A Step-by-step Guide on How to Construct quasi-Markov State Models to Study Functional Conformational Changes of Biological Macromolecules

Andrew Kai-hei Yik[1], Yunrui Qiu[1], Ilona Christy Unarta[1], Siqin Cao[1], Xuhui Huang[1*]

[1]*Department of Chemistry, University of Wisconsin−Madison, Madison, Wisconsin 53706, United States*

*To whom correspondence should be addressed. Email: xhuang@chem.wisc.edu

Abstract

Conformational changes play an important role for many biomolecules to perform their functions. In recent years, Markov State Model (MSM) has become a powerful tool to investigate these functional conformational changes by predicting long time-scale dynamics from many short molecular dynamics (MD) simulations. In MSM, dynamics are modelled by a first-order master equation, in which a biomolecule undergoes Markovian transitions among conformational states at discrete time intervals, called lag time. The lag time has to be sufficiently long to build a Markovian model, but this parameter is often bound by the length of MD simulations available for estimating the frequency of interstate transitions. To address this challenge, we recently employed the generalized master equation (GME) formalism (e.g., the quasi-Markov State Model or qMSM) to encode the non-Markovian dynamics in a time-dependent memory kernel. When applied to study protein dynamics, our qMSM can be built from MD simulations that are an order-of-magnitude shorter than MSM would have required. The construction of qMSM is more complicated than that of MSMs, as time-dependent memory kernels need to be properly extracted from the MD simulation trajectories. Here, we present a step-by-step guide on how to build qMSM from MD simulation datasets, and the materials accompanying this protocol are publicly available on Github: https://github.com/ykhdrew/qMSM_tutorial. We hope this protocol is useful for researchers who want to apply qMSM and study functional conformational changes in biomolecules.

A. Introduction

Functional conformational changes refer to the dynamic transitions of a biomolecule between conformational states, which play an important role in a wide range of biological processes such as protein-protein interactions, catalysis, and cell signalling (1). While experimental techniques, such as X-ray crystallography and Cryo-EM, can provide high-resolution structures of biomolecular complexes to better understand their functions (2), these structures only capture a static metastable state of the biomolecule and provide limited information on the dynamics of functional conformational changes (3). Although other experimental methods, such as nuclear magnetic resonance spectroscopy and single-molecular fluorescence resonance energy transfer (smFRET) can be applied to monitor conformational changes in biomolecules, they only provide the dynamic information pertaining to one or a few parameters, e.g., the distance between the FRET donor and acceptor (4).

Complementing experimental techniques, Molecular Dynamics (MD) simulation is a powerful tool to model the dynamics of biomolecular complexes at atomistic resolution (2, 5). While functional conformational changes in large biomolecules, such as the translocation of RNA polymerases II (6), can span across milliseconds-to-seconds timescales (7), MD trajectories of biomolecular systems are often limited to microsecond timescales unless specialized supercomputers, such as Anton3, are used (8). To bridge this timescale gap, Markov State Models (MSM) were developed as a powerful statistical mechanics framework that extracts long-timescale kinetics from short MD simulations initiated from different parts of the free energy landscape (9-19).

In MSM, the conformational space of a biomolecule is discretized into metastable states, and each frame in an MD trajectory is assigned to a state. By counting the number of transitions from one state to another after a discrete-time interval (i.e., lag time), we can estimate the probability of the transitions between states and obtain the transition probability matrix (TPM)

(10, 11, 20). At a Markovian lag time, at which the transition probability no longer depends on the states previously visited by the system, dynamics can be modelled by propagating TPM with the first-order master equation (Eq. 1) (9-11, 20). TPM also yields equilibrium state populations and other thermodynamic properties underlying the functional conformational changes of interest. MSM has been extensively applied to elucidate the mechanisms of functional conformational changes in biological processes(17, 21-30), such as translocation in DNA repair enzyme Alkylpurine glycosylase D (AlkD) (31), backtracking (32), and translocation (6) in RNA polymerase II (Pol II).

To allow relaxation of the intrastate dynamics within the lag time, a MSM often contains a large number of states such that each state is sufficiently small (33, 34). For example, Pande and co-workers showed that they need an MSM containing 2,000 states (with a lag time of 12ns) to model the millisecond folding of the NTL9 peptide (35). Additionally, in our previous work regarding a 37-residue intrinsically disordered peptide, we showed that an MSM consisting of as many as 10,000 states was needed to build a Markovian model (36). This limitation also applies to MSMs that study functional conformational changes. In particular, our previous work on backtracking in RNA Pol II led to an MSM composed of 800 states (32). However, MSM containing hundreds of states often hinder human interpretation of the biological mechanisms underlying functional conformational changes.

To address this bottleneck in MSM, we have developed a new theoretical framework called the quasi-Markov state model (qMSM) that goes beyond the Markovian dynamics in MSMs. Through the Generalized Master Equation (GME) framework (34), qMSM encodes non-Markovian dynamics in a generally time-dependent memory kernel, whose characteristic decay timescale corresponds to the kernel lifetime (34). We showed that qMSM can accurately predict the long-timescale dynamics of several peptide systems using significantly shorter MD simulations compared to those required by a memoryless MSM (34). This, thereby, reduces the

number of states in a model. We have successfully applied qMSM to model the gate opening dynamics in *Thermus aquaticus* (Taq) RNA Polymerase (RNAP) (37) and the mRNA recognition mechanism in the human Argonaute-2 (hAgo2) protein (38).

In this protocol, we will provide a step-by-step walkthrough, with sample input and detailed commands, for the construction and analysis of qMSM. Using alanine dipeptide as the model system, this guide is building upon a recently published review on studying functional conformational changes with qMSM (33) (see Figure 1). Apart from the computation of memory kernels, we highlight another key step in our protocol – to properly identify molecular features using a machine learning algorithm (i.e., Spectral oASIS) (39). We hope this protocol can provide a hands-on guide for researchers who are interested in using qMSM to understand functional dynamics in biological systems.

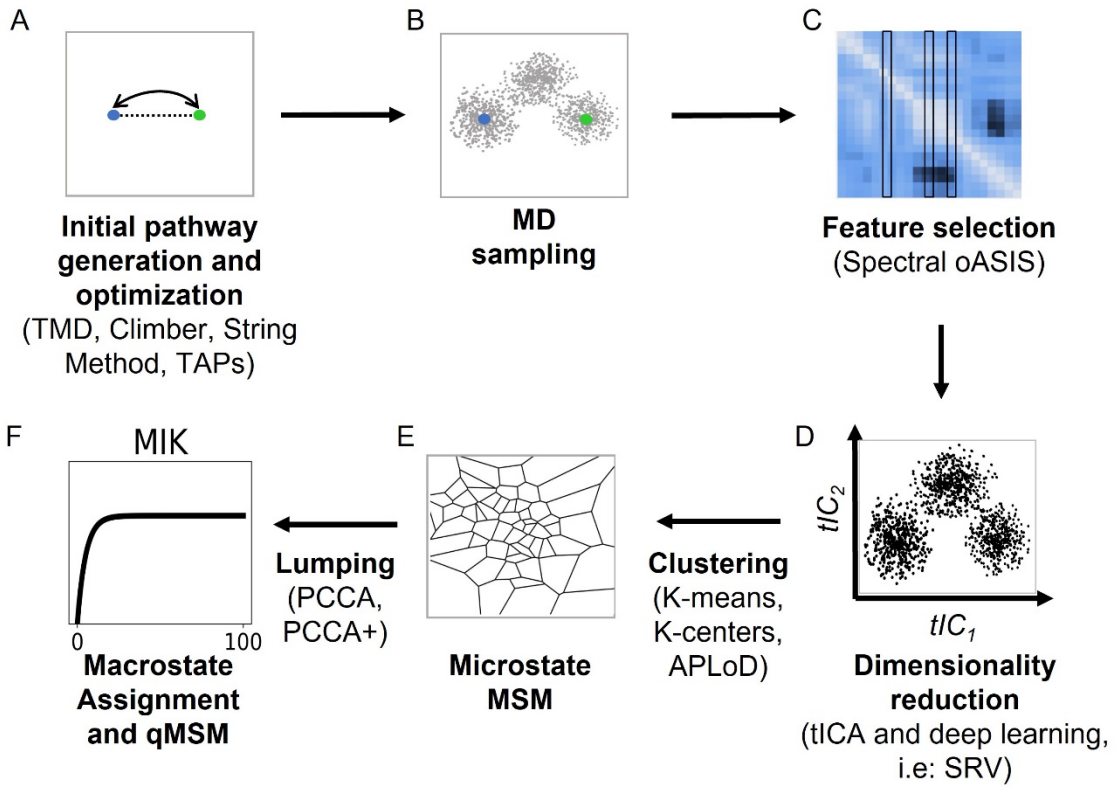

Figure. 1 Workflow for building qMSM

B. Theory

B.1 Markov State Model (MSM)

In MSM, interstate transitions at a Markovian lag time τ are described as memoryless, meaning that transitions only depend on the current state of the system and not the history of the states previously visited (10, 20, 40-42). Hence a master equation can be used to propagate long-timescale dynamics (10, 11, 40-43):

$$\boldsymbol{P}(n\tau) = [\boldsymbol{T}(\tau)]^n \boldsymbol{P}(0) \qquad \text{(Eq. 1)}$$

, where $\boldsymbol{P}(n\tau)$ is a vector of state population at the *n*-th lag time $n\tau$ and $\boldsymbol{T}(\tau)$ is the transition probability matrix. We can also calculate the implied timescale $t_i(\tau)$ using the following equation (10, 11, 40-43):

$$\boldsymbol{t}_i(\tau) = -\frac{\tau}{\log \lambda_i(\tau)} \qquad \text{(Eq. 2)}$$

, where $\lambda_i(\tau)$ is the *i*-th eigenvalue of $T(\tau)$. Each implied timescale corresponds to the timescale of a transition mode encoded in an eigenvector that represents the transition between two subsets of states. For a more detailed theory on MSMs, we refer readers to a review by Wang et al. (44) and a book by Pande et al. (45).

B.2 quasi-Markov State Model (qMSM)

In qMSM, the memory kernels of conformational dynamics are explicitly considered, and the dynamics is propagated using a Generalized Master equation (GME) (34):

$$\dot{\boldsymbol{T}}(n\Delta t) = \boldsymbol{T}(n\Delta t)\dot{\boldsymbol{T}}(0) + \int_0^{\min\{t,\tau_K/\Delta t\}} \boldsymbol{T}(n - m(\Delta t))\boldsymbol{K}(m\Delta t)dt \qquad \text{(Eq. 3)}$$

, where memory kernels $\boldsymbol{K}(m\Delta t)$ can be calculated iteratively based on TPMs ($\boldsymbol{T}(n\Delta t)$) and their derivatives ($\dot{\boldsymbol{T}}(n\Delta t)$) at t=0, Δt, $2\Delta t$ … $n\Delta t$.

$$\boldsymbol{K}(n\Delta t) = \frac{\dot{\boldsymbol{T}}(n\Delta t) - \boldsymbol{T}(0)\dot{\boldsymbol{T}}(n\Delta t)}{\Delta t} - \sum_{m=1}^{n-1} \boldsymbol{T}(n - m(\Delta t))\boldsymbol{K}(m\Delta t) \quad \text{(Eq. 4)}$$

The obtained memory kernel $\boldsymbol{K}(n\Delta t)$ is a transition tensor, where a memory kernel relaxation function is associated with the transition between a pair of states. From the memory kernels at different time intervals (i.e. at $t = 0, \Delta t, 2\Delta t, 3\Delta t \ldots n\Delta t$), we can calculate the mean integration of the memory kernel (MIK) using the following expression (34):

$$MIK(t) = \frac{1}{N}\sqrt{\sum_{i,j=1}^{N}\left(\int_0^t K_{ij}(t')dt'\right)^2} \quad \text{(Eq. 5)}$$

We define the memory kernel lifetime (τ_k) as the time at which the $MIK(t)$ converges. With τ_k, we can apply the GME (Eq. 3) to propagate the dynamics. The detailed theory of qMSM is available in ref. (34).

C. Outline for building a qMSM

qMSM relies on the same state assignments as those used in MSMs. Hence, several key steps to build qMSM are the same as in the MSM construction. Our workflow presented below is adapted from our recently published MSM protocols (1, 33, 44):

1) <u>Modelling and generating the initial path between functional states</u>

   Structures resolved from experiments, such as Cryo-EM and X-ray crystallography, often represent the free energy minima of the conformational space and the starting conformations for an MD simulation. When studying functional conformational changes, we will have at least two structures (green and blue dots on Figure 1A) representing either the starting or ending states of a biological process. For example, in our qMSM that depicts the gate opening motion in *Taq* RNAP (37), the two modelled starting structures are a closed DNA loading gate and an open DNA loading gate. To prepare the system for MD

simulations, modelling tools such as SwissModeller (46) and MODELLER (47) are commonly used to fill in missing protein residues and nucleotides in the starting structures. Packages, such as Propka3, (48) are frequently used to calculate the protonation state of individual residues in the local environment.

Various tools can be used to obtain an initial pathway (Figure 1A) for the transition between these functional states, including Climber (49), Steered MD/Targeted MD (50, 51), Coarse-grained MD simulations (CG-MD (52)), and Metadynamics simulations (53). Optimization of the initial path is recommended using tools such as the String method (54) and traveling-salesman-based automatic path searching (TAPS) (55).

2) All-atom MD sampling

To build a qMSM, MD simulations are conducted from seed conformations selected from the initial pathways (Figure 1B). These simulations are often run in parallel to sample the free energy landscape extensively. This approach reduces the computational time compared to conducting individual ultra-long MD simulations. For the alanine dipeptide system used in this tutorial, GROMACS (56) was used to perform MD simulations. A detailed tutorial to conduct MD simulation in GROMACS can be referred to in ref. (57).

3) Automatic feature selection

To correctly identify slow dynamic modes, the selected structural features should preferably correlate with the conformational changes being studied. Internal coordinates (such as pairwise distances and dihedral angles) are calculated from MD trajectories and used as the structural features. In a large biological system, complete featurization is impractical and redundant. For instance, the *Taq* RNAP system used for qMSM construction has ~3300 residues (37). If all residues were considered, it would provide a sizeable feature set of ~5 million pairwise distances between all Cα atoms. Thus, for

studying functional conformational changes, we recommend pre-selecting the atoms that may be involved in the conformational changes based on prior knowledge and biological intuition as the initial feature set.

To further reduce the number of features, we recommend an automatic feature selection algorithm called Spectral oASIS (39) (implemented in PyEMMA 2.5.7), which is based on the Nyström matrix operation theory to reconstruct the leading eigenvalues and eigenvectors of the full time-lagged covariance matrix from a subset of the matrix (Figure 1C). Spectral oASIS can be employed to select a subset of features that can approximate the timescale of the time-lagged covariance analysis obtained from the initial feature set (39).

4) Dimensionality reduction

Time-lagged independent component analysis (tICA) is commonly used to reduce the dimension of the input features selected in the previous step and obtain the collective variables (CVs) used for clustering (Figure 1D). tICA identifies the slow modes by finding the optimal linear combination of the input features corresponding to the maximal time-lagged autocorrelation (42, 58, 59). The first few eigenvectors, or tICs, identified by the tICA analysis can be regarded as a linear approximation of the slowest collective variables, providing reaction coordinates for state decomposition.

5) Clustering MD conformations into microstates

If MD conformations belong to the same energy basin in the tlCA subspace, where the system can transition in between these conformations rapidly, they are clustered into the same microstate (Figure 1E). Commonly used clustering methods for the MSM construction include centered-based algorithms (e.g., k-means (60) and k-centers (61)) and density-based clustering methods (e.g., DBSCAN (62) and APLoD (63)).

6) Selection of hyperparameters for the microstate MSM construction

To avoid overfitting and to find the optimum hyperparameters to build MSMs, Cross-validation can be employed, where the model is constructed in the training dataset and validated using the test dataset. MSM hyperparameters include the feature sets, the tICA relaxation time, the number of tlCs, and the number of microstates. The models constructed from different hyperparameter sets are scored by objective metrics such as generalized matrix Rayleigh quotient (GMRQ) (64) and VAMP-2 scores (65). GMRQ and VAMP-2 evaluate the quality of MSMs based on their ability to capture the slowest dynamic modes via the variational principle.

7) Kinetic lumping to obtain macrostates and the construction of qMSMs

In a microstate-MSM, the large number of microstates (hundreds to thousands) often hinders the comprehension of the molecular mechanisms underlying functional conformational changes. Kinetic lumping methods such as PCCA+(66), Spectral clustering (67) and Gibbs clustering (68) can further group microstates into a few 'macro'-states (hereafter macrostates) for the interpretation of biological mechanisms.

In qMSM, we will use the macrostate-TPMs and their derivatives at different lag times as an input to calculate memory kernels and determine their lifetime τ_K according to (Eq. 5). The memory kernels are then used to construct qMSM, which can be further validated using the Chapman-Kolmogorov test. In this test, a good model should demonstrate the consistency between the residence probabilities predicted by the qMSM and the ones obtained directly from MD simulations.

8) Calculation of the mean first passage time (MFPT)

One useful kinetic property to describe the functional conformational changes is the mean first passage time (MFPT). MFPT refers to the average time for the first transition from a

starting state *i* to the final state *j* (69). Unlike implied timescales that describe an aggregated transition time across groups of states, MFPT represents the transition time between a pair of specific conformational states. It contains contributions from both direct transitions from state *i* to *j* and indirect transitions from state *i* to state *j* bypassing other intermediate states.

D. Protocol to Build a qMSM

D.1 Materials and Prerequisite

In this protocol, we will use alanine dipeptide to demonstrate how to build a qMSM. The MD trajectories of the alanine dipeptide system (100 10ns-long trajectories, with a total size of ~2.2GBs) and other materials needed for this protocol can be downloaded from the following website: https://github.com/ykhdrew/qMSM_tutorial.

We will use two popular Python packages to build MSMs and qMSMs: MSMbuilder (70) and PyEMMA (71). On the Github page, the folder 'notebook' contains a series of Jupyter notebooks at different stages of the model construction. Please do not move the Jupyter notebooks out of the folder, or the path string will be incorrect when loading input/output files at different stages of this demonstration.

To install these packages, we recommend using Anaconda that cross-checks for package dependencies and automatically installs missing programs. Please install Anaconda if not already done so.

After installing Anaconda, running the following shell commands will create a conda environment named "msmbuilder" that contains MSMbuilder and other libraries (e.g., Jupyter) used throughout this protocol:

```
$conda env create -n msmbuilder -f environment.yml
```

For the installation of PyEMMA, the readers may refer to http://pyemma.org.

D.2 Feature selection with Spectral oASIS

In the alanine dipeptide system, we will generate an initial feature set composed of 45 pairwise distances between all heavy atoms. We will then reduce the feature set size with Spectral oASIS.

D.2a Featurization

The Jupyter notebook Featurization.pynb contains the scripts for featurization in the example system. Opening the notebook, we first import the libraries to be used e.g.:

```
>>>from msmbuilder.featurizer import AtomPairsFeaturizer
>>>import mdtraj as md
```

and indicate the folder path for MD trajectories:

```
>>>trajlist=glob(trajDir+"*.xtc")
>>>trajlist.sort(key=lambda f: int(re.sub('\D', '', f)))
```

Running the function create_pairwise_index() will generate an atom pair index from the selected list of atoms (i.e., AtomIndices.dat):

```
>>>atom_set =featdir+"AtomIndices.dat"
>>>atom_pair_list=create_pairwise_index(atom_set)
```

Meanwhile, feat () uses the featurizer module in MSMBuilder to calculate the pairwise distances from our MD data:

```
>>>for n,i in enumerate(trajlist):
...   feat_ = feat(atom_pairs_feat,i,topfile)
...   np.save("{}features/{}.npy".format(flatirons), feat_)
```

This will output a folder named “features” that contains the numpy binary files for the pairwise distances calculated from each MD trajectory.

D.3b. Spectral oASIS

Important features are then chosen from this initial feature set using Spectral oASIS (39), as implemented in PyEMMA. Before running this part of the demonstration, please make sure PyEMMA is installed in the current conda/python environment.

Opening Spectral oASIS-Parallel.ipynb, we first import the Spectral oASIS library:

```
>>>from pyemma.coordinates.transform.nystroem_tica import *
```

At a particular tICA lag time (0.2ps in this example), Spectral oASIS selects the features for different sizes (number of features) of feature sets. After running the second cell, it should print

```
>>>no. of features tested: [4,8,12,16,20,24,28]
```

out a series of feature set sizes to be tested:

Spectral oASIS may take a long time to finish and we include a wrapper SpectralOasis() for running different conditions in parallel:

```
>>>with Pool() as pool:
...   t_timescales=dict(pool.imap_unordered(SpectralOasis, columns))
```

This gives an index for the raw feature set (i.e., selected_feature_column(feature set size).txt). Please note that the output from Spectral oASIS is stochastic in nature and the features selected vary slightly for every run.

Plotting the 1st timescale against the sizes of feature set, we can select a feature set of 24 pairwise distances for dimensionality reduction, as the timescale becomes invariant from this point onwards (Figure 2A).

Notes on feature selection:

1. The initial atom set for feature selection (the atom set in AtomIndices.dat) largely depends on the conformational change to be studied, e.g., the interdomain distance and ligand-protein distance.

2. For demonstration purposes, we only show one tICA lag time for running Spectral oASIS. In practice, Spectral oASIS using several tICA lag times should be performed and the quality of the selected feature sets should be tested using GMRQ. We will cover this more in the "Selection of Optimal Hyperparameters by GMRQ" section

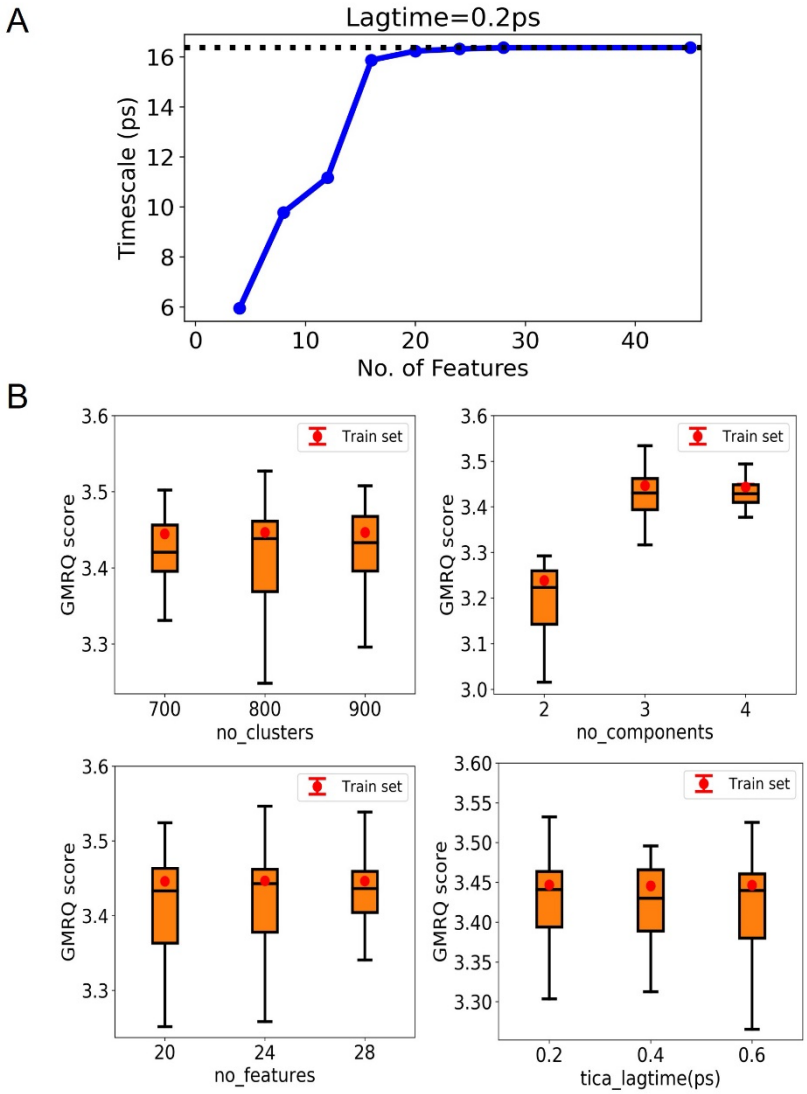

Figure 2. (A). tlCA timescale (at tICA lag time=0.2ps) against number of features. **(B).** Box and Whisker plot for GMRQ test scores with different MSM hyperparameters.

D.4 Dimensionality reduction with tICA

In this example, tICA is performed on the selected features to obtain a conformational subspace composed 3 tICs.

Opening TICA.ipynb, we first import the tICA module from MSMbuilder:

```
>>>from msmbuilder.io import load_trajs, save_trajs, save_generic
>>>from msmbuilder.decomposition import tICA
```

Here, the tlCA lag time used for model construction was 0.2ps. To compare the quality of qMSM built from different parameters, we will also test different tICA lag times (0.4ps and 0.6ps) and several feature sets (20 features and 28 features):

```
>>>tica=tICA(n_components=10, lag_time=tica_lagtime[n][m]],
             kinetic_mapping=True)
>>>tica.fit(ftrajs.values())
```

Under each output directory named after the features and tlCA lag time used, there is a folder containing the tlCA coordinates for the MD trajectories as well as a pickle file (tica.pickl) containing the parameters for tlCA.

Notes on performing tICA:

1. After performing tlCA, we can project MD conformations along the first few tlCs to check if they are relevant to the protein conformational change being studied.

2. Since a MD trajectory can be trapped in certain local minima and disconnected from the rest of the MD conformations on the tlCA subspace, performing tlCA is one way to check for the quality of MD sampling. Projection of MD conformations onto the first few tICs provides a way to visualize the data and identifies disconnected trajectories.

3. When a MD trajectory is indeed disconnected, additional MD sampling (either by extending the MD trajectories or running MD simulations with a new random seed) or adaptive sampling with packages such as FAST (72) and HTMD (73) can be performed.

D.5 Clustering MD conformations using KCenters

MD conformations are then grouped into microstates according to their kinetic similarity on the reduced dimensional space. Here, we employed K-Centers (61) algorithm and used 800 clusters for clustering.

In the Clustering.ipynb notebook, a wrapper run_KCenters_parallel() parallelly performs clustering with different parameters (See the comments in the Jupyter notebook for details):

```
>>>if __name__=="__main__":
...   run_KCenters_parallel(no_of_features,no_components,
                            no_clusters ,tica_lagtime)
```

This will output the clustering assignment (saved as clustering_assignments.npy) for each set of the parameters tested.

D.6 Selection of the optimal hyperparameters by GMRQ

After clustering, cross-validation with GMRQ scoring is performed. This step aims to select hyperparameters that lead to a model with the highest test score. Using the KFold algorithm from scikit-learn (74), the MD data can be split into a training and testing dataset for cross-validation

In the GMRQ.ipynb notebook, the GMRQ test scores for models built using a different number of features, number of tlCA components, tlCA lag time, and number of clusters are calculated. Here, we vary these parameters one at a time. For instance, models are built from a different number of features (20, 24 and 28 features) while the rest of the parameters are kept the same (i.e., number of tlCA component=3, tlCA lag time=0.2ps and number of K-Centers=800).

Below, we first import relevant modules from scikit-learn and MSMbuilder :

```
>>>from msmbuilder.msm import MarkovStateModel
>>>from sklearn.cross_validation import KFold
>>>from sklearn.pipeline  import Pipeline
```

The function of GMRQ() is to accept lists of parameters as arguments and load the corresponding clustering assignment:

```
>>>GMRQ(no_of_features,no_of_components,
        no_clusters ,tica_lagtime,parameter,gmrq_dir,aplod_dir)
```

KFold cross-validation will be performed 10 times, giving 60 GMRQ scores for each parameter set. In the next cell of the notebook, we can display the test scores as a Box and Whisker Plot with the function GMRQ_Plot(). Since the splitting of the dataset is performed randomly, the GMRQ test scores vary slightly every time the program is run.

In our calculation, the median score is the highest when the number of tlCA component=3, tlCA lag time=0.2ps and number of K-Centers clusters=800 (Figure 2B). Hence, we are using these parameters to generate a microstate MSM.

Notes on GMRQ

1. In this tutorial, we only tested a subset of hyperparameters and generated their GMRQ test scores. When building a microstate MSM, all possible combinations of the hyperparameters should be tested.
2. GMRQ scores across different MSM lag times are not comparable. Hence, multiple MSM lag times should also be tested.

D.7 Construction of microstate MSM and kinetic lumping

D.7a . Microstate MSM

After selecting the hyperparameters, we can now construct a microstate MSM. Opening the microstate_MSM&PCCA.ipynb notebook, we first import the library for MSM construction:

```
>>>from msmbuilder.msm import MarkovStateModel
```

The function its_bootstrap() subsamples the cluster assignment for constructing an MSM and calculates the implied timescales at different MSM lag times:

```
>>>data=its_bootstrap(lagtimes,no_of_traj)
>>>data.to_pickle("timescales.pickl')
```

This returns a pandas DataFrame object containing the raw data. Then, load_its() calculates the average and standard deviation of the implied timescales:

```
>>>av_dict,error_dict=load_its(msm_dir+'timescales.pickl',
                               n_timescales=4,lagtimes=lagtimes)
```

Plotting the implied timescale against the MSM lag time shows that the implied timescales become invariant from 10ps onwards (Figure 3A):

```
>>>fig, ax = plt.subplots(figsize=(7,5))
>>>plot_timescales(ax,4,lagtimes,av_dict,error_dict)
```

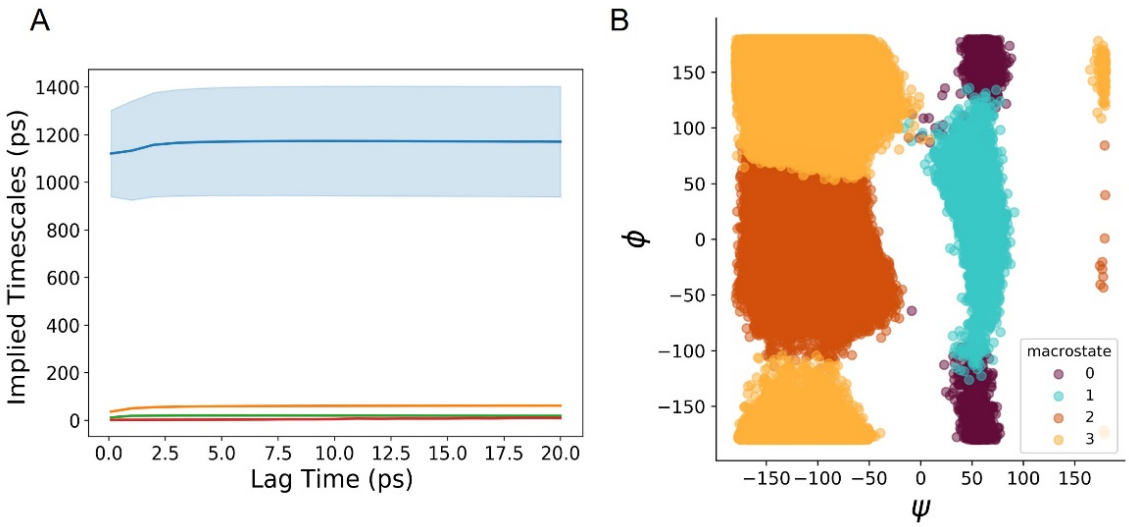

Figure 3. (A). Implied timescale against MSM lag time plot. **(B).** Projection of MD trajectories on the torsional angles of alanine dipeptide. Each dot represents a MD conformation, with the color indicating its macrostate assignment.

D.7b. Kinetic Lumping and Generating Macrostate TPM

To facilitate biological interpretation of the model, the microstates are further lumped into macrostates using PCCA+ (66). Eigen-decomposition of the microstate transition matrix shows a stable gap between the 3rd and the 4th slowest timescales (Figure 3A). Hence, 4 macrostates are chosen for our model. The following scripts in micorstate_MSM&PCCA.ipynb perform lumping at lag time = 10ps:

```
>>>from msmbuilder.lumping import PCCAPlus
>>>msm=MarkovStateModel(verbose=True,lag_time=100,
                        reversible_type='transpose',
                        ergodic_cutoff='off')
>>>pcca = PCCAPlus.from_msm(msm, n_macrostates=4)
>>>lumped_trajs = pcca.fit_transform(assignments)
>>>lumped_trajs=np.concatenate(lumped_trajs)
>>>save_generic(pcca,"{}pcca.pickl".format(msm_dir))
>>>np.save("{}lumping_assignment.npy".format(qmsm_dir),lumped_trajs)
```

This will output a pickle file containing the parameters for lumping and a numpy binary file with the macrostate assignment for the MD trajectories.

Here, we are visualizing the lumping results with seaborn (75) by projecting the metastable states onto the two torsional angle, ϕ *and* ψ, of alanine dipeptide:

```
>>>sns.lmplot(data=df,x='psi',y='phi',hue='macrostate’,
              palette=macrostate_color,fit_reg=False,legend=True,
              legend_out=False)
```

On the Ramachandran plot, PCCA+ groups the microstate into four different metastable states (Figure 3B). Note that the macrostate indices may change every time you run the PCCA+ module in MSMbuilder but the microstate mapping to macrostate should be consistent.

After obtaining the macrostate assignment from PCCA+, a series of macrostate TPMs at different lag times can be generated for MIK calculation (Eq. 5) in the next section. In the fifth cell of the notebook, prep_macroTPM() generates 500 TPMs constructed at different lag times, τ = ({0.1ps,0.2ps,0.3ps…50ps}.Note the unit used in this function is in picosecond) :

```
>>>prep_macroTPM(initial_lagtime=1,ending_lagtime=500,
                  delta_time=1, md_timestep=0.1)
```

D.8. The qMSM construction

D.8a Memory kernel and MIK calculation

After obtaining the macrostate TPM, we can start building our qMSM.

Opening the qMSM.ipynb notebook, the qMSM libraries and the macrostate TPM are loaded:

```
>>>import sys
>>>sys.path.append('./qMSM')
>>>from QuasiMSM_ModuleBuilder import *
>>>input_data = np.load("./qMSM/qMSM_TPM.npy",
                        dtype=float)
```

In this part of the tutorial, the time unit is 0.1ps, as defined in the variable delta_time.

The following scripts calculate the derivative of TPM $\dot{\boldsymbol{T}}$ (Eq.2):

```
>>>qmsm = QuasiMSM(input_len=500, delta_time=0.1, dimension=4)
>>>qmsm.GetData(input_data)
>>>qmsm.Pre_SetData()
>>>qmsm.Get_dTPM_dt()
```

Then, the memory $\boldsymbol{K}$ kernel and the Mean Integration of Memory Kernel (MIK) are also calculated using the following scripts:

```
>>>km = qmsm.Calculate_K_matrix(cal_step=400)
>>>qmsm.MeanIntegralKernel(MIK_time=50 , figure=True,
                           outdir=qmsm_dir)
```

Plotting the MIK against lag time, MIK converges at t =15ps, giving the memory lifetime, τ_K (Figure 4A):

```
>>>qmsm.KernelPlot(km)
```

Hence, a qMSM can be built from $\boldsymbol{K}(t = 15ps)$:

```
>>>qmsm_tpm, qmsm_tpm_time=qmsm.QuasiMSMPrediction(kernel_matrix=km,
                                                   tau_k=20,
                                                   end_point=200,
                                                   outasfile=True,
                                                   outdir=qmsm_dir)
```

The output is a file containing 200 propagated TPMs constructed using lag time = {0.1ps, 0.2ps, …, 20ps}. The TPMs for extended lag time, which is longer than the length of the MD simulation (e.g.: 20ns), can be obtained by modifying the end_point (e.g.: to 200,000 steps). Generating TPMs at extended lag time may be needed to calculate the slow dynamics between the metastable states as discussed in section C.8 calculation of MFPT.

Calculating the time-averaged root mean squared error (RMSE, the formula is in ref. (34)) for qMSM dynamics serves as another way to validate the choice of τ_K:

```
>>>qMSM_RMSE, MSM_RMSE=qmsm.RMSE(kernel=km, end_point=200,
                                  figure=True,
                                  outasfile=False,outdir=qmsm_dir)
```

Figure 4C shows the RMSE for qMSM-predicted dynamics that converge at 15ps.

8b. Chapman Kolmogorov Test

To validate the qMSM, a Chapman Kolmogorov Test can be performed (20, 76) to compare the kinetics predicted by qMSM with that from MD trajectories and MSM (Figure 4B):

```
>>>qmsm.CK_figure(qMSM_TPM=qmsm_tpm_time, MSM_TPM=msm_tpm_time,
                  add_iden_mat=True, diag=True, grid=[4,4],
                  slice_dot=10, outdir=qmsm_dir)
```

This will generate four plots each representing the residence probability of a macrostate against lag time (Figure 4B). In the plot, qMSM reproduces the kinetics (red lines) predicted by MD trajectories (grey dots) while MSM does not (green lines).

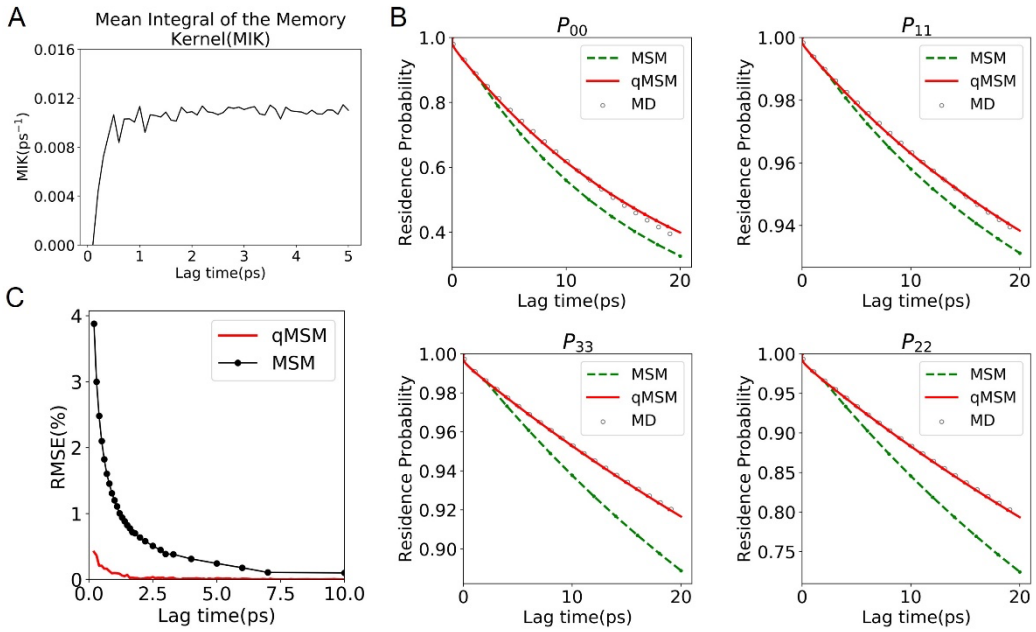

Figure 4. (A). Mean Integration of Memory Kernel (MIK) against lag time. **(B).** CK test on macrostate residence probability. **(C).** Root mean square errors (RMSEs) between the qMSM-predicted dynamics and MD simulations.

D.9. Analysis

D.9a. MFPT calculation

At this part of the tutorial, we can calculate MFPTs between each pair of macrostates from the 4-state qMSM to understand the kinetics for interstate transitions.

Opening the Analysis notebook, there is a function to calculate MFPT from $\boldsymbol{T}(\tau = 20ps)$:

```
>>>mfpt=mfpt(tpm,lagtime=200)
```

The output contains a matrix $\in \mathbb{R}^{4\times4}$, in which the *ij*-th element represents the MFPT (sharing the same time unit as the input TPM, 0.1ps in this tutorial) from the *i*-th marcostate to the *j*-th marcostate.

Notes on MFPT calculation:

1. The MFPT script provided here is solving a linear system of equations that describes MFPT from state i to the final state f, F_{if} ,where $F_{if} = \tau + \sum_{i\neq j} P_{ij}F_{jf}$. A detailed theory can be referred to in ref (69).

2. When calculating MFPT, we should test TPMs constructed at different lag times and chose the lag time at which the MFPT converges. Supplementary Note 9 of Ref. (38) documents the workflow for calculating MFPT with the TPM constructed from qMSM.

D.9b.Macrostate sampling

In the second cell of the notebook, we can also sample the conformations in each macrostate:

```
>>>sample_macrostate(trajDir,topfile,lumped_assignment,
                      analysis_dir,no_of_sample)
```

From these conformations, one can perform residue-residue contact analysis and Pearson Correlation coefficient calculations among residues of interest. This helps to understand the biological significance of the macrostates.

E. Concluding Remarks

In this procotol, we showcased the workflow for quasi-Markov State Model construction using the alanine dipeptide system as an example. Using a GME to propagate dynamics, we demonstrated that our qMSM framework is advantageous over MSM since it greatly reduces the number of metastable states required to describe a biological event. Among existing variants of MSM (e.g., core-set MSM (77) and Hidden MSM (78)), we believe that qMSM holds promise in studying functional dynamics of biomolecules.

F. Acknowledgements

We acknowledge the support from the Office of the Vice-Chancellor for Research and Graduate Education at the University of Wisconsin-Madison with funding from the Wisconsin Alumni Research Foundation. We also acknowledge the support from the Hirschfelder Professorship Fund and start-up funds from the College of Letters and Sciences of the University of Wisconsin-Madison.

H. Reference

1. Wang X, Unarta IC, Cheung PP, Huang X. Elucidating molecular mechanisms of functional conformational changes of proteins via Markov state models. Curr Opin Struct Biol. 2021;67:69-77.
2. Hollingsworth SA, Dror RO. Molecular Dynamics Simulation for All. Neuron. 2018;99(6):1129-43.
3. Karplus M, Kuriyan J. Molecular dynamics and protein function. Proc Natl Acad Sci U S A. 2005;102(19):6679-85.
4. Piston DW, Kremers GJ. Fluorescent protein FRET: the good, the bad and the ugly. Trends Biochem Sci. 2007;32(9):407-14.
5. Karplus M, McCammon JA. Molecular dynamics simulations of biomolecules. Nat Struct Biol. 2002;9(9):646-52.
6. Silva D-A, Weiss DR, Pardo Avila F, Da L-T, Levitt M, Wang D, et al. Millisecond dynamics of RNA polymerase II translocation at atomic resolution. Proceedings of the National Academy of Sciences. 2014;111(21):7665-70.
7. Markwick PRL, McCammon JA. Studying functional dynamics in bio-molecules using accelerated molecular dynamics. Physical Chemistry Chemical Physics. 2011;13(45):20053-65.
8. Shaw DE, Adams PJ, Azaria A, Bank JA, Batson B, Bell A, et al. Anton 3: twenty microseconds of molecular dynamics simulation before lunch. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis; St. Louis, Missouri: Association for Computing Machinery; 2021. p. Article 1.
9. Pande VS, Beauchamp K, Bowman GR. Everything you wanted to know about Markov State Models but were afraid to ask. Methods. 2010;52(1):99-105.
10. Husic BE, Pande VS. Markov State Models: From an Art to a Science. J Am Chem Soc. 2018;140(7):2386-96.
11. Chodera JD, Noe F. Markov state models of biomolecular conformational dynamics. Curr Opin Struct Biol. 2014;25:135-44.
12. Chodera JD, Singhal N, Pande VS, Dill KA, Swope WC. Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. The Journal of Chemical Physics. 2007;126(15):155101.
13. Pan AC, Roux B. Building Markov state models along pathways to determine free energies and rates of transitions. The Journal of Chemical Physics. 2008;129(6):064107.
14. Zhang BW, Dai W, Gallicchio E, He P, Xia J, Tan Z, et al. Simulating Replica Exchange: Markov State Models, Proposal Schemes, and the Infinite Swapping Limit. The Journal of Physical Chemistry B. 2016;120(33):8289-301.
15. Morcos F, Chatterjee S, McClendon CL, Brenner PR, López-Rendón R, Zintsmaster J, et al. Modeling Conformational Ensembles of Slow Functional Motions in Pin1-WW. PLOS Computational Biology. 2010;6(12):e1001015.
16. Huang X, Bowman GR, Bacallado S, Pande VS. Rapid equilibrium sampling initiated from nonequilibrium data. Proceedings of the National Academy of Sciences. 2009;106(47):19765.
17. Malmstrom RD, Lee CT, Van Wart AT, Amaro RE. Application of Molecular-Dynamics Based Markov State Models to Functional Proteins. Journal of Chemical Theory and Computation. 2014;10(7):2648-57.
18. Buchete N-V, Hummer G. Coarse Master Equations for Peptide Folding Dynamics. The Journal of Physical Chemistry B. 2008;112(19):6057-69.
19. Yao Y, Cui RZ, Bowman GR, Silva D-A, Sun J, Huang X. Hierarchical Nyström methods for constructing Markov state models for conformational dynamics. The Journal of Chemical Physics. 2013;138(17):174106.
20. Prinz JH, Wu H, Sarich M, Keller B, Senne M, Held M, et al. Markov models of molecular kinetics: generation and validation. J Chem Phys. 2011;134(17):174105.

21. Kohlhoff KJ, Shukla D, Lawrenz M, Bowman GR, Konerding DE, Belov D, et al. Cloud-based simulations on Google Exacycle reveal ligand modulation of GPCR activation pathways. Nature Chemistry. 2014;6(1):15-21.
22. Da L-T, E C, Duan B, Zhang C, Zhou X, Yu J. A Jump-from-Cavity Pyrophosphate Ion Release Assisted by a Key Lysine Residue in T7 RNA Polymerase Transcription Elongation. PLOS Computational Biology. 2015;11(11):e1004624.
23. Da L-T, Wang D, Huang X. Dynamics of Pyrophosphate Ion Release and Its Coupled Trigger Loop Motion from Closed to Open State in RNA Polymerase II. Journal of the American Chemical Society. 2012;134(4):2399-406.
24. Malmstrom RD, Kornev AP, Taylor SS, Amaro RE. Allostery through the computational microscope: cAMP activation of a canonical signalling domain. Nature Communications. 2015;6(1):7588.
25. Wang B, Sexton RE, Feig M. Kinetics of nucleotide entry into RNA polymerase active site provides mechanism for efficiency and fidelity. Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms. 2017;1860(4):482-90.
26. Khaled M, Gorfe A, Sayyed-Ahmad A. Conformational and Dynamical Effects of Tyr32 Phosphorylation in K-Ras: Molecular Dynamics Simulation and Markov State Models Analysis. The Journal of Physical Chemistry B. 2019;123(36):7667-75.
27. Barros EP, Demir Ö, Soto J, Cocco MJ, Amaro RE. Markov state models and NMR uncover an overlooked allosteric loop in p53. Chemical Science. 2021;12(5):1891-900.
28. Feng J, Selvam B, Shukla D. How do antiporters exchange substrates across the cell membrane? An atomic-level description of the complete exchange cycle in NarK. Structure. 2021;29(8):922-33.e3.
29. Son CY, Yethiraj A, Cui Q. Cavity hydration dynamics in cytochrome c oxidase and functional implications. Proceedings of the National Academy of Sciences. 2017;114(42):E8830.
30. Da L-T, Pardo Avila F, Wang D, Huang X. A Two-State Model for the Dynamics of the Pyrophosphate Ion Release in Bacterial RNA Polymerase. PLOS Computational Biology. 2013;9(4):e1003020.
31. Peng S, Wang X, Zhang L, He S, Zhao XS, Huang X, et al. Target search and recognition mechanisms of glycosylase AlkD revealed by scanning FRET-FCS and Markov state models. Proc Natl Acad Sci U S A. 2020;117(36):21889-95.
32. Da LT, Pardo-Avila F, Xu L, Silva DA, Zhang L, Gao X, et al. Bridge helix bending promotes RNA polymerase II backtracking through a critical and conserved threonine residue. Nat Commun. 2016;7:11244.
33. Konovalov KA, Unarta IC, Cao S, Goonetilleke EC, Huang X. Markov State Models to Study the Functional Dynamics of Proteins in the Wake of Machine Learning. JACS Au. 2021;1(9):1330-41.
34. Cao S, Montoya-Castillo A, Wang W, Markland TE, Huang X. On the advantages of exploiting memory in Markov state models for biomolecular dynamics. J Chem Phys. 2020;153(1):014105.
35. Voelz VA, Bowman GR, Beauchamp K, Pande VS. Molecular Simulation of ab Initio Protein Folding for a Millisecond Folder NTL9(1−39). Journal of the American Chemical Society. 2010;132(5):1526-8.
36. Qiao Q, Bowman GR, Huang X. Dynamics of an Intrinsically Disordered Protein Reveal Metastable Conformations That Potentially Seed Aggregation. Journal of the American Chemical Society. 2013;135(43):16092-101.
37. Unarta IC, Cao S, Kubo S, Wang W, Cheung PP-H, Gao X, et al. Role of bacterial RNA polymerase gate opening dynamics in DNA loading and antibiotics inhibition elucidated by quasi-Markov State Model. Proceedings of the National Academy of Sciences. 2021;118(17):e2024324118.
38. Zhu L, Jiang H, Cao S, Unarta IC, Gao X, Huang X. Critical role of backbone coordination in the mRNA recognition by RNA induced silencing complex. Communications Biology. 2021;4(1):1345.

39. Litzinger F, Boninsegna L, Wu H, Nüske F, Patel R, Baraniuk R, et al. Rapid Calculation of Molecular Kinetics Using Compressed Sensing. Journal of Chemical Theory and Computation. 2018;14(5):2771-83.
40. Lane TJ, Shukla D, Beauchamp KA, Pande VS. To milliseconds and beyond: challenges in the simulation of protein folding. Curr Opin Struct Biol. 2013;23(1):58-65.
41. Noe F, Fischer S. Transition networks for modeling the kinetics of conformational change in macromolecules. Curr Opin Struct Biol. 2008;18(2):154-62.
42. Pérez-Hernández G, Paul F, Giorgino T, De Fabritiis G, Noé F. Identification of slow molecular order parameters for Markov model construction. J Chem Phys. 2013;139(1):015102.
43. Bowman GR, Huang X, Pande VS. Using generalized ensemble simulations and Markov state models to identify conformational states. Methods. 2009;49(2):197-201.
44. Wang W, Cao S, Zhu L, Huang X. Constructing Markov State Models to elucidate the functional conformational changes of complex biomolecules. WIREs Computational Molecular Science. 2018;8(1):e1343.
45. Bowman GR, Pande VS, Noé F, editors. An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation. Advances in Experimental Medicine and Biology; 2014.
46. Arnold K, Bordoli L, Kopp J, Schwede T. The SWISS-MODEL workspace: a web-based environment for protein structure homology modelling. Bioinformatics. 2006;22(2):195-201.
47. Webb B, Sali A. Comparative Protein Structure Modeling Using MODELLER. Curr Protoc Bioinformatics. 2016;54:5.6.1-5.6.37.
48. Olsson MHM, Søndergaard CR, Rostkowski M, Jensen JH. PROPKA3: Consistent Treatment of Internal and Surface Residues in Empirical pKa Predictions. Journal of Chemical Theory and Computation. 2011;7(2):525-37.
49. Weiss DR, Levitt M. Can morphing methods predict intermediate structures? J Mol Biol. 2009;385(2):665-74.
50. Schlitter J, Engels M, Kruger P. Targeted molecular dynamics: a new approach for searching pathways of conformational transitions. J Mol Graph. 1994;12(2):84-9.
51. Isralewitz B, Gao M, Schulten K. Steered molecular dynamics and mechanical functions of proteins. Curr Opin Struct Biol. 2001;11(2):224-30.
52. Kenzaki H, Koga N, Hori N, Kanada R, Li W, Okazaki K, et al. CafeMol: A Coarse-Grained Biomolecular Simulator for Simulating Proteins at Work. J Chem Theory Comput. 2011;7(6):1979-89.
53. Laio A, Parrinello M. Escaping free-energy minima. Proc Natl Acad Sci U S A. 2002;99(20):12562-6.
54. Pan AC, Sezer D, Roux B. Finding transition pathways using the string method with swarms of trajectories. J Phys Chem B. 2008;112(11):3432-40.
55. Zhu L, Sheong FK, Cao S, Liu S, Unarta IC, Huang X. TAPS: A traveling-salesman based automated path searching method for functional conformational changes of biological macromolecules. J Chem Phys. 2019;150(12):124105.
56. Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, et al. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX. 2015;1-2:19-25.
57. Lemkul JA. From Proteins to Perturbed Hamiltonians: A Suite of Tutorials for the GROMACS-2018 Molecular Simulation Package [Article v1.0]. Living Journal of Computational Molecular Science. 2018;1(1):5068.
58. Naritomi Y, Fuchigami S. Slow dynamics of a protein backbone in molecular dynamics simulation revealed by time-structure based independent component analysis. J Chem Phys. 2013;139(21):215102.
59. Schwantes CR, Pande VS. Improvements in Markov State Model Construction Reveal Many Non-Native Interactions in the Folding of NTL9. Journal of Chemical Theory and Computation. 2013;9(4):2000-9.

60. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002;24(7):881-92.
61. Gonzalez TF. Clustering to minimize the maximum intercluster distance. Theoretical Computer Science. 1985;38:293-306.
62. Ester M, Kriegel H-P, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining; Portland, Oregon: AAAI Press; 1996. p. 226–31.
63. Liu S, Zhu L, Sheong FK, Wang W, Huang X. Adaptive partitioning by local density-peaks: An efficient density-based clustering algorithm for analyzing molecular dynamics trajectories. J Comput Chem. 2017;38(3):152-60.
64. Husic BE, McGibbon RT, Sultan MM, Pande VS. Optimized parameter selection reveals trends in Markov state models for protein folding. The Journal of Chemical Physics. 2016;145(19):194103.
65. Wu H, Noé F. Variational Approach for Learning Markov Processes from Time Series Data. Journal of Nonlinear Science. 2020;30(1):23-66.
66. Röblitz S, Weber M. Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification. Advances in Data Analysis and Classification. 2013;7(2):147-79.
67. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic; Vancouver, British Columbia, Canada: MIT Press; 2001. p. 849–56.
68. Wang W, Liang T, Sheong FK, Fan X, Huang X. An efficient Bayesian kinetic lumping algorithm to identify metastable conformational states via Gibbs sampling. The Journal of Chemical Physics. 2018;149(7):072337.
69. Singhal N, Pande VS. Error analysis and efficient sampling in Markovian state models for molecular dynamics. The Journal of Chemical Physics. 2005;123(20):204909.
70. Harrigan MP, Sultan MM, Hernández CX, Husic BE, Eastman P, Schwantes CR, et al. MSMBuilder: Statistical Models for Biomolecular Dynamics. Biophysical Journal. 2017;112(1):10-5.
71. Scherer MK, Trendelkamp-Schroer B, Paul F, Pérez-Hernández G, Hoffmann M, Plattner N, et al. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. Journal of Chemical Theory and Computation. 2015;11(11):5525-42.
72. Zimmerman MI, Bowman GR. FAST Conformational Searches by Balancing Exploration/Exploitation Trade-Offs. J Chem Theory Comput. 2015;11(12):5747-57.
73. Doerr S, Harvey MJ, Noe F, De Fabritiis G. HTMD: High-Throughput Molecular Dynamics for Molecular Discovery. J Chem Theory Comput. 2016;12(4):1845-52.
74. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12(null):2825–30.
75. Waskom ML. seaborn: statistical data visualization. Journal of Open Source Software. 2021;6(60):3021.
76. Prinz J-H, Keller B, Noé F. Probing molecular kinetics with Markov models: metastable states, transition pathways and spectroscopic observables. Physical Chemistry Chemical Physics. 2011;13(38):16912-27.
77. Schütte C, Noé F, Lu J, Sarich M, Vanden-Eijnden E. Markov state models based on milestoning. The Journal of Chemical Physics. 2011;134(20):204105.
78. Noe F, Wu H, Prinz JH, Plattner N. Projected and hidden Markov models for calculating kinetics and metastable states of complex molecules. J Chem Phys. 2013;139(18):184114.