guarantee exists to really determine the global optimum. Absolute guarantee of a global minimum exists only, if the whole parameter space has been sampled. But even though genetic algorithms are just heuristics, they provide a powerful way to explore the parameter space and restrict to essential regions in that parameter space.

Especially when applied to HMM-VAR additional problems have to be taken into account:

- The parameter space of HMM-VAR parameters is large,
- numerical instabilities and problems due to improper parameters can occur during HMM-VAR calculations,
- mutation of HMM-VAR parameters is difficult,
- the likelihood landscape is far from being “nice”.

4.4 Implementation Details

The implementation of the genetic algorithm is strictly object-oriented (OO). Therefore, the implementation of the genetic algorithm can be easily applied to various problems without the requirement for major program code modifications. With the OO approach, the exchange of individuals, which are adapted to specific problems, can be performed by the implementation of subclasses of the class *Individual*. The appropriate set of genes, accessible by the chromosome, can as well be exchanged by subclasses of the superclass gene and thereby allows easy adaptation to new problem instances.

The basis is formed by the class *GeneticAlgorithm*. The relationship between the classes is given as a class diagram in Figure 4.6. The subclassing of the class *Individual* is shown in Figure 4.7.

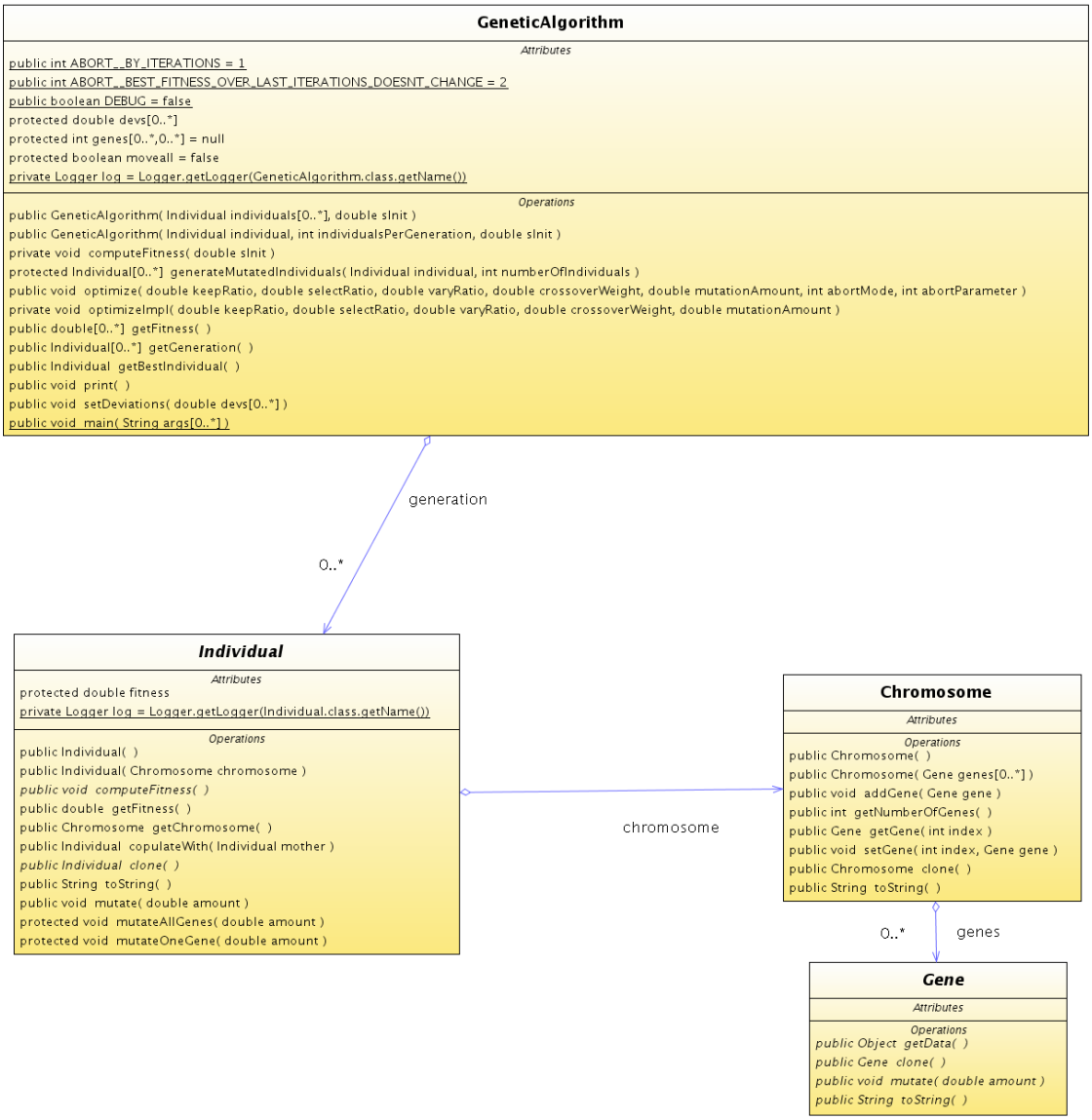

Figure 4.6: Class diagram representing the classes required for the implementation of the genetic algorithm. The basis is build by the class *GeneticAlgorithm*. In its member variable *generation* the set of *Individuals* is held, which builds the current generation. Each *Individual* has a *Chromosome*, the calculation of the fitness function is directly performed in subclasses of the *Individual*. A *Chromosome* holds the different *Genes* it its member variable *genes*.

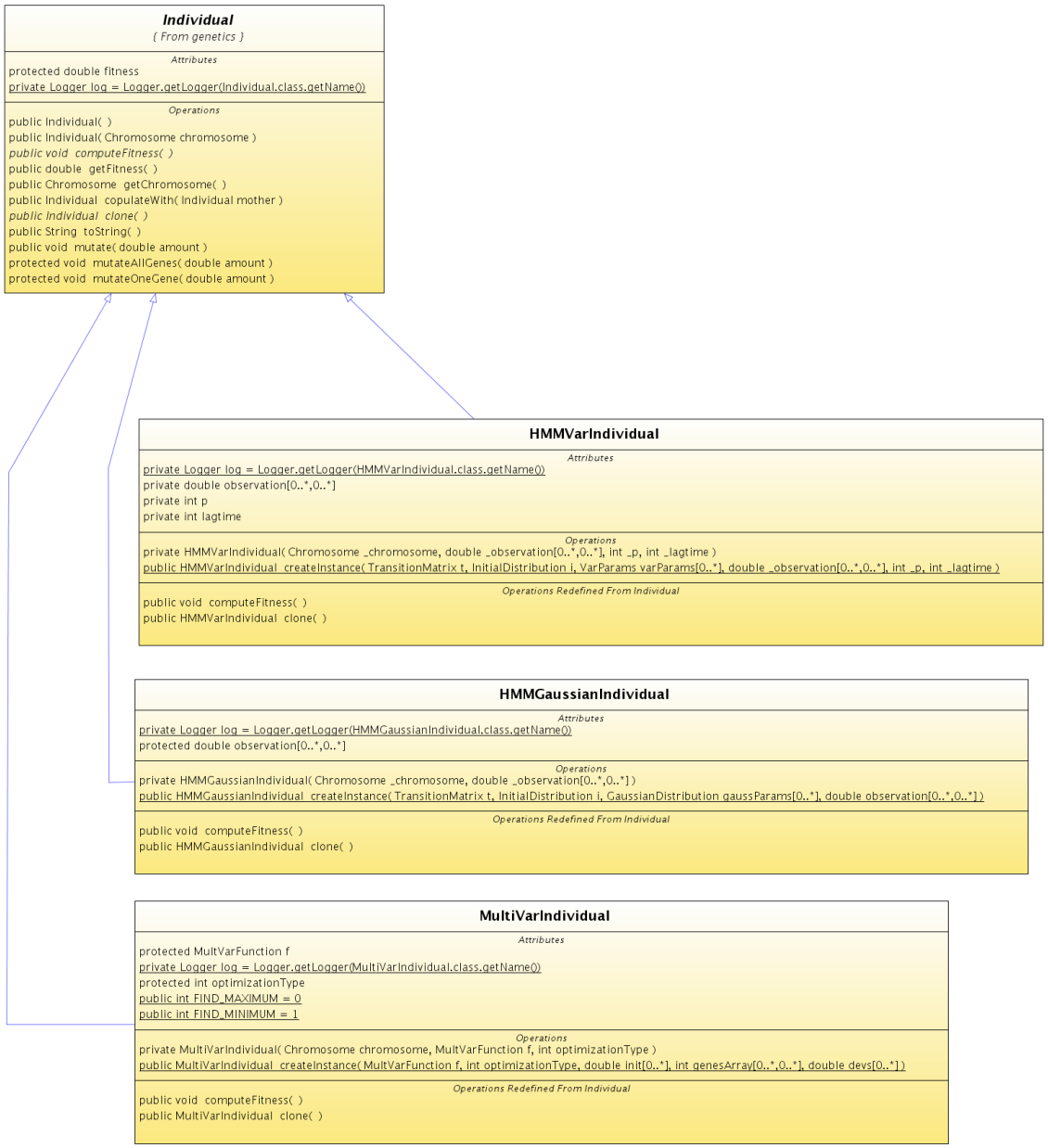

Figure 4.7: Subclasses of *Individual*. Shown are the subclasses *HMMVarIndividual* for the application to HMM-VAR problems, *HMMGaussianIndividual* for the application to HMM-Gaussian and *MultiVarIndividual* for the application to arbitrary functions $f : \mathbb{R}^X \to \mathbb{R}$. *MultiVarIndividual* was used for the application of the genetic algorithm to the test cases f_{harm} and f_{rosen}. Each of the classes uses a problem specific, related set of genes. These genes are subclasses of the class *Gene*.

5 PCCA and HMM on Molecular Trajectories

5.1 MR121-GSGSW Peptide

The MR121-GSGSW peptide is a small glycine-serine repeat modified with the oxazine derivative MR121 (a fluorescent dye) and tryptophan residue (a specific quencher) at the terminal ends -MR121-GSGSW (see Figure 5.1) [15]. In proteins, the glycine-serine motif is frequently found in hairpins and loops, and hence glycine-serine-based peptides are commonly used as model systems for studying end-to-end contact formation in unstructured polypeptide chains [10].

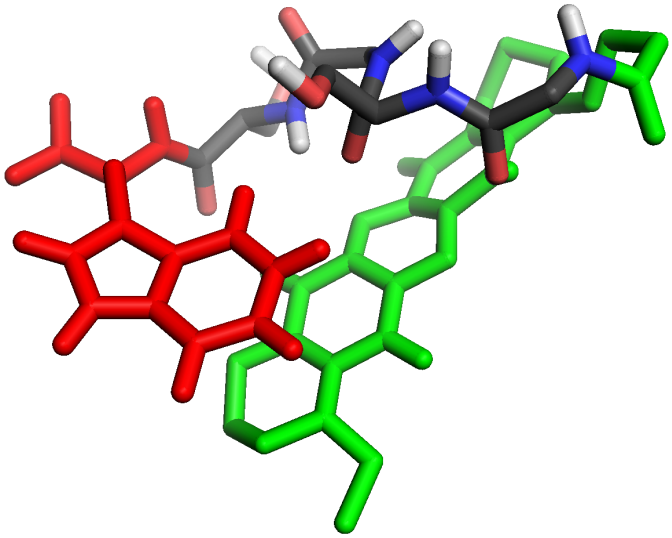

Figure 5.1: Structure of the MR121-GSGSW peptide. The dye MR121 is in green, the backbone of glycine-serine is colored per element (gray for C, blue for N and red for O) and the tryptophan is colored red.

5.2 Approach

Using GROMACS [19] for molecular dynamics simulation a trajectory of the MR121-GSGSW has been calculated. The molecular dynamics simulations are itself based on

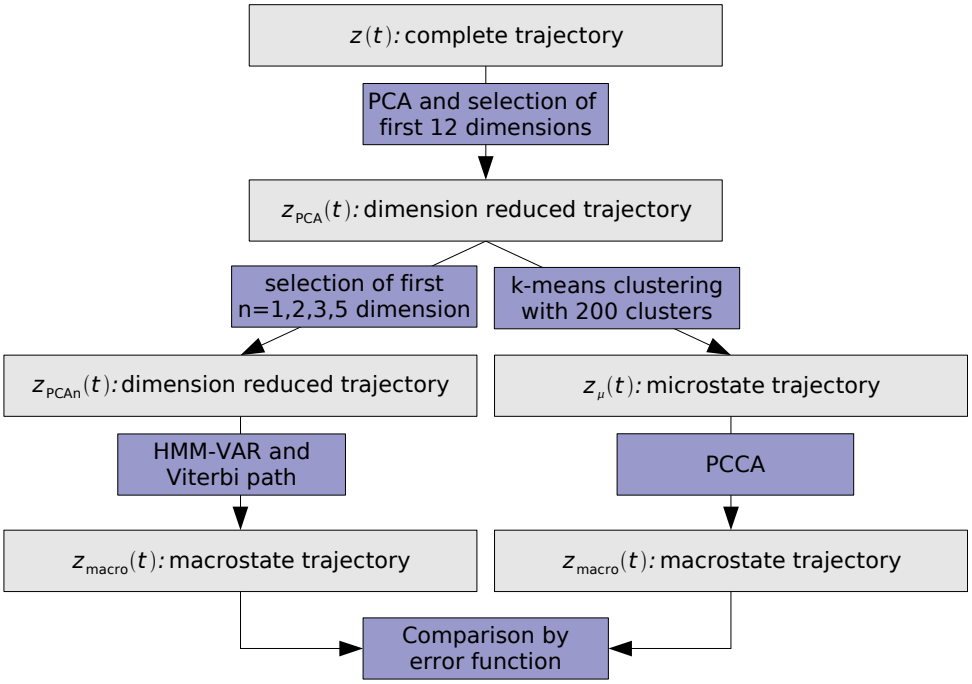

Figure 5.2: Approach for the evaluation and comparison of clustering results for the MR121-GSGSW system.

modeling and presentation by classical mechanics and force fields. The simulation was performed with the GROMOS96 force field [20]. The trajectory contains $T = 5 \cdot 10^6$ time steps with each time step of $2 \cdot 10^{-13}s = 200fs$ length, thus the whole trajectory simulation covers the time of $10^{-6}s = 1\mu s$. The MD simulation is performed with explicit water solvent.

Regarding the trajectory, the system consists of 81 atoms. With three positional degrees of freedom there are $N = 243$ dimensions. So the whole trajectory $z(t)$ is a function $z(t) : \mathbb{R} \rightarrow \mathbb{R}^{243}$.

5.3 Clustering with PCCA

The approach for the execution of the clustering via PCCA is given in the right branch of Figure 5.2. First, a dimension reduction is done by the principal component analysis (PCA).

Using PCA the trajectory is reduced to the 12 first principal components, thus $z_{\mathrm{PCA}}(t) : \mathbb{R} \rightarrow \mathbb{R}^{12}$. The choice of the first 12 dimensions of PCA for further analysis is by far not random. PCA exhibits within these number of components a significant variance in the molecular trajectory, while PCA components above 12 seem not to deliver any significant contribution in terms of variance.

The k-means algorithm was applied to the dimension-reduced trajectory z_{PCA} in order to obtain microstates. Here, the k-means algorithm was performed with a total number of

$C = 200$ clusters. Interpreting the microstate trajectory as a function it maps every time step of the trajectory to one of the C microstates. The microstate trajectory $z_\mu(t) : \mathbb{R} \to \mathbb{R}$ contains for each time step t one microstate of the C clusters.

The number of microstates of 200 is chosen[1], since

- the number is sufficient to describe the dynamics,
- this number does not destroy the dynamics by accidentally joining energy-separated microstates.

In order to be able to apply PCCA a transition matrix needs to be constructed. This transition matrix is constructed by evaluation and counting transitions in the microstate trajectory $z_\mu(t)$. Thus, the obtained transition matrix $T_\mu(\tau)$ is of dimension $\mathbb{R}^{200\times200}$. Here, we used the marker μ to denote that T is related to microstates.

5.4 Clustering with HMM-VAR

The approach for the execution of the clustering with HMM-VAR is given in the left branch of Figure 5.2. The trajectory $z(t)$ is dimension reduced via PCA to the first twelve PCA components, resulting in $z_{\mathrm{PCA}}(t)$. From this trajectory the first 1, 2, 3 or 5 dimensions are taken as observation input for HMM-VAR, denoted as $z_{\mathrm{PCAn}}(t)$. The initialization of HMM-VAR is performed by an initial path, for which the macrostate trajectory $z_m(t)$ obtained from PCCA is taken.

With this initialization the HMM-VAR clustering is performed for each of the dimensions above in combination with the lagtimes $\tau \in \{1, 10, 100, 500, 1000, 2000, 5000, 10000\}$. The EM algorithm is started with these input parameters. With the convergence of the EM algorithm, the Viterbi path is calculated from the hidden Markov model according to the HMM-VAR model parameters and given observation $z_{\mathrm{PCAn}}(t)$.

5.5 Comparison of Clustering Results

The evaluation of the quality of the clustering results of PCCA and HMM-VAR is extremely difficult. The first problem is the missing availability of a reference clustering, thus a clustering which can be regarded as optimal in the sense of metastable states. Since this reference system is not available the only possibility to compare the obtained clustering results is to define a measure which compares the different clustering results of PCCA and HMM-VAR with respect to the membership assignment.

A visual representation and comparison of the clusters is not possible. Instead we compare the Viterbi path of HMM-VAR with the macrostate trajectory of PCCA by a comparison error function, which is derived below.

The second reliable comparison criterion available are the implied timescales, which deliver information about the timescales of switching clusters / metastable states.

[1] Basically, a serious analysis on the number of required microstates has to performed. This work has already been done by Jan Wigger during his investigation of the MR121-GSGSW system.

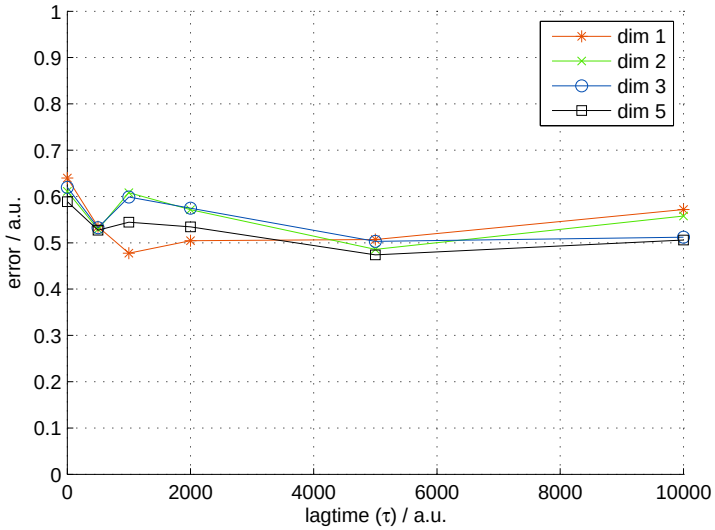

Figure 5.3: Error of agreement between Viterbi path of HMM-VAR for dimensions $\{1, 2, 3, 5\}$ and macrostate trajectory obtained by PCCA, measured with the error function E.

5.5.1 Measure for Comparison of Clustering Results

A membership assignment is represented as a matrix $M \in \mathbb{R}^{T \times C}$. This matrix has T rows according to the length of the trajectory. Each row of the matrix has C entries and represents the cluster assignment at time step t, with $0 < t < T$. So, an entry M_{ij} denotes the membership at time $t = i$ for a cluster j. Furthermore, the membership for a certain time step i sums up to 1 as

$$\sum_{j=1}^{C} M_{ij} = 1.$$

In order to compare two membership assignments with each other, a function is derived which delivers a reliable measure. Fur further analysis the following measure

$$E(M, M') = \frac{1}{T} \sum_{i=1}^{T} \left(\frac{1}{2} \sum_{j=1}^{C} |M_{ij} - M'_{ij}| \right) \tag{5.1}$$

is chosen, where both matrices M and M' are of dimension $\mathbb{R}^{T \times C}$. This error measure $E(M, M')$ delivers values between 0 and 1: 0 for the case that the assignments of M and M' agree for every time step, 1 if the membership assignment totally mismatches.

The macrostate trajectory from PCCA and the Viterbi path of HMM-VAR have been compared by this error measure E. Precisely, the Viterbi path of HMM-VAR for each of the input dimensions $\{1, 2, 3, 5\}$ has been compared with the PCCA macrostate trajectory. This comparison was performed for each lagtime τ.

Since the numbering of metastable states is arbitrary for PCCA and HMM-VAR a direct comparison of the membership matrices with the above error function E is not possible. Thus, it is not possible to say which metastable state i of PCCA corresponds to which metastable or hidden state j of HMM-VAR. Thus, an assignment of each state i to exactly one state j is required. The "correct" assignment will result in the lowest error of E.

A first attempt to try all permutations of assignment must be given up due to the long computation time of $O(n!)$, with n being the number of clusters. In our case n is 10.

Therefore, we have used the Hungarian algorithm [14], which is an optimization algorithm to solve assignment problems in $O(n^3)$. The Hungarian algorithm uses a (cost) matrix D as input, which defines in its entry d_{ij} the costs to assign i to j. In order to construct this matrix, each entry is calculated as follows (compare Equation 5.1):

$$d_{ij} = \frac{1}{T} \sum_{k=1}^{T} \left(\frac{1}{2} |M_{ki} - M'_{kj}| \right).$$

The output of the Hungarian algorithm is the best assignment with minimal costs. Here, it is the best assignment of metastable states i to hidden/metastable states j.

The results are shown in Figure 5.3. The results show a significant discrepancy between HMM-VAR and PCCA, the agreement of clusters lies slight below 50%, since the error is above 0.5. The error is neither heavily influenced by the lagtime τ nor by the number of dimensions for HMM-VAR.

5.5.2 Timescales

The true timescales are obtained by the calculation of the eigenvalues of the transition matrix $T_\mu(\tau)$. The transition matrix is setup from the microstate trajectory $z_\mu(t)$, which was obtained from k-means clustering of state space coordinates. The true timescales are plotted in Figure 5.4.

The implied timescales are calculated from the macrostate trajectory $z_m(t)$, which is the result of PCCA clustering. As can be seen from Figure 5.4, PCCA underestimates the implied timescales for the investigated lagtimes τ. But for increased lagtime τ the implied timescale converge to the true timescales.

A plot of the implied timescales from HMM-VAR clustering is given in Figure 5.5. For low-dimensional projections of dimension 1 and 2, the implied timescales are reasonable. With increased dimension the ITS drifts down to a regime of faster processes. At this point we believe, that this may be due to the approximation of conformational states via Gaussians within HMM-VAR. The description through that model can possibly be not appropriate. Second, we reckon, that due to the increasing number of fit parameters in higher dimensions the correct parametrization becomes more difficult.

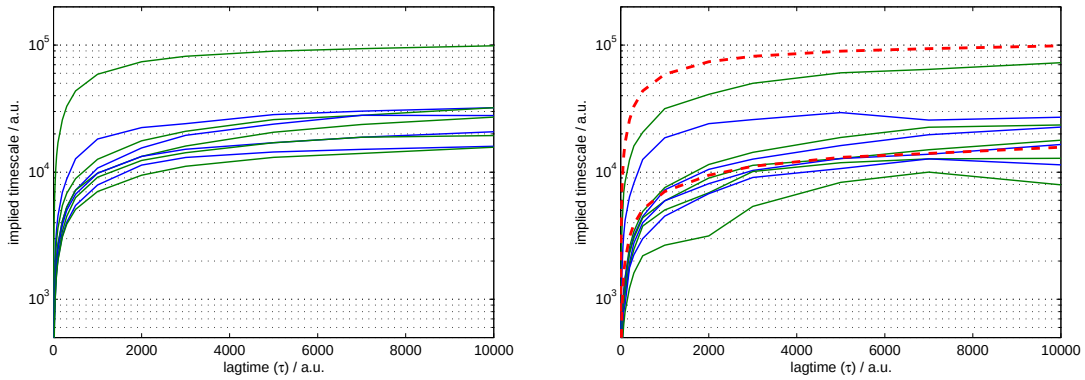

Figure 5.4: Left: True timescales obtained by the eigenvalues 2-9 of the transition matrix $T_\mu(\tau)$. Right: Timescales for PCCA, obtained by eigenvalues 2-9 of the transition matrix $T_m(\tau)$. The transition matrix is constructed from the macrostate trajectory $z_m(t)$, which is the result of PCCA clustering.

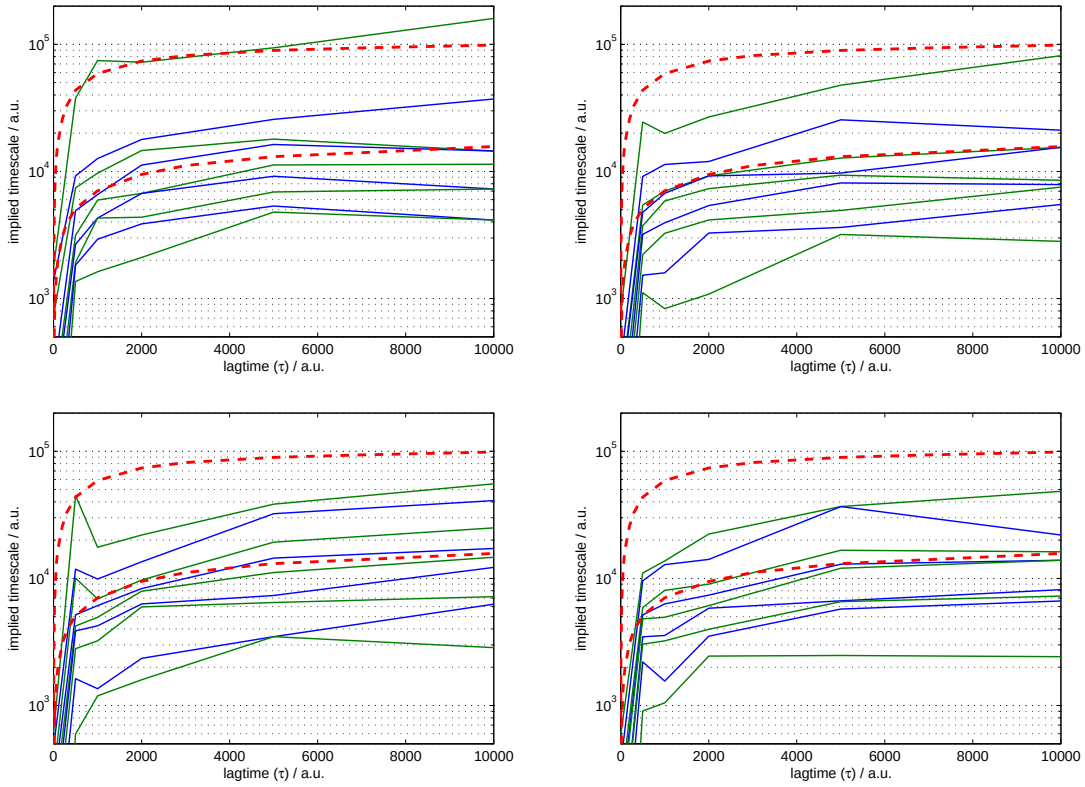

Figure 5.5: Timescales obtained by HMM-VAR for dimension 1 (top left), dimension 2 (top right), dimension 3 (bottom left) and dimension 5 (bottom right).

6 Conclusion

The analysis, evaluation and comparison of the two dynamically clustering methods PCCA and HMM-VAR is the main purpose of this thesis.

By the successful application of PCCA and HMM-VAR on several model trajectories and on a molecular dynamics trajectory of the MR121-GSGSW system, this aim is achieved.

Therefore, a new implementation of HMM-VAR was tested on model trajectories and compared with results of an old implementation. Afterwards, the HMM-VAR implementation was extended to calculate the maximum likelihood for multiple trajectories simultaneously, thus allowing the application of HMM-VAR to a trajectory for different lagtimes τ.

The extended HMM-VAR for multiple trajectories was applied to model trajectories. Failures in clustering of HMM-VAR were shown and investigated. A comparison with clustering results of PCCA was performed.

The problem of the convergence of the Baum-Welch algorithm to local minima during likelihood optimization was analyzed. A resort with the use of Genetic Algorithm was investigated. A fully object-oriented implementation of a genetic algorithm was established and integrated into the MolTools framework. Furthermore, the genetic algorithm was implemented, such that it could be applied to HMM-VAR problems. Then the genetic algorithm, placed on top of the Baum-Welch algorithm, was successfully applied to a model trajectory and showed the possibility to escape the local minima convergence problem.

The results of an molecular dynamics simulation for the MR121-GSGSW peptide were clustered using HMM-VAR and PCCA. A comparison of the clustering results was performed.

6.1 Outlook

Several aspects seem to be worth further investigations.

During the comparison of clustering results of the MR121-GSGSW system it was apparent that the lack of a reference system, which is reliably clustered in the sense of metastability, is a real obstacle. Such a reference system would allow absolute statements about the quality of clustering methods.

While searching for a reference a second approach could be the combination and comparison with experimental results.

Due to long expected calculation times it was not possible to apply the genetic algorithm on top of HMM-VAR to the MR121-GSGSW peptide. Since the genetic algorithm seems to be a promising candidate the quality of clustering results can be hopefully improved.

Nevertheless the approach of directly using HMM-VAR parameter as individual genes for the genetic algorithm seems questionable. The HMM-VAR is sensitive to these parameters and "bad" parameters cause serious numerical problems. So instead of the use of these parameters the individuals of the genetic algorithm could use Markov chains of the length of the observation and apply genetic operations to that Markov chains. This would hopefully circumvent numerical problems.

Instead of using a genetic algorithm, the method of simulated annealing seems promising in combination with HMM-models.

From the mathematical point of view a clean derivation of the likelihood and estimators for HMM-VAR with multiple trajectories is desirable.

Bibliography

[1] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models, 1997.

[2] R.E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998.

[3] P. Deuflhard and M. Weber. Robust perron cluster analysis in conformation dynamics. *ZIB Report*, 03-09, 2003.

[4] S. P. Elmer, S. Park, and V. S. Pande. Foldamer dynamics expressed via markov state models. ii. state space decomposition. *J. Chem. Phys.*, 123:114903, 2005.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Professional, January 1989.

[6] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[7] I. Horenko, E. Dittmer, A. Fischer, and Ch. Schütte. Automated model reduction for complex systems exhibiting metastability. *Multiscale Model. Sim.*, 5:802–827, 2006.

[8] I. Horenko, C. Hartmann, C. Schütte, and F. Noé. Data-based parameter estimation of generalized multidimensional Langevin processes. *Phys. Rev. E*, 76:016706, 2007.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[10] F. Krieger, B. Fierz, O. Bieri, M. Drewello, and T. Kiefhaber. Dynamics of unfolded polypeptide chains as model for the earliest steps in protein folding. *J. Mol. Biol.*, 332:265–274, 2003.

[11] W. J. Krzanowski, editor. *Principles of multivariate analysis: a user's perspective.* Oxford University Press, Inc., New York, NY, USA, 2000.

[12] G. F. Lawler. *Introduction to Stochastic Processes.* Chapman and Hall, Boca Raton, FL, USA, 2006.

[13] E. Meerbach and C. Schuette. Sequential change point detection in molecular dynamics trajectories, 2008, to be published.

[14] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[15] F. Noe, I. Daidone, J. C. Smith, A. di Nola, and A. Amadei. Solvent electrostriction-driven peptide folding revealed by quasi-gaussian entropy theory and molecular dynamics simulation. *Journal of Physical Chemistry B*, 112(35):11155–11163, 2008.

[16] C. Schuette and I. Horenko. Likelihood-based estimation of multidimensional langevin models and its application to biomolecular dynamics, multiscale modeling and simulation, 2007, to appear.

[17] C. Schütte, A. Fischer, W. Huisinga, and P. Deuflhard. A direct approach to conformational dynamics based on hybrid monte carlo. *J. Comp. Phys.*, 151:146–168, 1999.

[18] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[19] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen. GROMACS: Fast, Flexible and Free. *J. Comp. Chem.*, 26:1701–1718, 2005.

[20] W. F. van Gunsteren and H. J. C. Berendsen. Computer simulation of molecular dynamics: Methodology, applications and perspectives in chemistry. *Angew. Chem. Int. Ed. Engl.*, 29:992–1023, 1990.

[21] N. G. van Kampen. *Stochastic Processes in Physics and Chemistry.* Elsevier, Amsterdam, 4th edition, 2006.

[22] M. Weber. Improved perron cluster analysis. *ZIB Report*, 03-04, 2003.

[23] M. Weber. *Meshless Methods in Conformation Dynamics.* PhD thesis, Freie Universitaet Berlin, 2006.

Acknowledgments

I want to thank everybody who made this thesis possible. Especially my gratitude involves:

- Dr. Frank Noe for being a superb supervisor. His 24/7 availability in all concerns was of great use and needs to be mentioned explicitly.
- Prof. Dr. Cristof Schütte and Dr. Frank Noe for giving me the chance and possibility to perform this thesis.
- Eike Meerbach for feeding me with the mathematical background of HMM-VAR.
- Jan-Hendrik Prinz for sharing his experience and knowledge of PCCA.
- Martin Held for fruitful discussions and help in many fields.
- Jan Wigger for providing his knowledge and experience of coarse-graining the MR121-GSGSW system.
- My roommate Arash Ashand for many interesting discussions, which have illuminated the theoretical physical background of stochastic processes.
- Axel Rack and Christopher Ozdoba for their help and experience with issues related to software architecture and design.
- Everybody out of the BioComp Group for being part of such a wonderful working group, whose social climate and environment is so pleasant due to the excellent cooperation of its members.
- Last, but most important, to my friends, to my family and to **Julia** for their patience and confidence and their support in all concerns.

Affirmation

Hereby I, Martin Fischbach, declare that the submitted diploma thesis is the result of my own work and has been conducted independently. Furthermore, I declare, that no other sources than those indicated were used. The thesis has not been presented to any other examination committee before nor has the thesis been published before.

Date .....

Signature .....