JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

EMMA: A Software Package for Markov Model Building and Analysis

Martin Senne, Benjamin Trendelkamp-Schroer, Antonia S.J.S. Mey, Christof Schütte, and Frank Noé*

Department for Mathematics and Computer Science, FU Berlin

ABSTRACT: The study of folding and conformational changes of macromolecules by molecular dynamics simulations often requires the generation of large amounts of simulation data that are difficult to analyze. Markov (state) models (MSMs) address this challenge by providing a systematic way to decompose the state space of the molecular system into substates and to estimate a transition matrix containing the transition probabilities between these substates. This transition matrix can be analyzed to reveal the metastable, i.e., long-living, states of the system, its slowest relaxation time scales, and transition pathways and rates, e.g., from unfolded to folded, or from dissociated to bound states. Markov models can also be used to calculate spectroscopic data and thus serve as a way to reconcile experimental and simulation data. To reduce the technical burden of constructing, validating, and analyzing such MSMs, we provide the software framework EMMA that is freely available at https://simtk.org/home/emma.

INTRODUCTION

Molecular dynamics (MD) or related simulation approaches are commonly used to investigate various complex processes on the molecular level. Examples include ion or water transport through pores,[1] protein folding,[2−6] the formation of polymer melts,[7] protein−ligand binding,[8,9] macromolecular aggregation,[10,11] and conformational transitions.[12−14] While these systems have high-dimensional conformation state spaces, many interesting processes occur as transitions between a relatively small number of subsets of this large state space. The difficulty of dealing with molecular systems stems from the fact that there is usually very little a priori knowledge available to help with the characterization of relevant states. Hence, good reaction coordinates reducing the high dimensionality of the system are hard to find. A projection onto a low-dimensional subspace of user-defined order parameters can thus lead to deceptive results.[15−17]

Moreover, many of these molecular processes involve rare events which require one to accumulate a large amount of sampling data from numerical simulations. Recent advances in computing technology have enabled researchers to substantially increase the amount of data available from direct simulations. This development has in particular been fostered by fast simulation codes,[18−20] public access to supercomputers, and efficient use of GPUs for molecular dynamics simulation.[21−24] Nowadays, up to aggregate millisecond simulation data can be generated with distributed computing frameworks such as folding@home[25] and GPUgrid, the Anton MD supercomputer,[26] and supercomputers.[27]

The combination of large amounts of trajectory data and the fact that relevant states are *a priori* unknown calls for efficient and objective ways to analyze the simulation data. Additionally, it would be desirable to start subsequent simulation in such as way that statistical accuracy is achieved with minimal sampling effort, based on the knowledge of the already obtained data. Markov (state) models (MSMs) address this problem and have received a surge of interest in the past few years.[14,16,17,28−36] In MSMs, the molecular state space is discretized into microstates, and the transition probabilities or rates between these microstates are estimated from the available simulation data. Due to the high dimensionality of macromolecular systems, microstates can usually not be defined in terms of a grid discretization but by a clustering approach. The resulting transition or rate matrix can then be analyzed in order to gain insight into the relevant metastable states,[31,37,38] the essential (slow) dynamical processes and their time scales,[6,39] and transition pathways between substates of special interest (such as unfolded and folded subsets).[6,8,40] It has also been shown that MSMs can be used to systematically reconcile simulations with experimental data, e.g., obtained from kinetic experiments such as temperature-jump, fluorescence correlation, or time-resolved infrared measurements.[39,41−44]

Despite the substantial advantages of MSM analysis, simple yet potentially misleading analysis techniques such as principal component analysis, coordinate mapping, and histogramming are still much more widely used. This may be due to the fact that the construction and analysis of MSMs is technically more challenging. In order to make MSMs more accessible, we provide EMMA: an easy to use software package/framework for Markov model building and analysis. The EMMA software is free and can be obtained from https://simtk.org/home/emma.

Another currently available MSM software package is MSMbuilder.[45] At the present stage, MSMbuilder and EMMA have similar functionalities permitting the basic operations of MSM construction (data clustering, transition matrix estimation, PCCA, transition path theory). MSMbuilder 2 has been optimized for rapid RMSD clustering and is suitable for constructing MSMs with very large numbers of clusters from data sets as they are generated by folding@home. In contrast, EMMA 1.3 puts more focus on smaller MSMs that can be statistically validated and provides tools such as the Chapman−Kolmogorow test for validating the MSM and transition matrix sampling for calculating the statistical uncertainties of quantities of interest. However, these features are version-specific and may change in future versions. While MSMbuilder is written in python, EMMA is Java-based.

Received: April 11, 2012
Published: May 15, 2012

   dx.doi.org/10.1021/ct300274u | *J. Chem. Theory Comput.* 2012, 8, 2223−2238

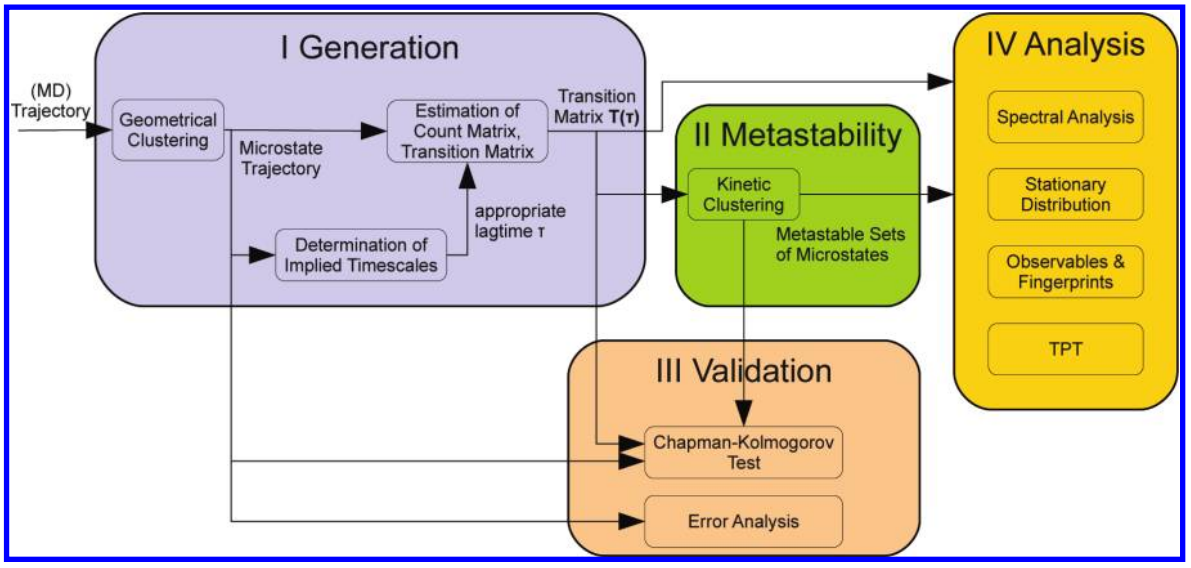

Figure 1. The steps involved in construction, validation, and analysis of Markov state models.

Currently, both packages are used through a command-line interface.

At this point, we would like to issue a word of caution. The construction of Markov models still involves a lot of choices to be made by the user (distance metric, clustering and estimation methods, several parameters), which involve some degree of experience to be made correctly. We are not yet at a stage where we can leave the choices involved in the construction of a Markov model to the algorithm altogether in order to hide all complexity from the user. Therefore, default parameters may not be appropriate in all situations, and EMMA is provided to "use at your own risk". The choice of some of the parameters and algorithms and appropriate references to the literature are given in this paper.

EMMA is written in Java. The EMMA command line tools for construction, validation, and analysis of MSMs can be invoked directly from the command line, making it suitable for use with bash scripts. Future versions of EMMA will expose an application programming interface (API) that is accessible from Java, Java-compatible user interfaces such as Matlab or Mathematica, and Python.

Here, we focus on the command-line-based MSM construction with EMMA that can roughly be described by the following sequence of steps:

1. **Generation of the MSM** from simulation trajectories—this step consists of the following:
   a. Clustering of the simulation data and assignment to microstates—currently, we support trajectory input from files in xtc (Gromacs), dcd (Charmm/NAMD), and ASCII formats; available clustering methods are k-centers, k-means, and equidistant clustering in space or time (Figure 1)
   b. Assignment of all trajectory frames of the input to discrete microstates using a Voronoi discretization (Figure 2)
   c. Ensuring that the microstates used to build the MSM are dynamically connected (Figure 2)
   d. Determination of an appropriate lagtime τ (the time resolution of the MSM, Figure 2)
   e. Maximum-likelihood estimation of a transition matrix $\mathbf{T}(\tau)$ describing the transition probabilities between microstates (Figure 3)

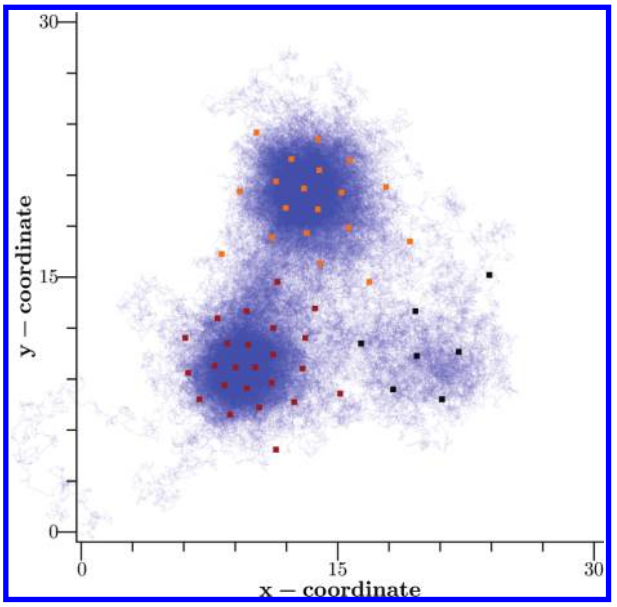

Figure 2. Illustration of the microscopic Brownian dynamics in a two-dimensional energy landscape given by a three basin model potential. Thirty microstates (rectangular dots) were determined by k-means clustering. Three metastable sets of microstates were obtained by the PCCA algorithm after MSM estimation. The coloring of each microstate represents its affiliation to one of the three metastable sets.

2. Determination of the **metastable sets** by means of kinetic clustering using the PCCA + method (eq 7)
3. Validation of the MSM
   a. Chapman−Kolmogorow test to compare long-time probabilities of states predicted by the MSM with those directly estimated from trajectory data (eq 1)
   b. Calculation of statistical uncertainties using transition matrix sampling, if desired (Figure 2)
4. **Analysis** of the transition matrix, e.g., by
   a. Calculation of the stationary probability distribution on microstates, ensemble averages of molecular observables, or the free energy differences between microstates (Figure 4)

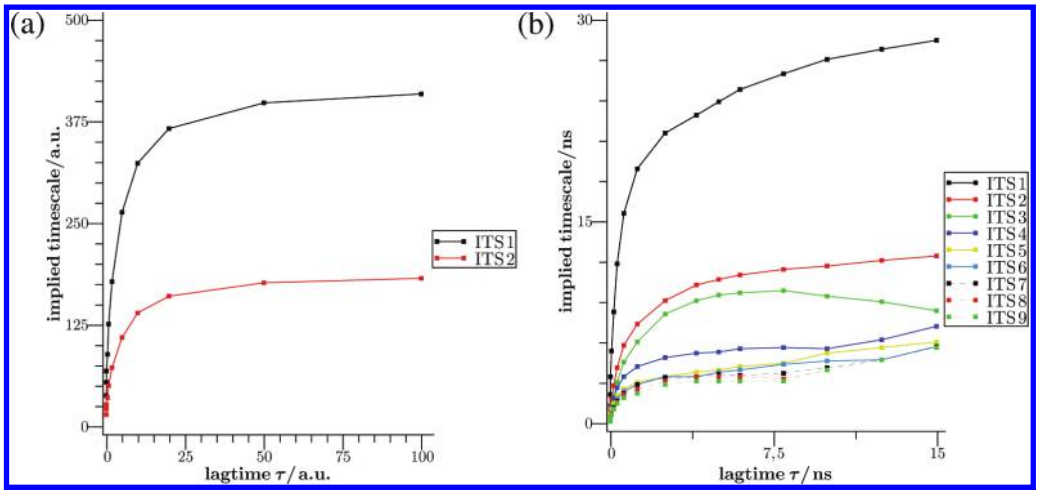

Figure 3. (a) Implied time scales of the MSM for the two-dimensional model of Figure 2 with 30 microstates. (b) Implied time scale of a MSM for the conformation dynamics of the MR121-GSGS-W peptide with 800 microstates.

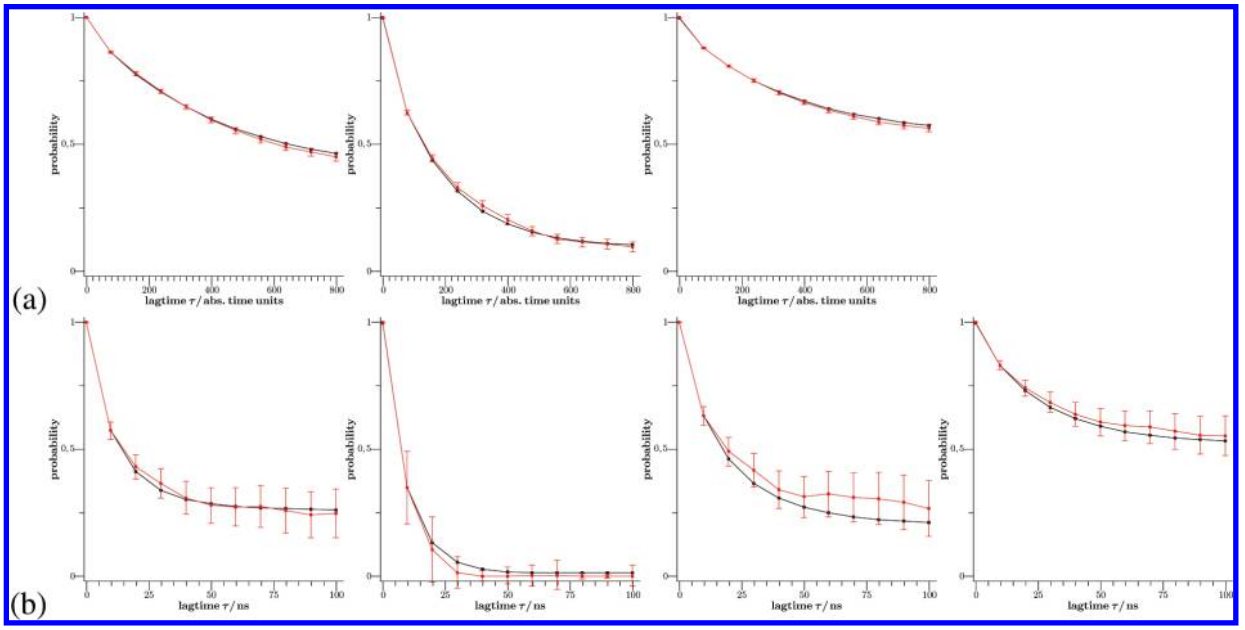

Figure 4. Results of the Chapman−Kolmogorov test (a) for the three metastable sets of the 2D-diffusion example of Figure 2 and (b) for four metastable sets of the MR121-GSGS-W peptide dynamics.

b. Spectral analysis, i.e., determination of the slowest relaxation time scales of the molecular process and the associated structural rearrangements (eq 13)
c. Calculation of transition pathways between subsets of special interest (e.g., unfolded→folded or dissociated→bound), see Figure 5
d. Calculation of dynamical observables such as the evolution of an experimental observable in a perturbation−relaxation experiment or time-correlation functions of experimental observables—these observables can be compared to kinetic experiments such as temperature-jump or fluorescence correlation spectroscopy (see eq 26)

These steps are discussed in the subsequent sections. For a detailed documentation of the EMMA syntax, we refer to the EMMA documentation and tutorial available at https://simtk.org/home/emma.

GENERATION OF MARKOV STATE MODELS (I)

In the process of generating a Markov model, a mapping between the highly dimensional molecular trajectory conformational space onto a smaller, more manageable subspace is achieved.

To demonstrate the functionality of EMMA, two examples have been chosen. The first one is a 2-d Gaussian model potential with three metastable states. The model potential and exemplary analysis scripts are provided with the EMMA software and are described in greater detail in the documentation found at https://simtk.org/home/emma.

The second example is molecular dynamics simulation data of the MR121-GSGS-W peptide, which are also available for download. For more detailed information about this data set, see ref 39.

Determination of Microstates by Clustering. The first step in building a MSM is to discretize the molecular state space $\Omega \in \mathbb{R}^d$,where d is the dimensionality of the system, into

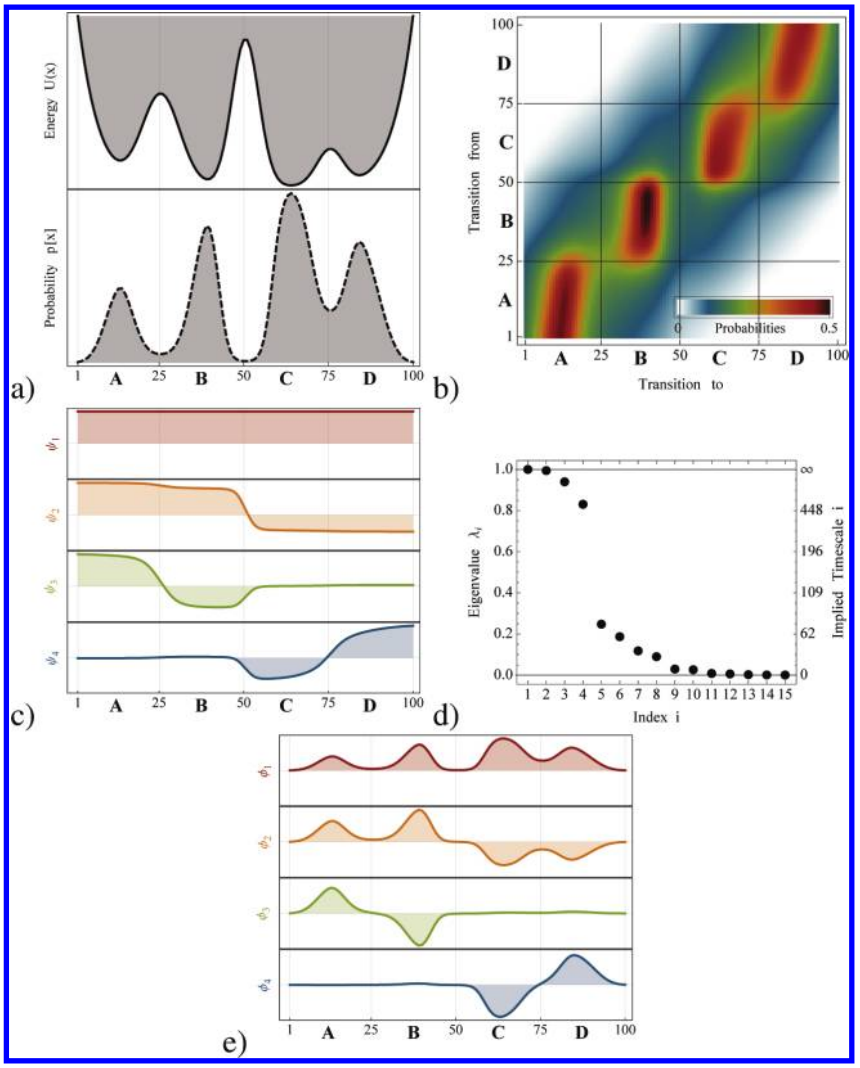

Figure 5. (a) Potential energy function with four metastable states and corresponding microscopic stationary density $\mu(\mathbf{x})$. (b) Density plot of the transition matrix for a simple diffusion dynamics in a 1D potential energy landscape. The 100 bins of the 1D-grid discretization correspond to the 100 microstates of the MSM. Red indicates high transition probability, and white indicates zero transition probability. Please note the nearly block-diagonal structure. The transition probability is large within the four blocks corresponding to the metastable sets. Rapid transition between states lying within a metastable basin are allowed, while probabilities for jumps between different basins are small. (c) The four dominant right eigenfunctions of the transition matrix, $\mathbf{r}_1$, ..., $\mathbf{r}_4$, which indicate the associated dynamical processes. The first eigenfunction is associated with the stationary process, the second to a transition between $A + B \leftrightarrow C + D$, and the third and fourth eigenfunctions to transitions between $A \leftrightarrow B$ and $C \leftrightarrow D$, respectively. (d) Eigenvalues of the transition matrix. The gap between the eigenvalues of the four metastable processes ($\lambda_i \approx 1$) and the eigenvalues of the fast processes is clearly visible. (e) The four dominant left eigenvectors, $\mathbf{l}_1$, ..., $\mathbf{l}_4$, of the transfer operator after weighting with the stationary density. Figure adapted with permission from ref 36. Copyright 2011, American Institute of Physics.

a microstate space X, which is defined on a clustering (set partition) $C = (C_1, ..., C_n)$ of Ω (i.e., $\cup_i C_i = \Omega$), by dividing the simulation data into these clusters. A microstate is defined as the set of molecular configurations exhibiting geometrical or structural similarity. EMMA provides a number of clustering tools for this purpose. The user can skip this step and employ a specific microstate definition obtained from other means than by the EMMA clustering command and proceed with the assignment step (Figure 2). The mm_cluster command requires three inputs: (1) trajectory data, (2) a clustering algorithm together with the desired number of cluster centers generated by the algorithm, and (3) an output destination for the cluster center coordinates. In the current version, cluster center coordinates can be written to file in either Charmm/NAMD (*.dcd) format, if the input trajectory was in a binary format, or ASCII format, if the input was in ASCII format.

The *input trajectory data* file formats may be either Gromacs (*.xtc), Charmm/NAMD (*.dcd), or plain ASCII format. It may consist of a single trajectory or multiple/many trajectories. A subset of all available trajectory frames for clustering can be specified using the -stepwidth r option, which only uses every rth trajectory frame as input for the clustering. Since simulation trajectories are stochastic samples of the equilibrium distribution of the molecular system, a substantial reduction of the input data is often possible and a representative clustering is still obtained. This option is especially useful to save computing time when clustering algorithms such as k-means or k-centers are employed, which require several iterations over the input data.

The *number of clusters* specifies the total number of microstates of the Markov model. The number of microstates has a severe influence on the quality of the Markov model. A small number of microstates may lead to microstates that

contain kinetically separated regions, thus leading to a poor Markov model. Increasing the number of microstates generally improves the quality of the MSM by reducing the discretization error[36] but may increase the negative effect of limited statistics. Please note that a high number of microstates will also make the vector and matrix operations involved in Markov model building and analysis more expensive in terms of required memory and running time of the algorithms. Iterative approaches to obtain a clustering can become very slow if the number of desired cluster centers is very large. The impact of the number of microstates on the quality of the Markov model will be further addressed in eq 8. Typical numbers of microstates are 100s to 10 000s.

In order to cluster data, a *metric* is required which assigns a distance $d(\mathbf{x}, \mathbf{y})$ to pairs of data points $\mathbf{x}$ and $\mathbf{y}$. When the overall position and orientation of a molecular system is not of interest, it is desirable to use a metric that captures intramolecular structural changes. The minimal RMSD metric[46] has shown to be suitable in protein folding and in large-scale protein conformational changes. Minimal RMSD is a proper distance metric[47] for situations in which the entire molecule can exhibit structural changes, but its global position and orientation is not of interest. In the case that the simulated system is fixed or can be meaningfully aligned to a reference structure, the direct Euclidean metric is also useful. Such a metric could be useful for the description of, e.g., conformational transitions that affect only parts of the molecule, transport processes through a pore, or the binding of a small ligand to a protein.[9] The Euclidean metric may also be used when the input data consists of angles (e.g., dihedral angle values for investigating peptide dynamics). Note that it is not strictly necessary to account for angle periodicity by the clustering metric—an artificial splitting of the data at the −180/+180 degree boundary may split kinetically connected states, but such kinetic information is contained in the transition matrix that is being estimated from the transitions between microstates. However, unnecessary splitting in angle microstates can be avoided by transforming each angle ϕ into two values ($\sin\phi$, $\cos\phi$) and performing the clustering on this extended set of input coordinates.[48]

Different clustering algorithms (EMMA option -algorithm) can be applied to determine cluster centers. The following clustering algorithms are implemented:

- **K-Means**: a well-known and established clustering approach that partitions the input data points $\mathbf{z}(t)$ into sets $C = \{C_1, ..., C_n\}$ such that the pairwise distances between points within the same cluster are minimal. Mathematically, $C = \arg\min_{C} \sum_{i=1}^{n} \sum_{\mathbf{z}(t)\in C_i} d(\mathbf{z}(t),c_i)^2$. This is achieved by iterating two steps for all clusters $i = 1...n$:
  - Voronoi assignment of $\mathbf{z}(t)$ to cluster representatives $\mathbf{c}_i$
  - Update clusters by: $\mathbf{c}_i = (1/|S_i|)\sum_{\mathbf{z}(t)\in S_i} \mathbf{z}(t)$.

  k-means is a good choice when internal or Cartesian coordinates are used.[9,37] Even if relatively few coordinates are used, such as the three-dimensional position of a ligand relative to a protein,[8,9] k-means significantly outperforms partitioning the space with a grid, because it focuses on the regions of space that are dense with data. Please note that K-means cannot be used with the minimal RMSD metric because there is no well-defined way of calculating a cluster center $\mathbf{c}_i$ as a mean of several existing structures. If the minimal RMSD metric is desired, please use one of the following clustering algorithms.
- **K-Centers**[49] is a fast algorithm to partition the input data points $\mathbf{z}(t)$ into sets $C = \{C_1, ..., C_n\}$ in a way such that the optimization problem $\sum_{i,j,k,l} \max_{\mathbf{x}_i\in C_j} \min_{\mathbf{y}_k\in C_l} d(\mathbf{x}_i,\mathbf{y}_k)$ is approximated.

Here, a simple and fast greedy approximation algorithm is used: The first iteration chooses an arbitrary data point. In the ith iteration, the data point that has the largest distance from all previously selected cluster centers $\mathbf{c}_1, ..., \mathbf{c}_{i-1}$ is picked, and the next cluster center $\mathbf{c}_i$ represents that point. In other words, $\mathbf{c}_i = \arg\max_{\mathbf{z}(t)} \sum_{m=1}^{i-1} d(\mathbf{z}(t),\mathbf{c}_m)$. Note that this version of k-centers has a tendency to find outliers as representatives which might only provide a good clustering for large numbers of clusters.[45] K-centers is not the recommended way of clustering data but was included to allow comparison with previous studies that have employed this clustering technique. In practice, regular spatial or regular temporal clustering achieves much better results.

- **Regular spatial clustering**: Clusters are chosen to be approximately equally separated in the conformation space with respect to the distance metric used. The distance between cluster centers is controlled by the parameter $d_{\min}$. In detail, the cluster centers are determined as follows:
  - The first data point $\mathbf{z}(t = 0)$ is taken to be the first cluster center $\mathbf{c}_1$. Let $n = 1$ be the current number of clusters.
  - Iterate data points $\mathbf{z}(t)$:
    When a data-point $\mathbf{z}(t)$ is found, for which $d(\mathbf{z}(t),\mathbf{c}_i) > d_{\min}$ is fulfilled for all $1 \le i \le n$ existing $\mathbf{c}_i$, then $\mathbf{c}_{n+1} = \mathbf{z}(t)$ becomes a new cluster center and n is incremented.

  Regular spatial clustering guarantees that the conformation space is partitioned in a roughly equidistant manner. Despite its simplicity, we have found it to be a good way to build microstates for an MSM.[36] Note that the total number of clusters k that will be obtained by the method strongly depends on the choice of the threshold $d_{\min}$.
- **Regular temporal clustering**: Given a step length r, every rth data point of the input trajectory set is selected as a cluster center. If the trajectory has length N, then $n = \lfloor (N/r) \rfloor$ cluster centers are selected. This approach will generate a sensible model if the system was equilibrated and the initial nonequilibrium part of the trajectory was discarded before clustering. Trajectories which are long compared to the system's slowest relaxation time can usually be regarded as sufficiently equilibrated. Also a multiensemble simulation such as parallel tempering/replica-exchange molecular dynamics can provide suitable data for this clustering approach. In this case, the algorithm will pick cluster centers from the equilibrium distribution of the system of study.

As an output, the clustering algorithms provides a set of n cluster centers $C = \{\mathbf{c}_1, ..., \mathbf{c}_n\}$ representing molecular structures in conformation space that are used as an input for the subsequent assignment step.

Usually, the execution time of a noniterative clustering algorithm (k-centers, regular spatial, regular temporal) is $O(Nn)$. Here, N denotes the number of data points and n, the number of cluster centers. Due to the large amounts of data available, clustering and the subsequent assignment are often the most expensive steps in MSM construction, and it is often

desirable to reduce the dimensionality of the input data to the coordinates that are relevant for describing the process of interest before clustering (e.g., α-carbon coordinates, domain centers, backbone dihedrals, etc.).

Assignment of Conformations to Microstates. The mm_assign command maps trajectories in continuos conformation space, $\mathbf{z}(t) \in \mathbb{R}^d$, to microstate trajectories, $s(t) \in X$, where $X \in \{1, ..., n\}$ is the set of microstates, from n cluster centers. All trajectory frames are assigned to the closest cluster center c_i resulting in a Voronoi-partitioning of the input data. In other words, the microstate trajectory is obtained as $s(t) = \arg\min_i d(\mathbf{z}(t),\mathbf{c}_i)$. These discrete trajectories are used to estimate a Markov model.

The mm_assign command requires three inputs: (1) trajectory data ($\mathbf{z}(t)$), (2) a file in ASCII or .dcd format containing the cluster center coordinates, and (3) a destination for writing the discretized trajectories. If the cluster centers have been obtained by the mm_cluster command, modifications are still possible. This may be desirable, if X-ray structures are known and including these as cluster centers would make the Markov model more informative. If the cluster centers were user generated, extra care should be taken to ensure that the dimensionality of the cluster centers matches those of the input trajectories.

Connectivity Test. The program mm_connectivity tests which microstates are dynamically connected and writes the largest connected subset of microstates to the specified output. Two microstates i and j are said to be connected if there exists a set of trajectories by which the system can move from i to j and also from j to i. In other words, there is a nonzero probability to go from i to j and back in a finite number of steps. A set of states is said to be connected if all states in this set are pairwise connected.

Consider for example the following sequence of microstates forming a single microstate trajectory {1, 2, 1, 4, 3, 5, 4, 3, 5, 4, 6}. The microstate space $X = \{1, 2, 3, 4, 5, 6\}$ contains the following connected sets of microstates, $S_1 = \{1, 2\}$, $S_2 = \{3, 4, 5\}$, and $S_3 = \{6\}$. Within the given microstate trajectory, S_1 will not be visited again once it was left, and S_3 will not be left once it has been entered. Thus, S_2 is the largest dynamically connected subset of X. If there is only one fully connected subset S_1, then $S_1 \equiv X$.

A unique stationary distribution exists only for Markov models on a connected set of states. Since many MSM algorithms use the stationary distribution as a starting point for further calculations, it is important to work with a connected set of microstates. If the set of microstates is split into several connected sets, the largest set can be determined and written out. Information about the largest connected set of microstates can be used as input in later stages of the MSM building process.

There may be different reasons why the microstate space X is separated into several connected sets:

- There may be unused cluster centers in the assignment process. Thus, microstate trajectories do not contain microstates corresponding to such cluster centers. Empty states will appear as single states ("singlets") in the connectivity analysis. They can be removed without a loss of information. Please note that there are also other reasons for the appearance of singlets in the connectivity analysis.
- There are sets of states that are only visited at the beginning or at the end of a microstate trajectory, for example, the sets S_1 and S_3 in the above example.
- The clustering is too fine. It misses the fact that cluster centers lying in the same kinetic region would be properly connected if more simulation data were available. When low connectivity is reported for microstates coming from the same kinetic region, a coarser clustering may produce more satisfactory results.
- Weakly connected parts of state space appear as disconnected due to insufficient simulation data. This is often the case for simulations of biomolecules where the transition times between metastable sets can easily be on the order of or even greater than the total simulation time. The MSM analysis may be restricted to the largest connected set of microstates, but all information about the neglected part of state space will be lost. In situations in which disconnectivity results from a lack of observed transitions between sets that are expected to be connected, it will be necessary to generate more data to build a meaningful Markov model.

It can be seen from the discussion above that there are many reasons why the state space can be separated into disconnected sets of microstates. The discrimination of these different situations can be quite challenging for complex systems and may require further insight into the dynamics.

We will not go into depth discussing the technicalities involved in determining the connectivity here. The algorithm used is Tarjan's algorithm, and we refer to ref 50 for a detailed discussion.

The (on-screen) output of mm_connectivity comprises the microstates sorted into different connected sets. Each row of microstates corresponds to a single communicating class. The largest connected set of microstates is written to file in order to provide a starting point for a construction of a MSM with a unique stationary distribution.

Selection of the Lagtime τ: Implied Time Scales. Once a dynamically connected set of microstates is identified, the number of transitions between all pairs of microstates is recorded in a count matrix $\mathbf{C}(\tau)$. Its elements $c_{ij}(\tau)$ contain the number of times the trajectory visited in state i at time t and in state j at time $t + \tau$, for all times t. $\mathbf{C}(\tau)$ is then converted into an estimate of the transition matrix $\mathbf{T}(\tau)$, which, together with the microstate definition, comprises the Markov model. While the continuous microscopic dynamics underlying the input trajectories is Markovian in full phase space, the discrete jump process between clusters of state space is no longer Markovian. Hence, a Markov model on this space can only be approximate. The correctness of the Markov approximation is determined by two properties: The lagtime τ and the quality of the microstate definition.[36,51]

A transition matrix $\mathbf{T}(\tau)$ has eigenvalue−eigenvector pairs:

$$\mathbf{T}(\tau)\mathbf{r}_i = \lambda_i(\tau)\mathbf{r}_i \tag{1}$$

A direct application of the Markov property shows that the eigenvalues of a transition matrix estimated at lagtime τ should follow an exponential decay in τ. The time scale (inverse rate) of this decay is called implied time scale and is given by

$$t_i^*(\tau) = \frac{-\tau}{\ln|\lambda_i(\tau)|} \tag{2}$$

These implied time scales are relaxation time scales that are experimentally observable[39] and should therefore be independent of τ if the Markov model was exact. Since the discretized dynamics are *not exactly* Markovian, $t_i^*(\tau)$ *do* depend on τ. However, assuming sufficiently good statistics, their dependence of τ should diminish for larger values of τ.[36,51] If lagtimes τ are taken to be too short, the time scales will always be underestimated. This behavior was first observed in ref 52, and it was suggested by the authors to test whether the time scales $t_i^*(\tau)$ become approximately constant in τ after some minimal value τ' and to then use the Markov model with lagtime τ'. Transition matrices $T(\tau_k)$ are estimated for a set of increasing lagtimes $\tau_1 ... \tau_m$, and the corresponding largest implied time scales are calculated from eq 2 as a function of τ. The smallest value τ for which the implied time scales become approximately constant in τ is selected as lagtime τ for our Markov model.

The test for implied time scales is included in EMMA and is conducted by the command mm_timescales. The main input options are the discretized trajectories (via -i), the lagtimes τ_k for which the implied time scales will be estimated (via -lagtimes), and the number of eigenvalues or implied time scales for which the analysis will be conducted (via the option -neig).

In addition to these mandatory options, it can specified whether transition matrices are estimated such that they fulfill detailed balance with the (-reversible) flag. This option should be considered, if the dynamics evolve at thermal equilibrium. Adding prior counts to make the time scale calculation numerically robust when statistics are poor via the -prior option is also possible. See Figure 3 for a short discussion of counting approaches and priors used for the count matrix estimation. The dynamically connected set of microstates is specified by the option -restrictToStates. It is advisible to use the largest connected subset of microstates for the time scale estimation. The output of the algorithm is a table containing the implied time scales at each lagtime that is either printed on the console or redirected into the output file when specified.

In practice, this test may behave in an unexpected way for a number of reasons:

1. When $\tau \gg t_i^*$, the numerical solution of the eigenvalue problem (eq 1) can fail. This can lead to an apparent linear increase of t_i^* for large lagtimes τ_k. This behavior can especially be observed for processes corresponding to short implied time scales and is not necessarily a signature of a poor Markov model.
2. The convergence of $t_i^*(\tau)$ to the true implied time scales occurs asymptotically with τ^{-1}, i.e., relatively slow. Convergence is especially poor if the clustering is poor.[36] Therefore, when poor convergence is observed, one may need to either refine the clustering or use very large lagtimes in order to obtain a Markov model with good approximation quality.
3. In the case of poor statistics, convergence in τ may not be observable.

Despite these issues, the implied time scale plot still provides a useful way to determine whether a good MSM can be found. Figure 3 shows implied time scales plots of Markov models of the 2D-test data set and the MR121-GSGS-W data set, both showing a reasonable convergence of the dominant implied time scales and allowing an appropriate choice of τ to be made.

Transition Matrix Estimation. The main purpose of the mm_estimate command is to generate a row-stochastic transition matrix $\mathbf{T}(\tau)$ using the discretized simulation trajectories and the selected lagtime τ as inputs. $\mathbf{T}(\tau)$ together with the definition of the microstate discretization comprises the actual Markov model and the main object of interest.

While transition matrix estimation was carried out for several lagtimes in order to generate the implied time scale plot as seen in Figure 3, it is now repeated a for the single lagtime τ that has been selected as the Markov model lagtime. It is assumed that the state space of our discrete input trajectories consists of a single completely connected set of microstates. If this is not the case, it is possible to restrict the transition matrix estimation to the largest connected subset using the -restrictToStates option.

In the following, the theory behind the transition matrix estimation procedure is briefly outlined. Consider a discretized trajectory $s(t)$ of total length $t_{\max}$. Each time step in $s(t)$ yields the index of one of the n microstates. The sequence of microstates given by $s(t)$ can be transformed into an $n \times n$ matrix of observed counts $\mathbf{C}^o(\tau)$ between the n microstates via

$$c_{ij}^o(\tau) = \sum_{t=0}^{t_{\max}-\tau} \delta_i[s(t)]\, \delta_j[s(t+\tau)] \tag{3}$$

where $\delta_i[s(t)] = 1$ if $s(t) = i$ and 0 otherwise. Thus, the observed count matrix element c_{ij}^0 is equal to the number of transitions between states i and j at lagtime τ that are contained in the given microstate trajectory $s(t)$. Since the count matrix is obtained from the trajectory by sliding a window of length τ along the trajectory, this method of counting is referred to as the *sliding window* method. The advantage of this method is that all transitions contained in the trajectory are included into the estimated count matrix. The resulting counts are however not statistically independent since the sliding window approach ignores the fact that the process is not Markovian at time scales smaller than τ. Statistically independent counts can be generated by using the *lagtime* counting method. See ref 36 for further discussions of the different counting approaches.

The user can choose to add a prior matrix $\mathbf{C}^p$ to avoid numerical problems with states that were rarely visited or never left:[53,54]

$$c_{ij} = c_{ij}^p + c_{ij}^o \tag{4}$$

The purpose of the prior $\mathbf{C}^p$ is to avoid numerical problems in scenarios with little data where trajectories contain insufficient transitions to conduct a numerically stable transition matrix estimation. Prior counts introduce a bias which is vanishing once c_{ij} is dominated by observed counts c_{ij}^0. In order to minimize the bias, the added prior counts should be as small as possible while guaranteeing numerical stability. In many situations, the *neighbor prior*[54,45] $c_{ij}^p = \alpha$ when $c_{ij}^o(1) + c_{ji}^o(1) > 0$ is a suitable prior for count matrix estimations. The neighbor prior ensures strict positivity of the stationary distribution on all microstates by adding a small pseudocount to neighboring entries of the count matrix whenever an element of the observed count matrix $\mathbf{C}^o$ is nonzero. The neighbor prior can be applied using the -prior option of the mm_timescales and the mm_estimate command. The count matrix can be written as a sparse matrix in coordinate format to an ascii file specified by the outputcountmatrix option.

The total number of outgoing transitions from state i is $c_i = \sum_{k=1}^n C_{ik}$. Following Bayes' theorem, the probability of a transition matrix $\mathbf{T}$ given the counts $\mathbf{C}$ is given by

$$\mathbb{P}(\mathbf{T}|\mathbf{C}^o) \propto \prod_{i,j} T_{ij}^{c_{ij}} \tag{5}$$

This probability is maximized by the maximum probability estimator $\hat{\mathbf{T}}_{ij}(\tau)$. When no constraints are imposed on $\hat{\mathbf{T}}_{ij}(\tau)$ other than that it is a stochastic matrix, the estimator is trivial:

$$\hat{\mathbf{T}}_{ij}(\tau) = \frac{c_{ij}}{c_i} \tag{6}$$

Molecular dynamics is usually conducted in thermal equilibrium, i.e., such that the equations of motion fulfill microscopic detailed balance. This translates to a detailed balance criterion for the microstates:

$$\pi_i T_{ij} = \pi_j T_{ji} \tag{7}$$

However, the simple maximum likelihood estimator of the transition matrix $\hat{\mathbf{T}}(\tau)$ in eq 6 will in general not fulfill a detailed balance as a result of statistical deviations from these constraints due to finite length simulation trajectories. Since a detailed balance of a Markov State Model is a prerequisite for the application of many advanced analysis methods, it is useful to enforce a detailed balance when estimating a transition matrix from a given count matrix. Previous work had suggested to enforce a detailed balance by using the sum of transition matrices from both forward- and backward-in-time counting.[37,55] However, this approach is only valid if the individual simulation trajectories are of such length that they sample from a global equilibrium. A better approach to enforce detailed balance is to build the constraints (7) into the transition matrix estimation.[36,56] EMMA implements the optimal reversible estimator for transition matrices described in ref 36. It is available through the -reversible option of the mm_estimation and mm_time scales commands.

METASTABILITY (II): LUMPING OF MICROSTATES TO METASTABLE SETS

A way to group microstates of a MSM into sets on which the model dynamics is metastable is a useful tool for investigating the essential characteristics of the underlying microscopic system.[37,38,47,57] Metastable sets are characterized by showing rapid interconversions between states lying within the set and a rare occurrence of transitions between different metastable sets. This grouping into dynamically similar sets, also known as kinetic clustering, can reduce the complexity of a MSM on a very large microstate space by identifying states belonging to the same kinetic cluster. This can be an enormous aid in visualization and further analysis.[37] Please note that metastable sets are not themselves used to calculate kinetic properties of a coarse grained MSM on the level of different kinetic clusters. Such a coarse graining would dramatically increase the error of the MSM[51] unless it is performed in a very specific manner.[57] Thus, the fine microstate model is always kept as a numerical means to approximately solve the molecular kinetics on a "fine grid", while kinetically grouped macrostates are useful for visualization of relevant sets or grouping of additive properties, such as the probability of states or transition fluxes (see below).

The method of choice for determining kinetic clusters from a transition matrix is PCCA, invented by Schütte et al.[38] and later improved by Weber and Deuflhard.[31,58] The robust version of PCCA (also called PCCA+)[58] is implemented by the EMMA command mm_pcca and described subsequently. Given a transition matrix $\mathbf{T}(\tau) \in \mathbb{R}^{n\times n}$ and a number of states $m < n$, PCCA assigns each of the microstates to one of m clusters $S_1 \ldots S_m$ that are metastable. PCCA makes this assignment in a fuzzy way; i.e., the primary result is not a clustering but a membership matrix $\chi \in \mathbb{R}^{m\times n}$ indicating by its elements χ_{ij} to what degree each microstate j belongs to metastable set m_i, where

$$\sum_i \chi_{ij} = 1 \;\forall\; j \tag{8}$$

This matrix is then used to generate a crisp assignment of sets in such a way that each microstate i is assigned to the metastable set m it most likely belongs to.

$$j \in M_i \text{ if } \arg\max_k \chi_{kj} = i$$

This assignment is made on the basis of first finding m microstates that are kinetic centers and therefore representative states for the metastable sets and then assigning kinetic distances by the coordinates all microstates have in the m-dimensional space of dominant eigenvectors of $\mathbf{T}(\tau)$. A more detailed description can be found in refs 31, 37, and 58.

The mm_pcca command implemented in EMMA can provide both microstate clustering and fuzzy memberships. The main input parameter is the transition matrix $\mathbf{T}(\tau)$ and the number of kinetic clusters/metastable sets which is set by the option -nclusters. The determined cluster assignments, either crisp or fuzzy, are written out by the options -ocrisp and -ofuzzy, respectively.

The result of PCCA clustering of the two-dimensional diffusion in a three-state energy landscape is shown in Figure 2. The PCCA clustering for an MSM obtained from 6 μs simulation data of the MR121-GSGS-W peptide is shown in Figure 7

VALIDATION (III)

The validation of the MSM is necessary in order to be able to judge the meaningfulness of the generated Markov model. If the Champman−Kolmogorov test discussed in the following holds, it is generally a good indication of the robustness of the model built and thus an essential part of the model validation process.

Chapman−Kolmogorov Test. A number of ways to test the validity of Markov models have been proposed in previous papers.[36,59−61] A rather direct and easy-to-interpret set of tests compares long-time observations generated from the estimated transition matrix $\hat{\mathbf{T}}(\tau)$ with long-time information available from the trajectory data.[36,61] In the following, we describe the Chapman−Kolmogorov test proposed in ref 36, which has been implemented in EMMA through the mm_chapman command. The Chapman−Kolmogorov equation is a strong test of the Markov property of the estimated model. Let $\hat{\mathbf{T}}(\tau)$ be the transition matrix estimated for lagtime τ, and let $\hat{\mathbf{T}}(k\tau)$ be the transition matrix estimated from the dynamics on a longer time scale. If the dynamics of the model is to a good approximation Markovian on the time scale of the lagtime τ, the model should be able to approximately reproduce the dynamics on the microstate space for longer times $t = k\tau$. Since the dynamics of the model is generated by repeated application of the transition matrix, the following approximation should be valid:

$$[\hat{\mathbf{T}}(\tau)]^k \approx \hat{\mathbf{T}}(k\tau) \tag{9}$$

If the microstate dynamics were perfectly Markovian, the above equation would have to strictly hold. That is, the long time dynamics of the system would have to be exactly determined by the transition matrix estimated from the short time dynamics

on scales of the lagtime τ. It is important that the test is conducted on time scales comparable with the time scale t_2 of the slowest dynamical process. If we would perform the test for longer times $k\tau \gg t_2$, we will only test how well the model approximates the stationary distribution and gain no information about the validity of the MSM for the description of the dynamical processes.

Matrices $\hat{\mathbf{T}}(\tau)^k$ and $\hat{\mathbf{T}}(k\tau)$ could be directly compared via a matrix norm. However, in order to arrive at a comparison that has a direct physical interpretation, we instead choose to propagate a distribution $\mathbf{p}_0$ using $\hat{\mathbf{T}}^k(\tau)$ while we are estimating the quantity $\mathbf{p}_0\hat{\mathbf{T}}(k\tau)$ directly from the trajectory. For a correct MSM we require

$$\mathbf{p}_0 \cdot [\hat{\mathbf{T}}(\tau)]^k \approx \mathbf{p}_0 \cdot \hat{\mathbf{T}}(k\tau) \tag{10}$$

Here, $\mathbf{p}_0$ is defined by the local stationary probability distribution confined to set C_i:

$$p_{0,i} = \begin{cases} \dfrac{\pi_i}{\sum_{k \in S_i} \pi_k} & i \in C_i \\ 0 & \text{else} \end{cases} \tag{11}$$

The test is visualized by plotting the total probability on each set S_i tested over times $k\tau$ and inspect whether the Markov model and the direct calculation agree within statistical error (see ref 36 for further details). That is, the dynamics are started from each of the metastable sets, and the evolution of the probability in that set toward the stationary probability is traced. This test effectively enforces that the approximation error in eq 10, $E(k\tau) = |\mathbf{p}_0 \cdot [\hat{\mathbf{T}}(\tau)]^k - \mathbf{p}_0 \cdot \hat{\mathbf{T}}(k\tau)|$, is bounded by the statistical error and thus enforces a balancing of these two errors. Thus, decreasing the statistical error by adding more simulation data will also increase the requirements for the Chapman−Kolmogorov test to be valid.

The program mm_chapman expects three main input parameters: (1) a previously estimated transition matrix $\hat{\mathbf{T}}(\tau)$, (2) the input trajectories, which were used to estimate the matrix $\hat{\mathbf{T}}(k\tau)$, and (3) the sets of states on which the propagation of eq 10 is tested. We suggest to use the PCCA sets calculated with mm_pcca at this point, as the metastable states are the ones which by definition have the fewest transitions between them. Thus, when successful, this test will validate that the MSM is a good model of the slow dynamics that are usually the ones of interest. Alternatively, user-specific sets, such as unfolded and folded sets in protein folding,[41] or random sets (-randomsets) can aslo be defined. The initial probability $\mathbf{p}_0$ may be specified by the user (option -pinit) or else the stationary distribution of $\hat{\mathbf{T}}(\tau)$ is used. The propagation length k is given in multiples of the lagtime τ selected by -kmax.

Figure 4 shows the test results for the 2D diffusion example available through the built in tutorial in EMMA as well as for the MR121-GSGS-W peptide. The sets chosen here are the three or four metastable sets that have been previously identified with PCCA (see above). Note that the agreement at the times $t = \tau$ and at $t \to \infty$ is always expected and is no test of the Markov model quality.[51] The critical test is whether the Markov model reproduces the relaxation for the range $\tau < t < 3t_2$*, where the upper time bound is of course limited by the maximal length of available trajectories.

Uncertainty Calculation. In Figure 3, a transition matrix was estimated whose maximum probability is given by eq 5. This maximum probability estimator (for a uniform prior called the maximum likelihood estimator) is a good estimator if the probability distribution is sharply peaked, and thus transition matrices that differ largely from the maximum probability estimator are very unlikely. Such a sharply peaked probability distribution exists if statistics are overwhelming, i.e., if all molecular processes (including the slowest ones) have been sampled frequently. Since this is usually not the case other than in toy models, it is important to calculate the statistical error of the estimate. The statistical errors can be essential for any future choice of starting points of simulations, with the aim to minimize statistical errors. This can be done adaptively and is generally referred to as adaptive sampling.[56,62,63] The statistical uncertainties of individual transition matrix elements can be directly calculated from the observed count matrix $\mathbf{C}^o$.[36] However, as a user, one is usually interested in the statistical uncertainties of potentially complex observables that are calculated from the transition matrix (see Figure 4 below). Therefore, a good approach is to sample the transition matrix probability density (eq 5) using a Markov chain Monte Carlo procedure.[64] Although this sampling method is computationally slower than using linear perturbation approaches,[54,62,63] it makes no restrictive assumptions about the form of the probability distribution sampled and the functional form of the observable used. Note that the uncertainties calculated from transition matrix sampling only contain the part of the statistical uncertainty from a finite number of transitions between microstates. For a full account of uncertainties, one may also need to include the uncertainties that arise from estimating the values of functions of states from samples in that state. A treatment of these combined uncertainties is described in ref 65.

Transition matrix sampling as described in ref 64 is implemented by the mm_transitionmatrixSampling command. As an input, the count matrix (-C) is required. Here, the choice of counts is imperative, as for obtaining correct uncertainties, all counts need to be statistically independent. This is not the case when the transition counts are generated from a sliding window approach (which is the default in transition matrix estimation). Therefore, -sampling lag should be used instead in order to generate statistically independent counts. Since this way of counting may reduce the connectivity of the transition matrix, the connectivity should be tested again and, if necessary, the count matrix restricted to the connected subset of states.

The most general way of using transition matrix sampling is to generate a series of sampled transition matrices (-sampleSaveT). The desired observable can then be calculated for each of them, and the statistical uncertainty can then be estimated from the standard deviation of the respective quantity. This approach is applicable to any numerical observable, no matter how complex it is. The number of sampling steps is controlled by three numbers: the number of samples (nsamples), the number of MCMC steps to initialize the sampling (steps_burn-in), and the number of MCMC steps for each sample steps_per-sample. The MCMC procedure does steps_burn-in steps first and then returns one transition matrix every steps_per-sample step. Typical values of these numbers are 1 000 000, 100, and 100 000, but these numbers may need to be increased for large matrices, making the sampling of these computationally demanding.

In order to make transition matrix sampling more easily usable, some common observables are directly implemented and can be accessed with the combination of the -sample and -observe... commands. For example, -observeSpectrum 2 will calculate the distribution of the two leading eigenvalues of the transition matrix. The output file will contain for each sample

the instantaneous sample value, the running mean, and the running variance. The mean and variance should be checked for convergence. If convergence is not achieved, the number of samples or the number of steps per sample must be increased.

ANALYSIS (IV)

In the following section, the analysis possibilities of the previously built MSM model will be discussed. Evidently, these tools can also be applied to transition matrices obtained through other sources than EMMA, as long as the file convention (see Appendix) is adhered to.

Stationary Distribution. A basic quantity of interest is the stationary probability π_i of any microstate i. When the molecular system under investigation is in thermal equilibrium with its environment, the stationary distribution is given by the average of the microscopic Boltzmann distribution over the Voronoi clusters A_i pertaining to each microstate i. The stationary distribution on the Markov model microstates can be calculated as

$$\pi^T = \pi^T \mathbf{T}(\tau) \tag{12}$$

that is, by computing the eigenvector of the transition matrix with eigenvalue 1, which is subsequently normalized such that $\sum_i \pi_i = 1$. When the transition matrix is fully connected, the largest eigenvalue is 1, and it is not degenerate. This also means that the stationary distribution is unique. Stationary probabilities can be calculated by the mm_transitionmatrixAnalysis program with the -stationarydistribution option.

Many scientists prefer to characterize the stability of states via their free energies. The free energies can be directly calculated (via option -freeenergies) from the stationary probabilities:

$$F_i = -k_B T \ln \frac{\pi_i}{\max_j \pi_j} \tag{13}$$

as a measure of the free energy differences with respect to the most stable microstate. See Figure 5a for an illustration of the relationship between stationary distribution and free energy in a one-dimensional model potential.

Spectral Analysis. The EMMA command mm_transitionmatrixAnalysis can be used to decompose the transition matrix into eigenvalues and eigenvectors. Such a spectral analysis is useful in order to gain information about the slowest conformational changes of a molecule. The slowest conformational changes correspond to the slowest dynamical processes of the Markov model. This correspondence can be used to identify metastable conformations and to estimate the time scales of conformational changes.[39−41,43]

In the following, we will outline the importance of the spectral properties of the Markov model. Consider the propagation of a probability vector $\mathbf{p}_0$ on the microstate space using the transition matrix $\mathbf{T}(\tau)$:

$$\mathbf{p}_{k\tau}^T = \mathbf{p}_0^T \mathbf{T}^k(\tau) \tag{14}$$

In the following, we will show that we can express this propagation as a sum of exponentially decaying processes. As a consequence of reversibility, it is possible to diagonalize the transition matrix using a complete set of left and right eigenvectors, $\{\mathbf{r}_i\}_{i=1}^n$ and $\{\mathbf{l}_i\}_{i=1}^n$. Recall that the stationary distribution p is the left eigenvector corresponding to the unique eigenvalue $\lambda_1 = 1$:

$$\pi = \mathbf{l}_1 \tag{15}$$

The left and right eigenvectors are normalized such that $\langle \mathbf{l}_i, \mathbf{r}_j \rangle = \delta_{ij}$. Here, $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$ denotes the standard scalar product. Another consequence of the detailed balance condition is that left and right eigenvectors are related as follows:

$$\mathbf{l}_{i,k} = \mathbf{r}_{i,k} \pi_k \tag{16}$$

Then, $\mathbf{T}(\tau)$ can be decomposed as

$$\mathbf{T}(\tau) = \sum_{i=1}^n \lambda_i \mathbf{r}_i \mathbf{l}_i^{\mathbf{T}} \tag{17}$$

so that

$$\mathbf{T}(\tau)\mathbf{r}_i = \lambda_i \mathbf{r}_i \tag{18}$$

$$\mathbf{l}_i^T \mathbf{T}(\tau) = \lambda_i \mathbf{l}_i^T \tag{19}$$

Now, we can write the propagation of $\mathbf{p}_0$ as

$$\mathbf{p}_{k\tau}^T = \mathbf{p}_0 \mathbf{T}^k(\tau) = \sum_{i=1}^n \lambda_i^k \langle \mathbf{p}_0, \mathbf{r}_i \rangle \mathbf{l}_i^T \tag{20}$$

Using the fact that the first eigenvalue $\lambda_1 = 1$ and that $|\lambda_i| = |\lambda_i(\tau)| = e^{-(\tau/(t_i))}$, we can finally conclude that

$$\mathbf{p}_{k\tau}^T = \langle \mathbf{p}_0, \mathbf{r}_i \rangle \mathbf{p} + \sum_{i=2}^n e^{-k\tau/t_i} \langle \mathbf{p}_0, \mathbf{r}_i \rangle \mathbf{l}_i^T \tag{21}$$

In other words, the relaxation of any initial distribution to the equilibrium distribution can be understood in terms of a sum of exponentials, each decaying with a rate governed by the eigenvalue λ_i or the associated implied time scale, which was previously summarized by eq 2.

The structural rearrangement associated with this time scale is expressed by the sign structure of the corresponding left eigenvector $\mathbf{l}_i$ or right eigenvector $\mathbf{r}_i$ (both the left and right eigenvector carry the same qualitative information; they are just differently weighted—with the use of p, eq 21 can be written in terms of left eigenvectors only or right eigenvectors only). Figure 5 shows the diffusion on a one-dimensional energy surface as an example. The yellow eigenvector corresponds to the largest implied time scale, i.e., the slowest relaxation process, and indicates that the corresponding transition occurs between energy basins (A + B) and (C + D), i.e., across the largest energy barrier. The green and blue eigenvectors further subdivide the (A + B) and (C + D) basins.

Since the relaxation time scales are observable in kinetic experiments, the eigenvalue decomposition (spectral decomposition) of a Markov model transition matrix provides direct insight into these structural changes. Many protein folding studies[40,41,66] analyze the second eigenvector because one usually assumes that folding is the process governing the slowest relaxation time scale.[67] This assumption may be wrong in certain cases as demonstrated in refs 39 and 68. In ref 39, the first five eigenvectors of a Markov state model were analyzed in order to understand the dominant conformation dynamics of glycine−serine peptides of different lengths.

Eigenvectors and eigenvalues can be directly calculated for a given transition matrix using the command mm_transitionmatrixAnalysis. The implied time scales can be obtained from the eigenvalues using eq 2.

Transition Path Theory (TPT). Transition path theory allows us to analyze the essential statistical features of the reactive transitions between two chosen subsets A and B.[41,69−71] This includes the analysis of transition pathways

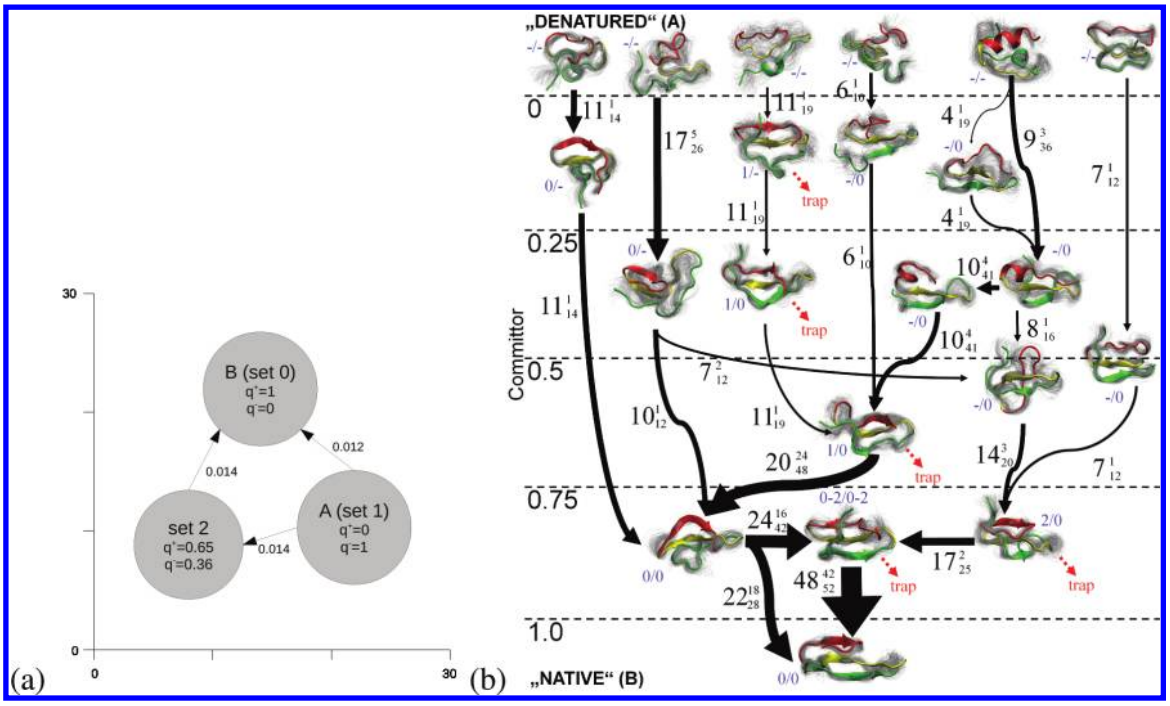

Figure 6. (a) Flux of the transition from the least-populated to the most-populated metastable state of the two-dimensional diffusion example from the EMMA tutorial. (b) Flux of the folding transitions between the metastable states of the PinWW protein. Figure adapted with permission from ref 41. Copyright 2009, National Academy of Sciences.

between A and B, their relative probabilities, the total $A \rightarrow B$ flux, and the $A \rightarrow B$ rate. The first application of transition path theory to the analysis of a molecular system has been carried out in an investigation of the ensemble of protein folding pathways in ref 41 and protein−ligand binding pathways in ref 8. Since then, several applications have been using TPT to tackle questions in protein folding and conformation dynamics.[9,40,43] The results of transition path theory can be calculated using the mm_tpt program. Transition path properties can be either calculated for the full set of microstates or for a coarse-graining of microstate space such as the one achieved by PCCA using the -coarsesets option.

The essential object needed to calculate statistics pertaining to the $A \rightarrow B$ reaction is the committor probability, also called splitting probability or probability of folding.[72−79] Let A and B be two disjoint sets of microstates. The forward commitor q_i^+ is a function on microstate space assigning to each state i the probability for the Markov process to visit set B before set A when starting in i. By definition, $q_i^+ = 0$ for $i \in A$ and $q_i^+ = 1$ for $i \in B$, while $q_i^+ \in [0,1]$ for all other states. Many standard reaction coordinates which are defined a priori without knowledge about the dynamics exhibit the problem of concealing relevant dynamics of the system. The commitor in contrast defines a dynamical reaction coordinate circumventing this problem.[72,76,79] Often the isocommitor surface $q_i^+ = 0.5$ is of special interest. The isocommitor interface has been interpreted as the transition state ensemble in protein folding theory.[80]

We also need the backward committor probability, q_i^-, which is the probability that the process came from A last rather than from B. In the case where dynamics are reversible, i.e., the transition matrix fulfills detailed balance, the backward and forward commitors are related in the following way: $q_i^- = 1 - q_i^+$.

A property that is of special interest for applications is the reactive flux. The reactive flux or current f_{ij} can be calculated using TPT in the following way:[71,41]

$$f_{ij} = \begin{cases} \pi_i q_i^- T_{ij} q_j^+ & i \neq j \\ 0 & i = j \end{cases} \tag{22}$$

The reactive flux counts the number of transition pathways per time going from state i to state j. f_{ij} does include transitions from unproductive recrossings and transitions into and out of dead ends. Since it is often desirable to only analyze path probabilities without recrossings or dead ends, one calculates the net flux via

$$f_{ij}^+ = \max(q_i^+ - q_i^-) \tag{23}$$

in the special case of reversible dynamics (detailed balance transition matrix); the net flux is given by[81]

$$f_{ij}^+ = \pi_i (q_j^+ - q_i^+) T_{ij} \tag{24}$$

when $q_i^+ \leq q_j^+$. The net flux is available via the option -onetflux.

Since f_{ij}^+ is defined for pairs of microstates, TPT will produce a possibly very large network consisting of many small fluxes. It is often useful to investigate fluxes on a coarse grained microstate space. The coarse-grained flux and netflux, as well as the coarse-grained forward and backward commitors are available applying the coarse prefix to the commands for the calculation of microstate commitors. Fluxes can be trivially added for any partition. A partition of special interest is the one into metastable states that has been generated by PCCA (see eq 7). As examples, we show the net fluxes among the four metastable states of MR121-GSGS-W and the folding flux network of PinWW (taken from ref 41) in Figure 6.

q_{ij}^{+} defines a flux-conserving network flow out of A and into B that can be decomposed into a set of $A \rightarrow B$ reaction pathways along with their probabilities.[41,69,71]

Finally, the total flux of the $A \rightarrow B$ reaction is given by

$$F_{AB} = \sum_{i \in A, j \notin A} f_{ij} = \sum_{i \in A, j \notin A} f_{ij}^{+} \tag{25}$$

and the $A \rightarrow B$ reaction rate[41] is given by:

$$k_{AB} = \frac{F_{AB}}{\sum_i \pi_i q_i^{-}} \tag{26}$$

Expectation Values, Correlation Functions, Fingerprints. The EMMA program mm_observables is a useful tool used to analyze MSMs in a way that allows comparison to experimental measurements. Let **a** be an observable in microstate space with scalar values a_i on each microstate i. In order to model experimental observations using such an observable, we have to provide this observable definition for computations with EMMA. Such an observable with scalar values is stored as a single column ascii file containing one measurement value a_i in each row. Scalar observables may be any functions taking real values on the state of microstates. They can model a variety of observations, e.g., intensities of fluorescence signals, FRET efficiencies, particle distances, etc.

Many experiments measure *ensemble averages*. Given a transition matrix $\mathbf{T}(\tau)$ with associated stationary distribution π and observable vector **a**, the ensemble average can be calculated with the -expectation command. It is simply estimated by

$$\mathbb{E}[a] = \sum_{i=1}^{n} \pi_i a_i \tag{27}$$

The most interesting features of mm_observables however allow dynamical observables to be calculated, such as perturbation−relaxation and correlation curves, as they can also be measured in kinetic experiments. Importantly, mm_observables can help to interpret these curves in terms of *dynamical fingerprints*, which can be dissected into dynamical features that are associated with individual relaxation time scales and structural rearrangement processes.[39,68] Here, we differentiate between two types of kinetic experiments: perturbation and correlation experiments.

In *perturbation experiments*, the ensemble average of an observable is tracked over time while the ensemble relaxes from some perturbed or triggered initial state at time $t = 0$ toward its stationary distribution. The initial trigger may consist of, e.g., a jump in temperature[82,83] or pressure,[84] a change in the chemical environment,[85] or a photoflash.[86−88] Such time-dependent ensemble averages can be calculated with the commands -perturbation and -relax *via*

$$\mathbb{E}[\mathbf{a}(k\tau)]_{\mathbf{p}_0} = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{0,i} T_{ij}(k\tau) a_j \tag{28}$$

A special perturbation experiment is the temperature-jump experiment where an ensemble is prepared at temperature T_1 at $t < 0$ and is then suddenly changed to temperature T_2 at $t = 0$. The system is kept at T_2 and relaxes from its old distribution $\mathbf{p}_0 = p(T_1)$ to its new stationary distribution $\pi(T_2)$. In cases where simulations have been conducted at both temperatures T_1 and T_2, the stationary distribution at T_1 can be straightforwardly obtained as an initial distribution to a perturbation experiment using the transition matrix at T_2, and the result can be calculated with the -perturbation command.

A very common type of kinetic experiment is *correlation experiments*. Correlation experiments may be realized either through scattering techniques such as inelastic neutron scattering[89] or *via* low concentration or single molecule experiments accumulating auto- or cross-correlations of fluctuations, e.g., correlation spectroscopy of the fluorescence intensity[90−94] or Förster resonance energy transfer efficiency.[95,96] The -autocorrelation command calculates equilibrium autocorrelations of observables **a** via

$$\mathbb{E}[\mathbf{a}(0)\ \mathbf{a}(k\tau)]_{\mathbf{p}} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \pi_i T_{ij}(k\tau) a_j \tag{29}$$

and the -crosscorrelation command calculates the cross-correlation between two observables **a** and **b** via

$$\mathbb{E}[\mathbf{a}(0)\ \mathbf{b}(k\tau)]_{\pi} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \pi_i T_{ij}(k\tau) b_j \tag{30}$$

Instead of directly printing the perturbation−relaxation or correlation curve via -relax, one can output the dynamical fingerprint of a perturbation or correlation experiment via the -fingerprint command. As explained in detail in refs 39 and 68, the long-time scale part of eqs 27, 28, and 29 can each be written in the form

$$y(k\tau) = \sum_{i=1}^{m} \gamma_i \exp\left(-\frac{k\tau}{t_i^*}\right) \tag{31}$$

where $y(k\tau)$ is the dynamical observable (expectation or correlation), t_i^* is the ith implied time scale, and γ_i is an amplitude that depends on the specific experiment conducted. The amplitudes γ_i are derived in refs 39 and 68 and can be calculated from scalar products of initial or stationary probability distributions, properly normalized left or right eigenvectors l_i and r_i and observable vectors **a** and **b**:

$$\gamma_i^{\text{perturbation}} = \langle \mathbf{p}_0, \mathbf{r}_i \rangle \langle \mathbf{a}, \mathbf{l}_i \rangle \tag{32}$$

$$\gamma_i^{\text{autocorrelation}} = \langle \mathbf{a}, \mathbf{l}_i \rangle^2 \tag{33}$$

$$\gamma_i^{\text{crosscorrelation}} = \langle \mathbf{a}, \mathbf{l}_i \rangle \langle \mathbf{b}, \mathbf{l}_i \rangle \tag{34}$$

The command -fingerprint outputs the amplitudes γ_i and time scales t_i^* for all spectral components with positive eigenvalues. This result can then be plotted into a dynamical fingerprint plot that can be directly compared to the fingerprint calculated from the experimental measurement. When modeling and sampling errors are small, one can match peaks between experimental and simulated fingerprints. On the basis of this match, one can assign structural processes to experimentally detectable relaxation time scales (see Figure 7) or even propose new experiments that optimally amplify low-amplitude features.[39,68] Note that for comparing the dynamical fingerprint from the MSM to its experimental counterpart, the experimental relaxation curve must somehow be transformed into a spectral density. One way to do this, suggested in refs 39 and 68,

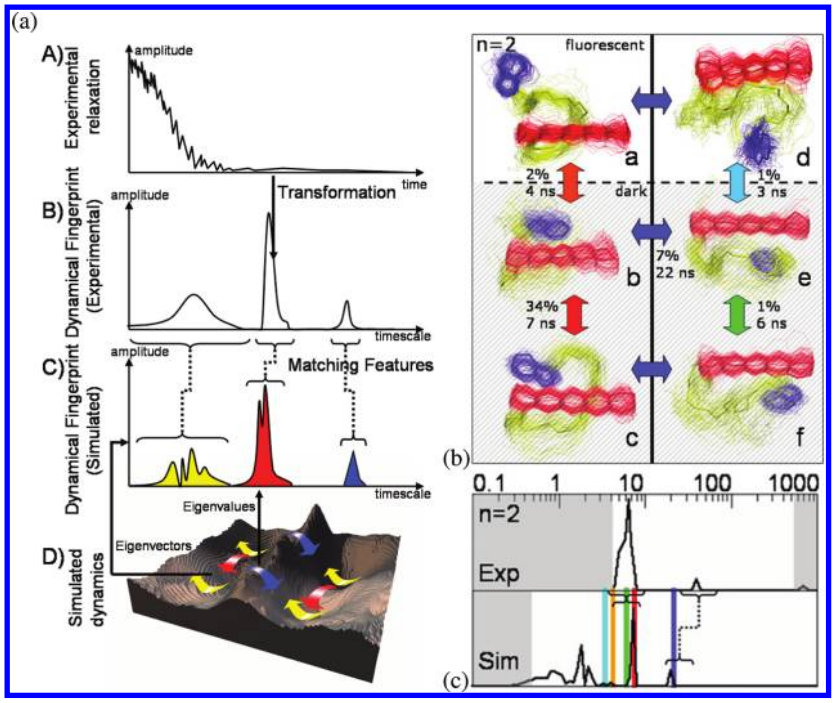

Figure 7. (a) Experimental relaxation profiles (A) can be transformed into dynamical fingerprints (B), which represent the time scales and amplitudes of the relaxation processes in the data without having to predetermine a particular model or number of processes. On the theoretical side, the dynamics on an energy landscape (D) also generates dynamical fingerprints (C), but here each feature can be uniquely assigned to a particular transition or diffusion process on the landscape. If the simulation model is a sufficiently accurate model of the experimental system, structural processes can be assigned to the experimental data by matching features in the experimental and theoretical fingerprint. (b) Partition of the conformation space for the peptide MR121-GSGS-W into the six most stable metastable (PCCA) states and the associated five slowest relaxations (as indicated by the transition matrix eigenvectors). Representative structures are shown in cartoon representation with flexibility indicated by the overlay of line structures. The fluorescent states are shown bright; the dark states are shaded. The slow relaxation processes are indicated by colored arrows. (c) Dynamical fingerprint for a fluorescence quenching experiment of MR121-GSGS-W extracted from single molecule FCS data directly (top), and from the MD simulation Markov model (bottom). Features in the experiment and simulation that can be qualitatively associated with each other are linked with dashed lines. The mean positions and amplitudes of the five slowest relaxations are marked in colors and correspond to the structural transitions shown on the left. Regions that are unreliable due to measurement or analysis artifacts are grayed out. Figures reprinted with permission from ref 39. Copyright 2011, National Academy of Sciences.

is implemented in the SciMex package available at simtk.org/home/scimex.

■ CONCLUSIONS

Markov state models are used by an increasingly wide community to model and analyze molecular dynamics data. EMMA provides a number of tools for Markov model construction of molecular kinetics. Molecular dynamics data can be partitioned into microstates using different clustering techniques. Different methods are available to estimate the Markov model transition matrix and to validate the model by testing its ability to reproduce the long-time dynamics of the given trajectory data. EMMA provides a number of analysis tools enabling the calculation of stationary probabilities, free energies, relaxation time scales, transition pathways, and observables measured in kinetic experiments.

In the future, we plan to support MSM software developers by releasing API documentation that will permit direct access to the functionality of the EMMA classes in Java. Furthermore, we plan on extending EMMA's functionality to match the demand for improved tools and algorithms for the application of Markov models to problems in molecular sciences.

■ APPENDIX: EMMA FILE TYPES

Here, a concise definition of the file formats used in EMMA is given. Input and output files are assumed to be human readable ascii files with the exception of trajectory files and cluster centers. Trajectory data can be either in ASCII format (see below), Gromacs compressed binary format (xtc), or Charmm/NAMD binary format (dcd). Cluster centers are stored either as ASCII (when the input data were ASCII) or dcd (when the input data were dcd or xtc). Subsequently, the file formats are described in detail.

ASCII Trajectory and Cluster Center Files

Given is a time-discrete trajectory $\mathbf{z}(t_j)$ with n frames, where $j \in 0, ..., n-1$ denotes the frame index of the trajectory. Each frame $\mathbf{z}(t_j)$ is of dimension d, thus $\mathbf{z}(t_j) \in \mathbb{R}^d$. The trajectory is stored in row-wise orientation; each line of the file contains one frame of the trajectory:

<x_1_t_0:double> ... <x_d_t_0:double>
<x_1_t_1:double> ... <x_d_t_1:double>
...
<x_1_t_n−1:double> ... <x_d_t_n−1:double>

It is also possible to include a time column as a first column. In this case, the option -timecolumn must be used in the

mm_cluster and mm_assign commands in order to avoid interpreting the time column as coordinate:

```
<t_0:double>     <x_1_t_0:double>     ... <x_d_t_0:double>
<t_1:double>     <x_1_t_1:double>     ... <x_d_t_1:double>
...
<t_n−1:double> <x_1_t_n−1:double> ... <x_d_t_n−1:double>
```

Discrete-State Trajectories

Given is a time-discrete microstate trajectory $s(t_j)$ with n frames, where $j \in 0, ..., n-1$ denotes the frame index of the trajectory. Each frame $s(t_j)$ denotes a single microstate. The trajectory is stored in row-wise orientation; each line of the file contains one frame of the trajectory:

```
<s_0:int>
<s_1:int>
...
<s_n−1:int>
```

Matrix Files

Dense Matrix Format. A matrix $M \in \mathbb{R}^{r\times c}$ is stored in dense format as defined below.

```
DENSE <r:int> <c:int>
  <m_0_0:double>    <m_0_1:doubl ... <m_0_c−1:do
                       e>              uble>
  <m_1_0:double>    <m_1_1:doubl ... <m_1_c−1:do
                       e>              uble>
  ...
  <m_r−1_0:double> <m_r−1_1:do  ... <m_r−1_c−1:
                       uble>           double>
```

Sparse Matrix Format. A matrix $M \in \mathbb{R}^{r\times c}$ is stored in a sparse format as defined below. The file contains only entries $m_{i,j}$, for which is $m_{i,j} \neq 0.0$. Such an entry is defined as i, j, and $m_{i,j}$.

```
SPARSE <r:int> <c:int>
  <row:int>  <col:int>  ... <m_row:double>
  <row:int>  <col:int>  ... <m_row:double>
  ...
  (lines of the above format for every entry not zero)
```

Vector Files

All vectors **v**, e.g., a stationary distribution vector or a vector of observables for mm_observables, are written column-wise and in dense format. A vector $\mathbf{v} \in \mathbb{R}^d$ is stored as defined below. Currently, there is no header. The i line represents the ith entry v_i of **v**.

```
<v_0:double>
<v_1:double>
      ...
<v_d:double>
```

State Selection Files

The state selection file contains a set of microstates. A file of this type is generated by the command mm_connectivity and contains, if generated by the above command, these microstates, which belong to the largest connected component.

This file can be used for input, where the option -restrictToStates is available, thus for the command mm_time scales, mm_transitionmatrixEstimation and mm_chapman.

Each row of the file contains one state.

```
<state:int>
<state:int>
    ...
```

Set Definition Format

A set definition file (provided as an output by mm_pcca and required as an input to mm_chapman and mm_tpt) contains a list of sets of microstates. The ith row of the file corresponds to the set $S_i \in S$ and contains a whitespace-separated list of microstates (arbitrarily many) belonging to that specific set.

```
<state_1_of_set_0:int>  <state_2_of_set_0:int>
<state_1_of_set_1:int>                ...
          ...
```

The following is an example, which defines three sets $S_0 = \{2, 3, 5, 6\}$, $S_1 = \{1, 4\}$, and $S_2 = \{0, 7, 8, 9\}$:

```
2 3 5 6
1 4
0 7 8 9
```

AUTHOR INFORMATION

Corresponding Author

*E-mail: frank.noe@fu-berlin.de.

Notes

The authors declare no competing financial interest.

REFERENCES

(1) Hummer, G.; Rasaiah, J. C.; Noworyta, J. P. *Nature* **2001**, *414*, 188−190.
(2) Klimov, D. K.; Thirumalai, D. *Chem. Phys.* **2004**, *307*, 251−258.
(3) Dill, K. A.; Chan, H. S. *Nat. Struct. Mol. Biol.* **1997**, *4*, 10−19.
(4) Wolynes, P. G.; Onuchic, J. N.; Thirumalai, D. *Science* **1995**, *267*, 1619−6616.
(5) Bowman, G. R.; Beauchamp, K. A.; Boxer, G.; Pande, V. S. *J. Chem. Phys.* **2009**, *131*, 124101.
(6) Noé, F.; Schütte, C.; Vanden-Eijnden, E.; Reich, L.; Weikl, T. R. *Proc. Natl. Acad. Sci. U.S.A.* **2009**, *106*, 19011−19016.
(7) Kremer, K.; Grest, G. S. *J. Chem. Phys.* **1990**, *92*, 5057−5086.
(8) Held, M.; Metzner, P.; Noé, F. *Biophys. J.* **2011**, *100*, 701−710.
(9) Buch, I.; Giorgino, T.; De Fabritiis, G. *Proc. Natl. Acad. Sci. U. S. A.* **2011**, *108*, 10184−10189.
(10) Spaar, A.; Dammer, C.; Gabdoulline, R. R.; Wade, R. C.; Helms, V. *Biophys. J.* **2006**, *90*, 1913−1924.
(11) Gabdoulline, R. R.; Wade, R. C. *J. Mol. Biol.* **2001**, *306*, 1139−1155.
(12) Fischer, S.; Windshuegel, B.; Horak, D.; Holmes, K. C.; Smith, J. C. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6873−6878.
(13) Noé, F.; Krachtus, D.; Smith, J. C.; Fischer, S. *J. Chem. Theory Comput.* **2006**, *2*, 840−857.
(14) Pan, A. C.; Roux, B. *J. Chem. Phys.* **2008**, *129*, 064107.
(15) Krivov, S. V.; Karplus, M. *Proc. Natl. Acad. Sci. U. S. A.* **2004**, *101*, 14766−14770.
(16) Noé, F.; Fischer, S. *Curr. Opin. Struct. Biol.* **2008**, *18*, 154−162.
(17) Muff, S.; Caflisch, A. *Proteins* **2007**, *70*, 1185−1195.
(18) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. *J. Chem. Theory Comput.* **2008**, *4*, 435−447.

(19) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(20) Mei, C.; Sun, Y.; Zheng, G.; Bohm, E. J.; Kale, L. V.; Phillips, J. C.; Harrison, C. *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*; SC '11; ACM: New York, 2011; pp 61:1−61:11.
(21) Stone, J. E.; Phillips, J. C.; Freddolino, P. L.; Hardy, D. J.; Trabuco, L. G.; Schulten, K. *J. Comput. Chem.* **2007**, *28*, 2618−2640.
(22) van Meel, J. A.; Arnold, A.; Frenkel, D.; Portegies; Belleman, R. G. *Mol. Simul.* **2008**, *34*, 259−266.
(23) Eastman, P.; Pande, V. S. *J. Comput. Chem.* **2010**, *31*, 1268−1272.
(24) Harvey, M. J.; De Fabritiis, G. *J. Chem. Theory Comput.* **2009**, *5*, 2371−2377.
(25) Shirts, M. R.; Pande, V. S. *Science* **2000**, *290*, 1903−1904.
(26) Shaw, D. E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Eastwood, M. P.; Bank, J. A.; Jumper, J. M.; Salmon, J. K.; Shan, Y.; Wriggers, W. *Science* **2010**, *330*, 341−346.
(27) Pronk, S.; Larsson, P.; Pouya, I.; Bowman, G. R.; Haque, I. S.; Beauchamp, K.; Hess, B.; Pande, V. S.; Kasson, P. M.; Lindahl, E. *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*; SC '11; ACM: New York, 2011; pp 60:1−60:10.
(28) Wales, D. J. *Energy Landscapes*; Cambridge University Press: Cambridge, U. K., 2003.
(29) Karpen, M. E.; Tobias, D. J.; Brooks, C. L. *Biochemistry* **1993**, *32*, 412−420.
(30) Hubner, I. A.; Deeds, E. J.; Shakhnovich, E. I. *Proc. Natl. Acad. Sci. U. S. A.* **2006**, *103*, 17747−17752.
(31) Weber, M. *Improved Perron Cluster Analysis*; ZIB Report 2003/4, ZIB: Berlin, Germany, 2003.
(32) Buchete, N. V.; Hummer, G. *J. Phys. Chem. B* **2008**, *112*, 6057−6069.
(33) Rao, F.; Caflisch, A. *J. Mol. Biol.* **2004**, *342*, 299−306.
(34) de Groot, B.; Daura, X.; Mark, A.; Grubmüller, H. *J. Mol. Biol.* **2001**, *301*, 299−313.
(35) Schultheis, V.; Hirschberger, T.; Carstens, H.; Tavan, P. *J. Chem. Theory Comput.* **2005**, *1*, 515−526.
(36) Prinz, J.-H.; Wu, H.; Sarich, M.; Keller, B.; Fischbach, M.; Held, M.; Chodera, J. D.; Schütte, C.; Noé, F. *J. Chem. Phys.* **2011**, *134*, 174105.
(37) Noé, F.; Horenko, I.; Schütte, C.; Smith, J. C. *J. Chem. Phys.* **2007**, *126*, 155102.
(38) Schütte, C.; Fischer, A.; Huisinga, W.; Deuflhard, P. *J. Comput. Phys.* **1999**, *151*, 146−168.
(39) Noé, F.; Doose, S.; Daidone, I.; Löllmann, M.; ChoderaJ, J. D.; Sauer, M.; Smith, J. C. *Proc. Natl. Acad. Sci. U. S. A.* **2011**, *108*, 4822−4827.
(40) Voelz, V. A.; Bowman, G. R.; Beauchamp, K.; Pande, V. S. *J. Am. Chem. Soc.* **2010**, *132*, 1526−1528.
(41) Noé, F.; Schütte, C.; Vanden-Eijnden, E.; Reich, L.; Weikl, T. R. *Proc. Natl. Acad. Sci. U. S. A.* **2009**, *106*, 19011−19016.
(42) Sezer, D.; Freed, J. H.; Roux, B. *J. Phys. Chem. B* **2008**, *112*, 11014−11027.
(43) Bowman, G. R.; Voelz, V. A.; Pande, V. S. *J. Am. Chem. Soc.* **2011**, *133*, 664−667.
(44) Zhuang, W.; Cui, R. Z.; Silva, D.-A.; Huang, X. *J. Phys. Chem. B* **2011**, *115*, 5415−5424.
(45) Beauchamp, K.; Bowman, G.; Lane, T.; Maibaum, L.; Haque, I.; Pande, V. *J. Chem. Theory Comput.* **2011**, *7*, 3412.
(46) Theobald, D. L. *Acta Crystallogr.* **2005**, *A61*, 478−480.
(47) Chodera, J. D.; Singhal, N.; Pande, V. S.; Dill, K. A.; Swope, W. C. *J. Chem. Phys.* **2007**, *126*, 155101.
(48) Altis, A.; Nguyen, P. H.; Hegger, R.; Stock, G. *J. Chem. Phys.* **2007**, *126*, 244111.
(49) Dasgupta, S.; Long, P. *J. Comput. Syst. Sci.* **2005**, *70*, 555−569.
(50) Tarjan, R. E. *SIAM J. Comput.* **1972**, *1*, 146−160.
(51) Sarich, M.; Noé, F.; Schütte, C. *SIAM Multiscale Model. Simul.* **2010**, *8*, 1154−1177.
(52) Swope, W. C.; Pitera, J. W.; Suits, F.; Pitman, M.; Eleftheriou, M.; Fitch, B. G.; Germain, R. S.; Rayshubskiy, A.; Ward, T. C.; Zhestkov, Y. *J. Phys. Chem. B* **2004**, *108*, 6582−6594.
(53) Noé, F. *J. Chem. Phys.* **2008**, *128*, 244103.
(54) Prinz, J.-H.; Held, M.; Smith, J. C.; Noé, F. *Multiscale Model. Simul.* **2011**, *9*, 545.
(55) Huang, D.; Caflisch, A. *PLoS Comput Biol* **2011**, *7*, e1002002.
(56) Bowman, G. R.; Ensign, D. L.; Pande, V. S. *J. Chem. Theory Comput.* **2010**, *6*, 787−794.
(57) Röblitz, S.; Weber, M. *J. Chem. Phys.* **2007**, *126*, 024103.
(58) Deuflhard, P.; Weber, M. *ZIB Report* **2003**, 03−09.
(59) Park, S.; Pande, V. S. *J. Chem. Phys.* **2006**, *124*, 054118.
(60) Swope, W. C.; Pitera, J. W.; Suits, F. *J. Phys. Chem. B* **2004**, *108*, 6571−6581.
(61) Buchete, N.-V.; Hummer, G. *J. Phys. Chem. B* **2008**, *112*, 6057−6069.
(62) Singhal, N.; Pande, V. S. *J. Chem. Phys.* **2005**, *123*, 204909.
(63) Hinrichs, N. S.; Pande, V. S. *J. Chem. Phys.* **2007**, *126*, 244101.
(64) Noé, F. *J. Chem. Phys.* **2008**, *128*, 244103.
(65) Chodera, J. D.; Noé, F. *J. Chem. Phys.* **2010**, *133*, 105102.
(66) Bowman, G. R.; Beauchamp, K. A.; Boxer, G.; Pande, V. S. *J. Chem. Phys.* **2009**, *131*, 124101.
(67) Jäger, M.; Nguyen, H.; Crane, J. C.; Kelly, J. W.; Gruebele, M. *J. Mol. Biol.* **2001**, *311*, 373−393.
(68) Keller, B.; Prinz, J.-H.; Noé, F. *Chem. Phys.* **2012**, *396*, 92−107.
(69) Vanden-Eijnden, E. In *Computer Simulations in Condensed Matter: From Materials to Chemical Biology*, 1st ed.; Ferrario, M., Binder, K., Ciccotti, G., Eds.; Springer: Berlin/Heidelberg, 2006; Vol. 1 (Lecture Notes in Physics), pp 453−493.
(70) Metzner, P.; Schütte, C.; Vanden-Eijnden, E. *J. Chem. Phys.* **2006**, *125*, 084110.
(71) Metzner, P.; Schütte, C.; Vanden-Eijnden, E. *Multiscale Model. Simul.* **2009**, *7*, 1192−1219.
(72) E, W.; Ren, W.; Vanden-Eijnden, E. *Chem. Phys. Lett.* **2005**, *413*, 242−247.
(73) Dellago, C.; Bolhuis, P. G.; Geissler, P. L. *Adv. Chem. Phys.* **2002**, *123*, 1.
(74) Bolhuis, P. G.; Chandler, D.; Dellago, C.; Geissler, P. L. *Annu. Rev. Phys. Chem.* **2002**, *53*, 291.
(75) Maragliano, L.; Fischer, A.; Vanden-Eijnden, E.; Ciccotti, G. *J. Chem. Phys.* **2006**, *125*, 24106.
(76) Best, R. B.; Hummer, G. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6732−6737.
(77) Du, R.; Pande, V. S.; Grosberg, A. Y.; Tanaka, T.; Shakhnovich, E. I. *J. Chem. Phys.* **1998**, *108*, 334.
(78) Hummer, G. *J. Chem. Phys.* **2004**, *120*, 516−523.
(79) Ma, A.; Dinner, A. R. *J. Phys. Chem. B* **2005**, *109*, 6769−6779.
(80) Pande, V. *Curr. Opin. Struct. Biol.* **1998**, *8*, 68−79.
(81) Berezhkovskii, A.; Hummer, G.; Szabo, A. *J. Chem. Phys.* **2009**, 130.
(82) Jäger, M.; Zhang, Y.; Bieschke, J.; Nguyen, H.; Dendle, M.; Bowman, M. E.; Noel, J. P.; Gruebele, M.; Kelly, J. W. *Proc. Natl. Acad. Sci. U. S. A.* **2006**, *103*, 10648−10653.
(83) Sadqi, M.; Lapidus, L. J.; Munoz, V. *Proc. Natl. Acad. Sci. U. S. A.* **2003**, *100*, 12117−12122.
(84) Dumont, C.; Emilsson, T.; Gruebele, M. *Nat. Methods* **2009**, *6*, 515−519.
(85) Chan, C.-K.; Hu, Y.; Takahashi, S.; Rousseau, D. L.; Eaton, W. A.; Hofrichter, J. *Proc. Natl. Acad. Sci. U. S. A.* **1997**, *94*, 1779−1784.
(86) Volkmer, A. *Biophys. J.* **2000**, *78*, 1589−1598.
(87) Schlichting, I.; Almo, S. C.; Rapp, G.; Wilson, K.; Petratos, K.; Lentfer, A.; Wittinghofer, A.; Kabsch, W.; Pai, E. F.; Petsko, G. A.; Goody, R. S. *Nature* **1990**, *345*, 309−315.
(88) Buck, J.; Fürtig, B.; Noeske, J.; Wöhnert, J.; Schwalbe, H. *Proc. Natl. Acad. Sci. U. S. A.* **2007**, *104*, 15699−15704.
(89) Doster, W.; Cusack, S.; Petry, W. *Nature* **1989**, *337*, 754−756.

(90) Lapidus, L. J.; Eaton, W. A.; Hofrichter, J. *Proc. Natl. Acad. Sci. U. S. A.* **2000**, *97*, 7220−7225.
(91) Neuweiler, H.; Löllmann, M.; Doose, S.; Sauer, M. *J. Mol. Biol.* **2007**, *365*, 856−869.
(92) Michalet, X.; Weiss, S.; Jäger, M. *Chem. Rev.* **2006**, *106*, 1785−1813.
(93) Tinnefeld, P.; Sauer, M. *Angew. Chem., Int. Ed.* **2005**, *44*, 2642−2671.
(94) Hudgins, R. R.; Huang, F.; Gramlich, G.; Nau, W. M. *J. Am. Chem. Soc.* **2002**, *124*, 556−564.
(95) Kim, H. D.; Nienhaus, G. U.; Ha, T.; Orr, J. W.; Williamson, J. R.; Chu, S. *Proc. Natl. Acad. Sci. U. S. A.* **2002**, *99*, 4284−4289.
(96) Nettels, D.; Hoffmann, A.; Schuler, B. *J. Phys. Chem. B* **2008**, *112*, 6137−6146.