JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

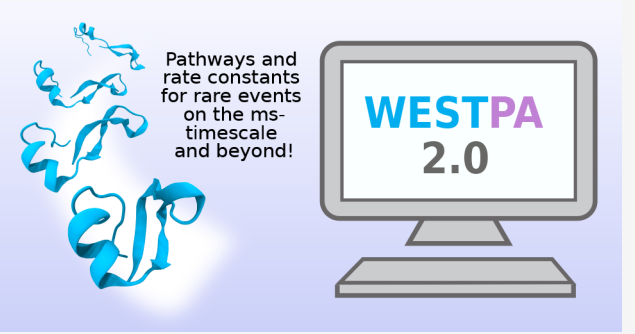
WESTPA 2.0: High-Performance Upgrades for Weighted Ensemble Simulations and Analysis of Longer-Timescale Applications

John D. Russo,[#] She Zhang,[#] Jeremy M. G. Leung,[#] Anthony T. Bogetti,[#] Jeff P. Thompson, Alex J. DeGrave, Paul A. Torrillo, A. J. Pratt, Kim F. Wong, Junchao Xia, Jeremy Copperman, Joshua L. Adelman, Matthew C. Zwier, David N. LeBard, Daniel M. Zuckerman, and Lillian T. Chong*

Cite This: https://doi.org/10.1021/acs.jctc.1c01154 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: The weighted ensemble (WE) family of methods is one of several statistical mechanics-based path sampling strategies that can provide estimates of key observables (rate constants and pathways) using a fraction of the time required by direct simulation methods such as molecular dynamics or discrete-state stochastic algorithms. WE methods oversee numerous parallel trajectories using intermittent overhead operations at fixed time intervals, enabling facile interoperability with any dynamics engine. Here, we report on the major upgrades to the WESTPA software package, an open-source, high-performance framework that implements both basic and recently developed WE methods. These upgrades offer substantial improvements over traditional WE methods. The key features of the new WESTPA 2.0 software enhance the efficiency and ease of use: an adaptive binning scheme for more efficient surmounting of large free energy barriers, streamlined handling of large simulation data sets, exponentially improved analysis of kinetics, and developer-friendly tools for creating new WE methods, including a Python API and resampler module for implementing both binned and "binless" WE strategies.



1. INTRODUCTION

The field of molecular dynamics (MD) simulations of biomolecules arguably is following a trajectory that is typical of mathematical modeling efforts: as scientific knowledge grows, models grow ever more complex and ambitious, rendering them challenging for computation. While early MD simulations focused on single-domain small proteins,[1] modern simulations have attacked ever larger complexes[2,3] and even entire virus particles.[4−7] This trend belies the fact that record-setting small-protein simulations in terms of total simulation time remain limited to the millisecond scale on special-purpose resources[8] and to <100 μs on typical university clusters. These limitations have motivated the development of numerous approaches to accelerate sampling, among which are rigorous path sampling approaches capable of providing unbiased kinetic and mechanistic observables.[9−18]

Our focus is the weighted ensemble (WE) path sampling approach,[17,19] which has helped transform what is feasible for molecular simulations in the generation of pathways for long-timescale processes (>μs) with rigorous kinetics. Among these simulations are notable applications, including atomically detailed simulations of protein folding,[20] coupled protein folding and binding,[21] protein−protein binding,[22] protein−ligand unbinding,[23] and the large-scale opening of the SARS-CoV-2 spike protein.[24] The latter is a significant milestone—both in the system size (half a million atoms) and timescale (seconds).[24] Instrumental to the success of the above applications have been advances in not only WE methods but also software.[24]

Here, we present the next generation (version 2.0) of the most cited, open-source WE software called WESTPA (WE Simulation Toolkit with Parallelization and Analysis).[25] WESTPA 2.0 is designed to further enhance the efficiency of WE simulations with high-performance algorithms for the following: (i) further enhanced sampling via restarting from reweighted trajectories, adaptive binning, and/or binless strategies, (ii) more efficient handling of large simulation data sets, and (iii) analysis tools for the estimation of first passage time (FPT) distributions and for more efficient estimation of rate constants. Similar to its predecessor, WESTPA 2.0 is a highly scalable, portable, and interoperable

Received: November 14, 2021

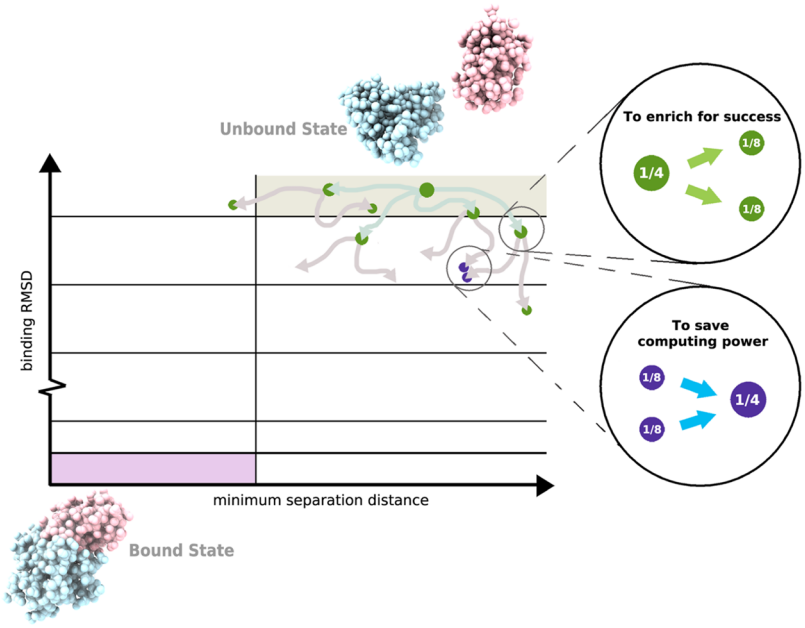


Figure 1. Basic WE protocol. As illustrated for the simulation of a protein−protein binding process, a two-dimensional progress coordinate is divided into bins with the goal of occupying each bin with a target number of four trajectories. Four equally weighted trajectories are initiated from the unbound state and subjected to a resampling procedure at periodic time intervals τ for the following: (i) to enrich for success, trajectories that make transitions to less-visited bins are replicated to generate a target of four trajectories in these bins, splitting the weights evenly among the child trajectories (green spheres) and (ii) to save computing time, the lowest-weight trajectories in bins that have exceeded four trajectories are terminated, merging their weights with those of higher-weight trajectories in these bins (purple spheres). Spheres are sized according to their statistical weights.

Python package that embodies the full range of the WE's capabilities, including a rigorous theory for any type of stochastic dynamics (e.g., MD and Monte Carlo simulations) that is agnostic to the model resolution.[26] In comparison to other open-source WE packages such as accelerated weighted ensemble with a "Work Queue" distributed-computing framework (AWE-WQ)[27] and a weighted ensemble python (wepy) tool,[28] WESTPA is unique in its (i) high scalability with nearly perfect scaling out to thousands of CPU cores[24] and GPUs and (ii) demonstrated ability to interface with a variety of dynamics engines and model resolutions, including atomistic,[22] coarse-grained,[29] whole-cell,[30] and nonspatial system models.[31,32]

After a brief overview of the WE strategy (Section 2), we describe the organization of WESTPA 2.0 (Section 3) and new analysis tools that further expand the capabilities of the software package (Section 4). Together, these features greatly facilitate the execution and analysis of WE simulations of even larger systems and/or slower timescales.

2. OVERVIEW OF THE WE PATH SAMPLING STRATEGY

The WE strategy enhances the sampling of rare events (e.g., protein folding, protein binding, and chemical reactions) by orchestrating the periodic resampling of multiple, parallel trajectories at fixed time intervals τ (Figure 1).[17] The statistically rigorous resampling scheme maintains an even coverage of the configurational space by replicating ("splitting") trajectories that have made transitions to newly visited regions and potentially terminating ("merging") trajectories that have overpopulated previously visited regions. The configurational space is typically defined by a progress coordinate that is divided into bins where an even coverage of this space is defined as a constant number of trajectories occupying each bin; alternatively, trajectories may be grouped by a desired feature for "binless" resampling schemes.[33] Importantly, trajectories are assigned statistical weights that are rigorously tracked during resampling; when trajectories are replicated in a given bin, the weights are split among child trajectories and when trajectories are terminated in a probabilistic fashion, the weights are merged with a continued trajectory of that bin. This rigorous tracking ensures that no bias is introduced into the ensemble dynamics, enabling direct estimates of rate constants.[26]

WE simulations can be run under equilibrium or nonequilibrium steady-state conditions. To maintain nonequilibrium steady-state conditions, trajectories that reach the target state are "recycled" back to the initial state, retaining the same statistical weight.[34] The advantage of equilibrium WE simulations over steady-state WE simulations is that the target state need not be strictly defined in advance since no recycling of trajectories at the target state is applied.[35] On the other hand, steady-state WE simulations have been more efficient in yielding successful pathways and estimates of rate constants. Equilibrium observables can be estimated from either equilibrium WE simulations or the combination of two nonequilibrium steady-state WE simulations in the opposite directions when the historical information is taken into account.[35]

3. ORGANIZATION OF WESTPA 2.0

Below, we present the organization of WESTPA 2.0, beginning with code reorganization to facilitate software development (Section 3.1) and then proceeding to a description of a Python application programming interface (API) for setting up, running, and analyzing WE simulations (Section 3.2); a minimal adaptive binning (MAB) mapper (Section 3.3); a

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

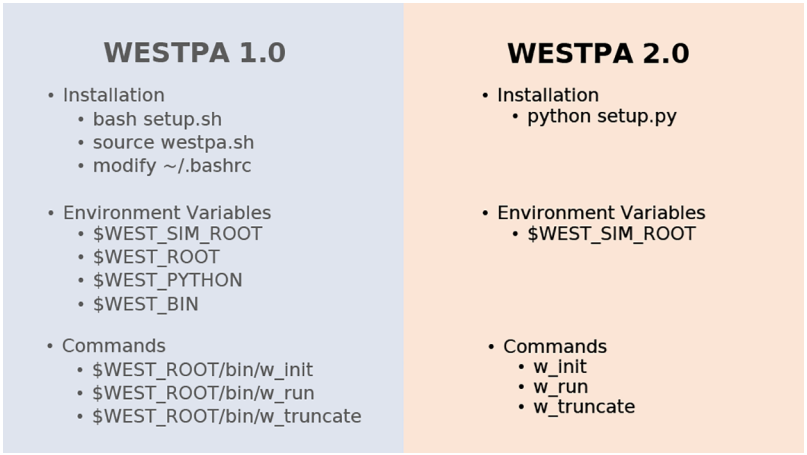


Figure 2. Reorganization of WESTPA 1.0 to WESTPA 2.0. In version 2.0, WESTPA is installed using Python and relies on only a single environment variable such that commands can be called directly through Python. To reflect these changes, we have updated our original suite of WESTPA tutorials for version 2.0 (https://github.com/westpa/westpa_tutorials/tree/westpa-2.0-restruct).[36,37]

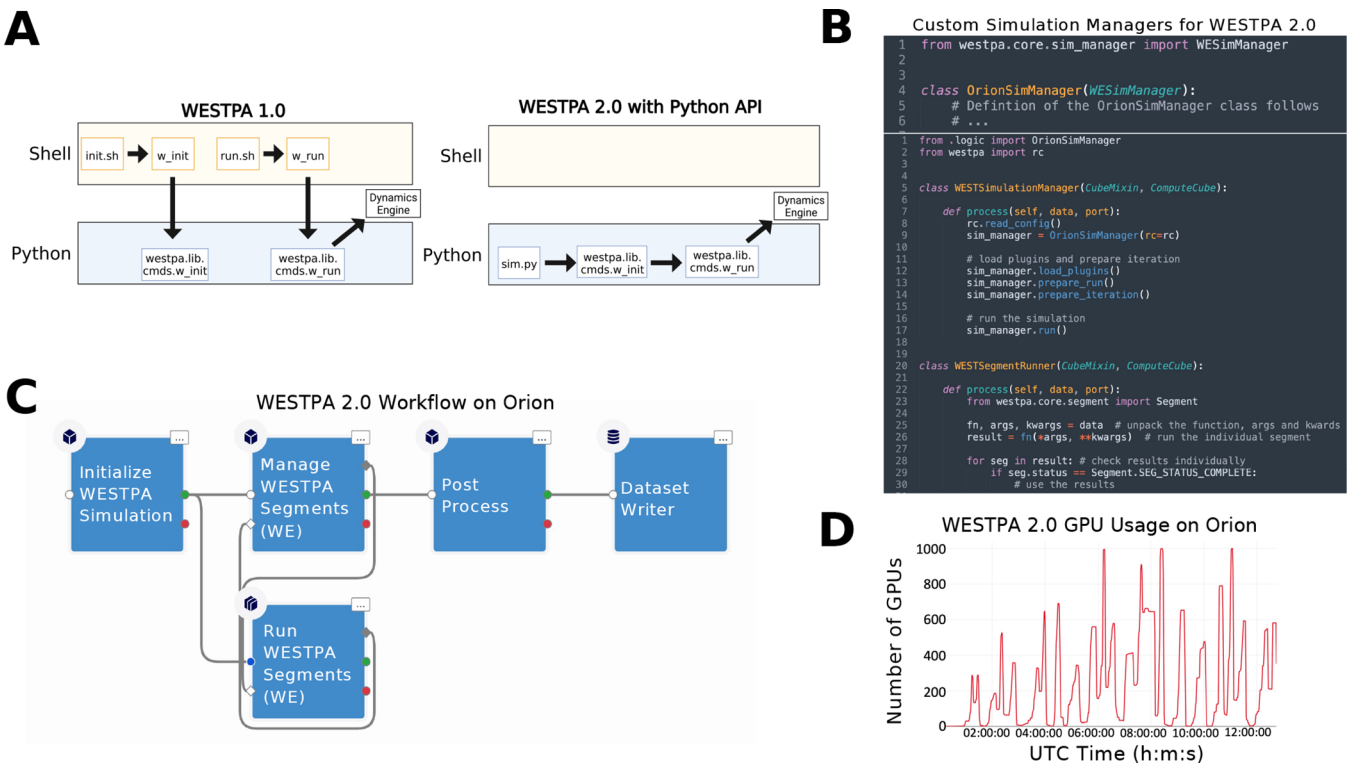


Figure 3. Comparison of workflows for setting up and running WE simulations using WESTPA 1.0 and 2.0, a demonstration of using the Python API for WESTPA 2.0, and GPU performance of the updated API within a cloud computing environment. (A) The Python API in WESTPA 2.0 enables a user to fully define, initialize, and run a WESTPA simulation from within a single Python script (right panel), without needing to invoke command line utilities required in WESTPA 1.0 (left panel). For backward compatibility, all original functionality provided in version 1.0 for invoking WESTPA (e.g., `w_init` and `w_run` tools) via shell scripts remains available in WESTPA 2.0. (B) Example of defining a custom simulation manager with the WESTPA 2.0 API (top panel) and using the newly defined simulation manager and WESTPA 2.0 API to programmatically control and run a WE simulation (bottom panel). Here, the `WESTSimulationManager` class sends work to the `WESTSegmentRunner` class, which unpacks and runs the scripts specified from the WESTPA config file (`west.cfg`). (C) Example workflow diagram from the Orion user interface using the Python classes constructed from the internal WESTPA APIs presented in Figure 3B. Here, a kernel (Initialize WESTPA Simulation) initializes both the `WESTSimulationManager` (Manage WESTPA Segments) and the `WEST-SimulationRunner` (Run WESTPA Segments) kernels from Figure 3B, which are connected in a cycle to manage splitting and merging. Finally, all data are exported through a Post Process and Dataset Writer kernel for final data processing and storage. (D) Performance of the WESTPA 2.0 API using the `WESTSimulationRunner` class from Figure 3B within an Amazon Web Services environment using a combination of numerous g4dn instances as a function of the wall clock time in Universal Coordinated Time (UTC) units. Here, the per-iteration scaling reaches thousands of GPUs in just under a few hours for a test system of butanol crossing a neat 1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine (POPC) membrane bilayer using the WESTPA 2.0 API with the OpenMM 7.5 MD engine.[41]

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

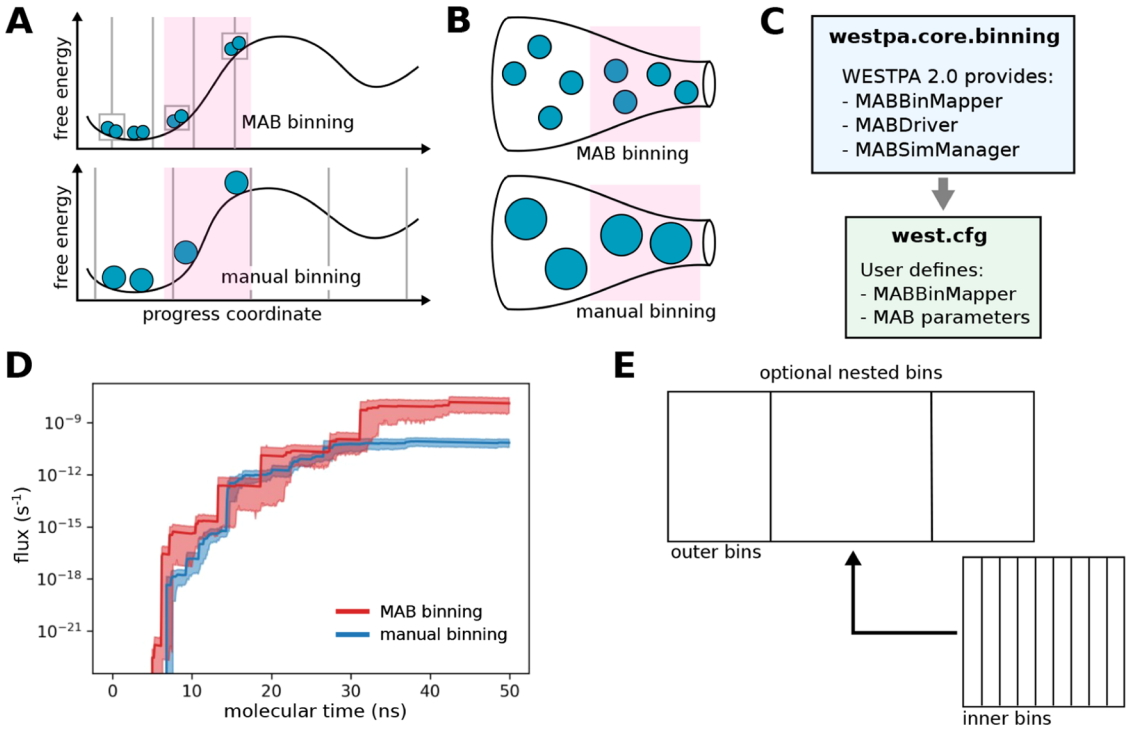


Figure 4. The MAB scheme is more efficient in surmounting free energy barriers than manual fixed binning schemes. (A) Bin positions and trajectories after replication using the MAB scheme vs a manual binning scheme with the same positions of trajectories (blue circles, sized according to statistical weights) along a chosen progress coordinate and a target of two trajectories per bin. The MAB scheme adaptively positions bins along the progress coordinate by placing equally spaced bins (in this case, three bins, as indicated by solid vertical lines) between the positions of the trailing and leading trajectories along with separate bins (boxes) for these trajectories and a third trajectory in a bottleneck region (pink) along the free energy barrier. (B) Enlarged "bottle" diagrams highlighting the bottleneck region (pink) along with the relative positions and weights of trajectories for the MAB and manual binning schemes in panel (A). In contrast to the manual binning scheme where trajectories may stall in a bottleneck region, the MAB scheme automatically detects trajectories in this region, replicating these trajectories to enrich for success in surmounting the barrier. (C) MAB scheme options in the `westpa.core.binning` module and the corresponding user-defined options in the `west.cfg` file. (D) Flux of a drug-like molecule (tacrine) permeating through a neat POPC membrane as a function of the molecular time using fixed binning (blue) or adaptive binning (MAB scheme) (red). Solid lines represent mean fluxes, and the shaded regions represent 95% confidence intervals. The molecular time is defined as $N\tau$, where N is the number of WE iterations and τ is the fixed time interval (100 ps) of each WE iteration. Simulations were run using WESTPA 2.0 and OpenMM 7.5 MD engine.41 (E) Schematic of a simple recursive binning case in which closely spaced inner bins are "nested" within a wider outer bin.

generalized resampler module that enables the implementation of both binned and binless schemes (Section 3.4); and an HDF5 framework for more efficient handling of large simulation data sets (Section 3.5).

3.1. Code Reorganization to Facilitate Software Development. The WESTPA 2.0 software is designed to facilitate the maintenance and further development of the software according to the established and emerging best practices for Python development and packaging. The code has been consolidated and reorganized to better indicate the role of each module (Figure 2). The software can now be installed as a standard Python package using pip or by running `setup.py`. The package will continue to be available through Conda via conda-forge, which streamlines the installation process by enabling WESTPA and all software dependencies to be installed at the same time. We have implemented automated GitHub Actions for continuous integration testing and code quality checks using the Black Python code formatter as a precommit hook, alongside flake8 for nonstyle linting. Templates are provided for GitHub issues and pull requests. Both the user's and developer's guides are available on the GitHub wiki along with the Sphinx documentation of key functions with autogenerated docstrings. Further support will continue to be provided through WESTPA users' and developers' email lists hosted on Google Groups (linked on https://westpa.github.io).

3.2. Python API for Setting up, Running, and Analysis of WE Simulations. To simplify the process of setting up and running WE simulations, WESTPA 2.0 features a Python API that enables the user to execute the relevant commands within a single Python script instead of invoking a series of command-line tools, as previously done in WESTPA 1.0 (Figure 3A). This also provides tools for third-party developers to build and develop WESTPA-based applications and plugins, for example, the integration of WESTPA into the cloud-based computing platform, OpenEye Scientific's Orion,[38,39] or the history-augmented Markov state model (haMSM) restarting plugin (Section 4.2), which uses the results of a WESTPA simulation to perform a steady-state analysis then restart the simulation based on the results of that analysis.

Figure 3B provides an example of how to programmatically call the WESTPA 2.0 API from the Orion cloud platform, which could in principle be any Python script within any supercomputing or personal computing environment. First, a developer can write any custom simulation or work manager of their choice by subclassing or completely rewriting core WESTPA components (top panel). Second, a workflow can be constructed by invoking a simple set of WESTPA 2.0 Python

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

commands to perform any WE simulation (bottom panel). Typically, a user of the WESTPA 2.0 Python API only needs a handful of API endpoints to perform a complicated simulation protocol. As an example of the power of the simplicity of the Python API, we demonstrate how a workflow can be constructed from the defined workflow kernels (Figure 3C) and show the GPU performance over wall-clock time (in Coordinated Universal Time; UTC) from a drug-like molecule in a membrane permeability simulation (Figure 3D). Using the internal API, a user's simulation can request large amounts of computational resources per iteration. In this case, thousands of GPUs are requested per WE iteration for a simulation of butanol crossing a natural membrane mimetic system (https://github.com/westpa/westpa2_tutorials).[40]

To facilitate the development of custom analysis workflows in cases where more flexibility is required than that of the existing `w_ipa` analysis tool,[36] WESTPA 2.0 includes the new `westpa.analysis` Python API. This API provides a high-level view of the data contained in the main WESTPA HDF5 file (`west.h5`) and facilitates retrieval of trajectory data, reducing the overhead of writing custom analysis code in Python and performing quick, interactive analysis of individual trajectories (or walkers). The `westpa.analysis` API is built on three core data types: *run*, *iteration*, and *walker*. A *run* is a sequence of iterations; an *iteration* is a collection of *walkers*. Key instance data can be accessed via attributes and methods. For example, a *walker* has attributes such as the statistical weight (`weight`), progress coordinate values (`pcoords`), starting conformation (`parent`), and child trajectories after replication (`children`) as well as a method, `trace`, to trace its history (as a pure Python alternative to the `w_trace` tool). The API also provides facilities for retrieving and concatenating trajectory segments. These include support for (i) type-aware concatenation of trajectory segments represented by NumPy arrays or MDTraj trajectories, (ii) use of multiple threads to potentially increase performance when segment retrieval is an I/O bound operation, and (iii) display of progress bars. Finally, the API provides a convenience function, `time_average`, for computing the time average of an observable over a sequence of *iterations* (e.g., all or part of a run).

3.3. MAB Mapper. To automate the placement of bins along a chosen progress coordinate during WE simulation, we have implemented the MAB scheme[42] as an option in the `westpa.core.binning` module. The MAB scheme positions a specified number of bins along a progress coordinate after each resampling interval τ by (1) tagging the positions of the trailing and leading trajectories along the progress coordinate and evenly placing a specified number of bins between these positions and (2) tagging "bottleneck" trajectories positioned on the steepest probability gradients and assigning these trajectories to their own bins (Figure 4A,B). Despite its simplicity, the MAB scheme requires less computing time than manual, fixed binning schemes in surmounting large free energy barriers, resulting in more efficient conformational sampling and estimation of rate constants.[42] To apply the MAB scheme, users specify the `MABBinMapper` option along with accompanying parameters such as the number of bins in the `west.cfg` file (Figure 4C).

Figure 4D illustrates the effectiveness of the MAB scheme in enhancing the efficiency of simulating the membrane permeability of a drug-like molecule (tacrine). Relative to a fixed binning scheme, the MAB scheme results in an earlier flux of tacrine through a model cellular membrane bilayer (~5 vs ~7 ns), and this flux increases more quickly, achieving values that are 2 orders of magnitude higher for the duration of the test.

The MAB scheme provides a general framework for the user creation of more complex adaptive binning schemes.[42] Users can now specify nested binning schemes in the `west.cfg` file (Figure 4E). To run WESTPA simulations under nonequilibrium steady-state conditions (i.e., with the "recycling" of trajectories that reach the target state) with the MAB scheme, users can nest a `MABBinMapper` inside of a `RecursiveBinMapper` bin and specify a target state as the outer bins. Multiple individual `MABBinMappers` can be created and placed at different locations of the outer bins using a recursive scheme, offering further flexibility in the creation of advanced binning schemes.

3.4. Generalized Resampler Module that Enables Binless Schemes. In the original (default) WE resampling scheme, trajectories are split and merged based on a predefined set of bins.[17] In WESTPA 2.0, we introduce a generalized resampler module that enables the users to implement both binned and "binless" resampling schemes, providing the flexibility to resample trajectories based on a property of interest by defining a grouping function. While grouping on the state last visited (e.g., initial or target state) was previously possible using the binning machinery in WESTPA 1.0,[43] our new resampler module provides a more general framework for creating binless schemes by defining a group/reward function of interest; such schemes enable the use of nonlinear progress coordinates that may be identified by machine learning techniques. Following others,[44] the resampler module includes options for (i) specifying a minimum threshold for trajectory weights to avoid running trajectories with inconsequentially low weights and (ii) specifying a maximum threshold for trajectory weights to avoid a single large-weight trajectory from dominating the sampling, increasing the number of uncorrelated successful events that reach the target state.

As illustrated in Figure 5, the implementation of a binless scheme requires two modifications to the default WESTPA simulation: (i) a user-provided group module containing the methods needed to process the resampling property of interest for each trajectory walker, and (ii) updates to the `west.cfg` file specifying the resampling method in the `group_function` keyword and the attribute in the `group_arguments` keyword.

We provide two examples of implementing binless schemes in the westpa-2.0-restruct branch of the WESTPA_Tutorials GitHub repository (https://github.com/westpa/westpa_tutorials/tree/westpa-2.0-restruct).[37] The `basic_nacl_group_by_history` example illustrates the grouping of the trajectory based on its "history", that is, a shared parent N WE iterations back. The parameter N is specified in the keyword `hist_length` under the `group_arguments` keyword in the `west.cfg` file. This WESTPA configuration file also specifies the name of the grouping function method, `group.walkers_by_history`, in the `group_function` keyword. In the `basic_nacl_group_by_color` example, trajectory walkers are tagged based on "color" according to the state last visited. Only walkers that have the same color are merged, thereby increasing the sampling of pathways in both directions.

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

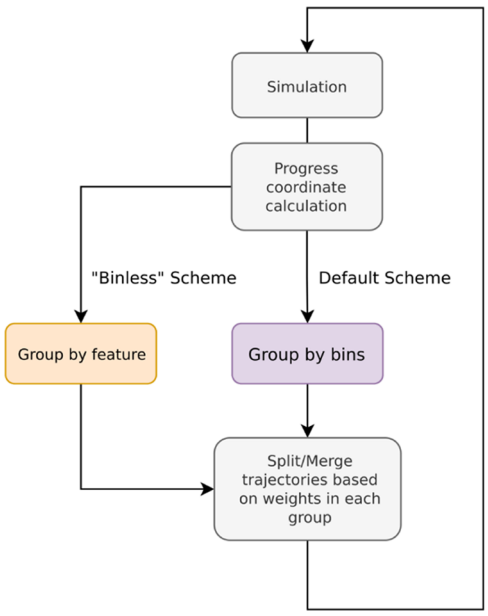


Figure 5. Flowchart for implementing binless resampling schemes in WESTPA 2.0. The implementation involves grouping trajectories by feature (using the `group_function` keyword defined in the group module) before splitting and merging. The functionality for positioning bins along a chosen progress coordinate remains available in WESTPA 2.0.

State definitions are declared within the `group_arguments` keyword in the `west.cfg` file.

3.5. HDF5 Framework for More Efficient Handling of Large Simulation Data Sets. One major challenge of running WE simulations has been the management of the resulting large data sets, which can amount to tens of terabytes over millions of trajectory files. To address this challenge, we have developed a framework for storing the trajectory data in a highly compressed and portable HDF5 file format. The format is derived from the HDFReporter class implemented in the MDTraj analysis suite[45] and maintains compatibility with NGLView,[46] an iPython/Jupyter widget for the interactive viewing of molecular structures and trajectories. A single HDF5 file is generated per WE iteration, which includes a link to each trajectory file stored in the main WESTPA data file (`west.h5`). Thus, the new HDF5 framework in WESTPA 2.0 enables users to restart a WE simulation from a single HDF5 file rather than millions of trajectory files and simplifies data sharing as well as analysis. The dramatic reduction in the number of trajectory files also eliminates a potentially large overhead from the file system that results from the storage of numerous small files. For example, a 53% overhead has been observed for a 7.5-GB data set of 103,260 trajectory files generated from NTL9 protein folding simulations (Figure 9), occupying 11.5 GB of actual disk storage on a Lustre file system.

To test the effectiveness of the HDF5 framework in reducing the amount of data storage required for WE simulations, we applied the framework to a set of three independent WE simulations of Na^+/Cl^- association and one WE simulation involving p53 peptide conformational sampling (Figure 6A,B). Our results revealed 27 and 85% average reduction in the total size of trajectory files generated during the Na^+/Cl^- association and p53 peptide simulations, respectively, relative to that obtained using WESTPA 1.0. Given a fixed number of bins, the sizes of per-iteration HDF5 files were also shown to converge as the simulation progresses (Figure 6C,D), suggesting that the storage of trajectory data by iteration not only facilitates the management of the data but also yields files of roughly equal sizes. The difference in the reduction efficiency that we observed between the Na^+/Cl^- and p53 peptide systems can be attributed to differences in the simulation configurations including the format of the output trajectories, restart files, and other factors such as the verbosity of logging.

Our tests revealed that the additional steps introduced by the HDF5 framework for managing the trajectory coordinate and restart files did not have any significant impact on the WESTPA runtime (Figure 6E), which is normalized by the number of trajectory segments per WE iteration given that the evolution of bin occupancies by trajectories can vary among different runs due to the stochastic nature of the MD simulations (after 60 iterations, the WESTPA 1.0 run occupied six more bins than the WESTPA 2.0/HDF5 run). This variation in the bin occupancy is unlikely to be affected by the HDF5 framework since it simply manages the trajectory and restart files and does not alter how the system is simulated. The differences in bin occupancies/total number of trajectories may also partially contribute to the large reduction in the per-iteration file sizes for the HDF5 run observed in Figure 6D for the p53 peptide. However, the majority of this file size reduction results from efficient HDF5 data compression of trajectory coordinate, restart, and log files. Finally, the trajectory data saved in the HDF5 files can be extracted and analyzed easily using MDTraj in combination with our new analysis framework presented in Section 3.2 (Figure 6F).

4. ANALYSIS TOOLS

WESTPA 2.0 features new analysis tools for estimating rate constants more efficiently using the distribution of "barrier crossing" times (Section 4.1), accelerating the convergence using a haMSM to reweight trajectories (Section 4.2) and estimating the distribution of FPTs (Section 4.3).

4.1. RED Scheme for Rate Constant Estimation. To more efficiently estimate the rate constants from WE simulations, we have implemented the rates from event durations (RED) scheme as an analysis tool called w_red in the WESTPA 2.0 software. The RED scheme exploits the transient ramp-up portion of a WE simulation by incorporating the probability distribution of event durations (or "barrier crossing" times) from a WE simulation as part of a correction factor (Figure 7A).[48] The correction factor accounts for the systematic error that results from the statistical bias toward the observation of events with short durations and reweights the event duration distribution accordingly. When applied to an atomistic WE simulation of a protein−protein binding process, the RED scheme is >25% more efficient than the original WE scheme[17] in estimating the association rate constant (Figure 7B).[48]

The code for estimating the rate constants using the RED scheme takes as an input the `assign.h5` files and `direct.h5` files generated by the `w_ipa` analysis tool. Users then specify in the analysis section of the `west.cfg` file that analysis scheme `w_red` should analyze along with the initial/final states and the number of frames per iteration. Executing `w_red` from the command line will output the corrected flux estimates as a new data set called `red_flux_evolution` to the users' existing `direct.h5`

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

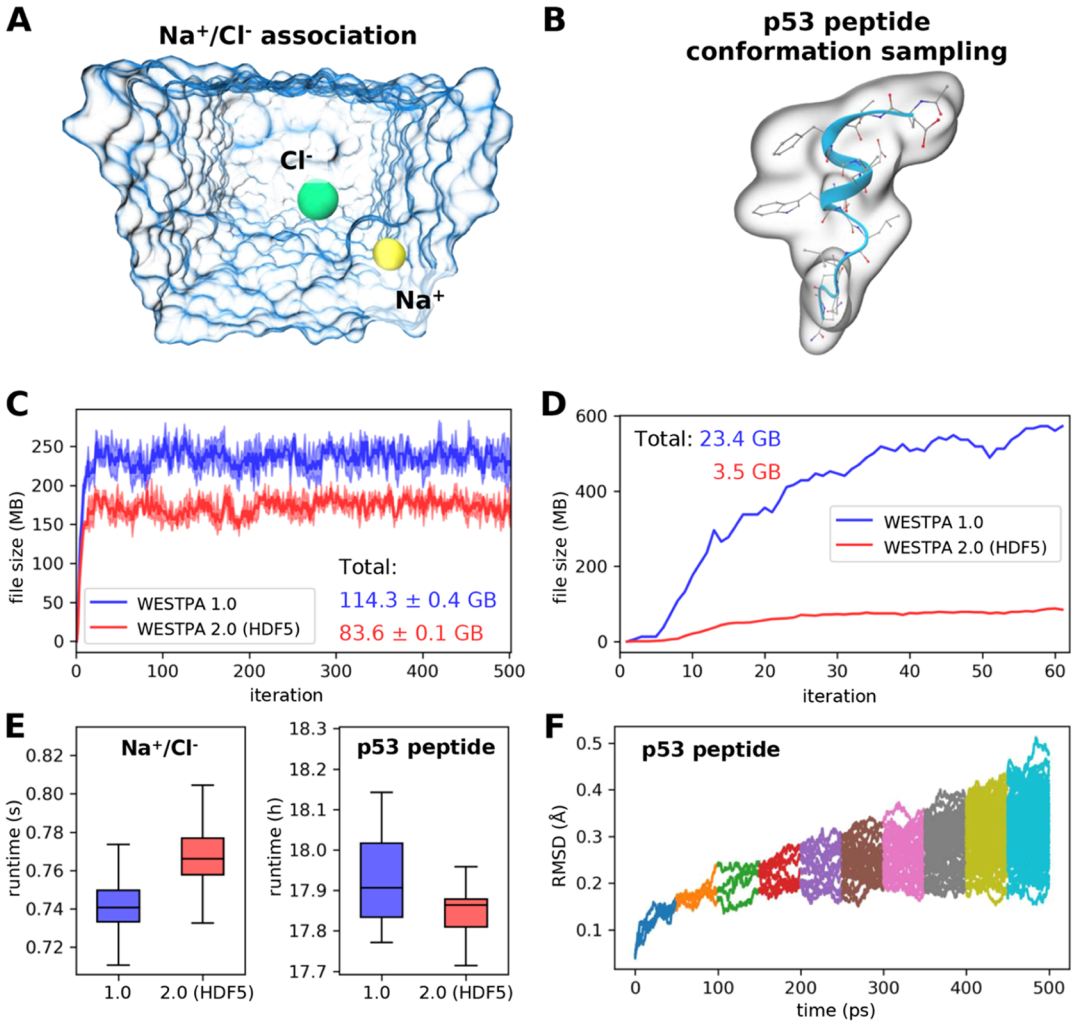


Figure 6. Demonstration of the usage of the HDF5 framework for two example systems. (A) Na^+/Cl^- association simulation where Na^+ (yellow sphere) and Cl^- (green sphere) ions were solvated in explicit water (blue transparent surface). The distance between the two ions serves as the progress coordinate. (B) Conformational sampling of a p53 peptide (residues 17−29) in a generalized Born implicit solvent using a progress coordinate consisting of the heavy-atom root mean square deviation (rmsd) of the peptide from its MDM2-bound conformation.[21] The molecular surface of the p53 peptide is rendered as a transparent surface, with both the secondary (blue ribbon) and atomic structures overlaid. (C) Comparison of file sizes of per-iteration HDF5 files for the Na^+/Cl^- association simulation as a function of the WE iteration using WESTPA 1.0 and 2.0 with the HDF5 framework. The result was obtained from three independent simulations where the solid curves show the mean file sizes, while the light bands show the standard deviations. (D) Same comparison as panel (C) for a single simulation of the p53 peptide; hence, no error bars are shown. (E) Comparison of wall-clock runtimes normalized by the number of trajectory segments per WE iteration using WESTPA 1.0 and 2.0 with the HDF5 framework option turned on. (F) Time evolution of the heavy-atom rmsd of the p53 peptide from its MDM2-bound conformation using trajectories obtained using WESTPA's analysis tools. Colors represent rmsds obtained from different iterations. WESTPA simulations of Na^+/Cl^- association and the p53 peptide were run using the OpenMM 7.5 MD engine.[41]

file (Figure 7C). The RED rate constant estimates can then be accessed through the Python `h5py` module and plotted versus time to assess the convergence of the estimates. To estimate the uncertainties in observables calculated from a small number of trials (i.e., the number of independent WE simulations), we recommend using the Bayesian bootstrap approach.[17,49] If it is not feasible to run multiple independent simulations with a certain system due to either the system size or the timescale of the process of interest, a user can apply a Monte Carlo bootstrapping approach to a single simulation's RED rate constant estimate.

4.2. haMSM Restarting Plugin. haMSMs provide a powerful tool for the estimation of stationary distributions and rate constants from transient, unconverged WE data.[50] Thus, the approach has a similar motivation to the RED scheme.[48] In haMSM analysis, instead of discretizing trajectories via the WE bins used by WESTPA, as in the `WESS` and `WEED` reweighting plugins for a non-equilibrium steady state and equilibrium state, respectively,[34,35] a much finer and more numerous set of "microbins" is employed to calculate the steady-state properties with a higher accuracy. These estimates, in turn, can be used to start new WE simulations from a steady-state estimate, accelerating the convergence of the simulation.[49] The new plugin provides a streamlined implementation of the restarting protocol that runs automatically as part of a WESTPA simulation, a capability which did not previously exist.

The `msm_we` package provides a set of analysis tools for using typical WESTPA HDF5 output files, augmented with atomic coordinates, to construct an haMSM. A nearly typical MSM model-building procedure[52] is used (Figure 8): WE

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

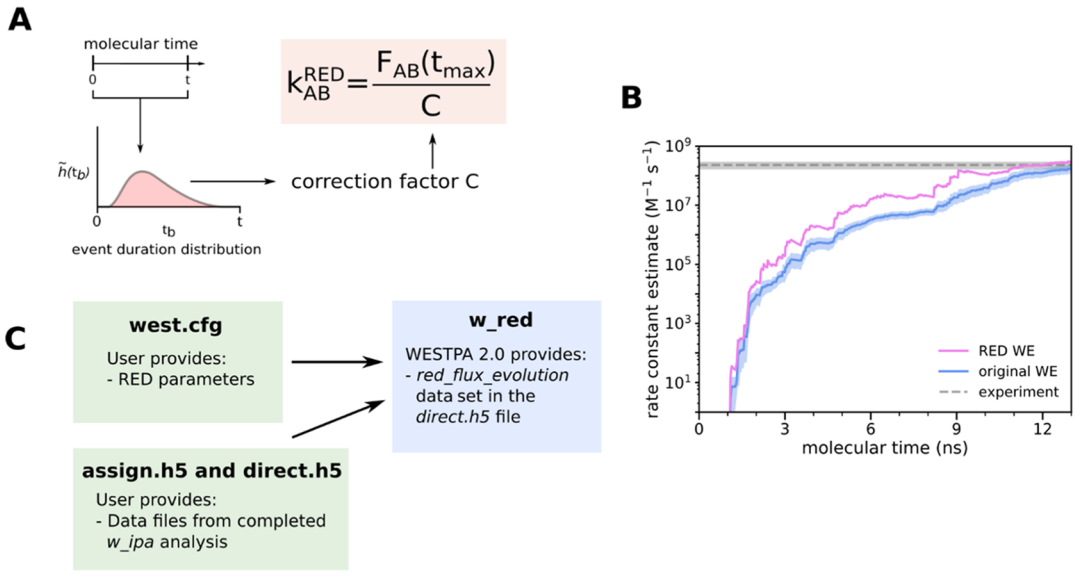


Figure 7. The RED scheme for more efficient rate constant estimation. (A) Schematic illustrating the RED scheme, which incorporates the distribution of event durations as part of a correction factor for rate constant estimates that account for the statistical bias toward the observation of events with short durations. (B) Application of the original and RED schemes to estimate the associate rate constant of a protein−protein binding process involving the barnase and barstar proteins as a function of the molecular time in a WE simulation. The molecular time is defined as $N\tau$, where N is the number of WE iterations and τ is the fixed time interval (20 ps) of each WE iteration. Simulations were previously run using WESTPA 1.0 with the GROMACS 4.6.7 MD engine.[47] (C) Schematic illustrating how users can generate a data set for calculating the RED scheme correction factor from the simulation data stored in the analysis HDF5 files and apply the correction factor to the rate constant estimate using the new `w_red` tool.

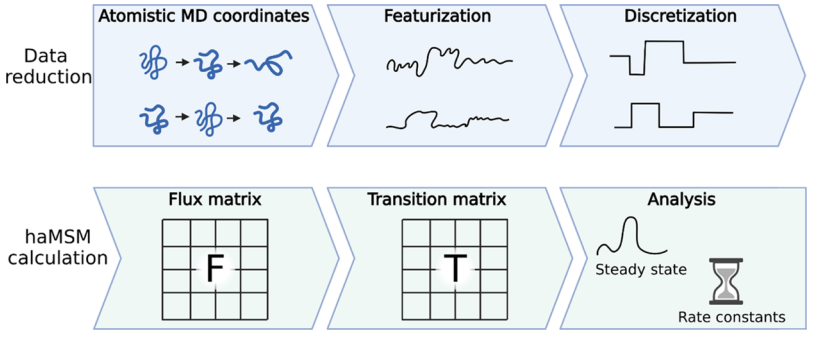


Figure 8. Workflow for constructing an haMSM from trajectories. First, the atomistic trajectories are featurized and discretized. The flux matrix is then computed by computing fluxes between discrete states. The flux matrix is row-normalized into a transition matrix. Estimates of steady-state populations and rate constants are obtained from the analysis of the transition matrix.[51]

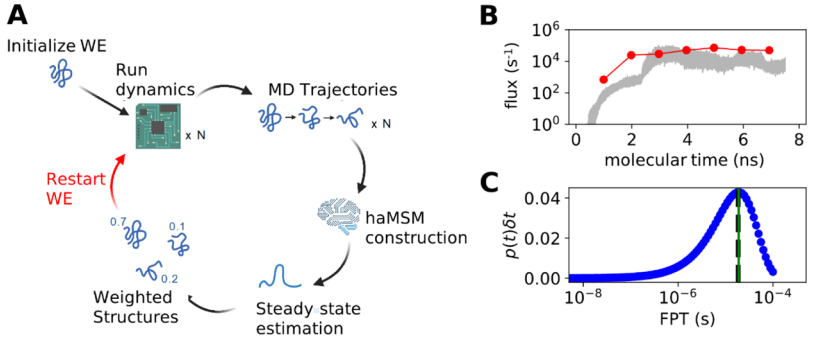


Figure 9. Application of the haMSM restarting plugin to the ms folding process of the NTL9 protein. (A) Diagram of the haMSM restarting plugin's functionality. (B) Example of the restarting plugin functionality in the accelerated convergence of NTL9 folding rate constants from a WESTPA 2.0 simulation using the AMBER 16 MD engine.[53] haMSM estimates at restarting points are shown as dots, WE direct fluxes are shown as red lines, and the 95% credibility region from the direct WE is shown in gray. (C) Distribution of the FPTs for NTL9 folding from the haMSM built at the final restart of the simulation in Figure 9B. The weighted average of the blue FPT distribution is shown in black dashed lines, and the MFPT estimate from the haMSM's steady-state estimate is shown in green.[51]

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

trajectories are discretized into clusters (microbins) and transitions among microbins are analyzed. However, instead of reconstructing entire trajectories, the `msm_we` analysis computes the flux matrix by taking each weighted parent/child segment pair, extracting and discretizing one frame from each, and measuring the flux between them—that is, the weight is transferred.

The haMSM restarting plugin in WESTPA 2.0 makes use of the analysis tools provided by `msm_we` to incorporate this functionality directly into WESTPA. It manages running a number of independent simulations, initialized from some starting configuration, and augments their output HDF5 with the necessary atomic coordinates. Data from all independent runs are gathered and used to build a single haMSM. Stationary probability distributions and rate constants are estimated from this haMSM.

This plugin can be used to start a set of new WE simulation runs, initialized closer to the steady state (Figure 9). The haMSM and the WE trajectory data are used to build a library of structures and their associated steady-state weights. These are used to initiate a new set of independent WE runs, which should start closer to the steady state and thus converge more quickly. The process can be repeated iteratively, as shown in Figure 9A. The result of this restarting procedure is shown in Figure 9B. For challenging systems, the quality of the haMSM will greatly affect the quality of the steady-state estimate. A further report is forthcoming on strategies for building high-quality haMSMs.

To use this plugin, users must specify a function that ingests coordinate data and featurizes the data. Dimensionality reduction may be performed on this featurized data. An effective choice of featurization provides a more granular structural description of the system without including a large number of irrelevant coordinates that add noise without adding useful information. For example, a limited subset of the full atoms such as only α-carbons or even a strided selection of the α-carbons, may be sufficient to capture the important structural information. Choosing the featurization based on rotation-invariant distances, such as pairwise atomic distances instead of atomic positions, can also help capture the structural fluctuations without sensitivity to large-scale motion of the entire system.

To validate the convergence of the restarted simulations, a number of independent replicates of the restarting protocol should be performed. These replicates should demonstrate both the stability in flux estimates across restarts and relatively constant-in-time direct fluxes within the restarts. If limited to a single replicate, the agreement between the haMSM flux estimate and the direct flux should also be validated.

4.3. Estimating FPT Distributions. FPTs and their mean values (MFPTs) are key kinetics quantities to characterize many stochastic processes (from a macrostate to another) in chemistry and biophysics such as chemical reactions, ligand binding and unbinding, protein folding, and diffusion processes of small molecules within crowded environments. WE simulations, via the Hill relation, provide unbiased estimates of the MFPT directly once the steady state is reached[34] or indirectly via non-Markovian haMSM analysis,[35] but the mathematically rigorous estimation of the FPT distribution is not available and has been a challenge for WE simulation. Suárez and coworkers, however, have shown that the FPT distributions estimated from haMSM models provide semi-quantitative agreement with unbiased reference distributions in different systems.[54] Details on building haMSMs are described above in Section 4.2, and more information can be found in the refs 35 and 54.

Here, we extend and strengthen the earlier FPT distribution analysis from WE data. The original code for calculating the FPT distribution was published on a separate GitHub repository (https://github.com/ZuckermanLab/NMpathAnalysis).[55] Recently, we reorganized and refactored the code in class hierarchical structures: a base class (`MatrixFPT`) for calculating MFPT and FPT distributions using a general transition matrix as an input parameter and two derived classes (`MarkovFPT` and `NonMarkovFPT`) using transition matrices from Markovian analysis and non-Markovian analysis, respectively, as mentioned in the haMSM in Section 4.2. The updated code has been merged into the `msm_we` package described in Section 4.2 along with some updates on building a transition matrix from classic MD simulation trajectories.

The new code enables the robust estimation of the FPT distribution. Figure 9C shows the non-Markovian estimation of the FPT distribution of transitions between macrostates A and B from the WE simulation of NTL9 protein folding.

5. SUMMARY

WESTPA is an open-source, highly scalable, interoperable software package for applying the WE strategy, which greatly increases the efficiency of simulating rare events (e.g., protein folding and protein binding) while maintaining rigorous kinetics. The latest WESTPA release (version 2.0) is a substantial upgrade from the original software with high-performance algorithms enabling the simulation of ever more complex systems and processes and implementing new analysis tools. WESTPA 2.0 has also been reorganized into a more standard Python package to facilitate the code development of new WE algorithms, including binless strategies. With these features available in the WESTPA toolbox, the WE community is well-poised to take advantage of the latest strategies for tackling major challenges in rare-event sampling, including the identification of slow coordinates using machine learning techniques,[56,57] and the interfacing of the WE strategy with other software involving complementary rare-event sampling strategies (e.g., OpenPathSampling,[58,59] SAFFIRE,[60] and ScMile[61]) and analysis tools (e.g., LOOS,[62,63] MDAnalysis,[64,65] and PyEmma[66]). WESTPA has also been interfaced with OpenEye Scientific's Orion platform[39] on the Amazon Web Services cloud computing facility. We hope that the above new features of WESTPA will greatly facilitate the efforts by the scientific community to tackle grand challenges in the simulation of rare events in a variety of fields, including the molecular sciences and systems biology.

AUTHOR INFORMATION

Corresponding Author

Lillian T. Chong − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States;* orcid.org/0000-0002-0590-483X; Email: ltchong@pitt.edu

Authors

John D. Russo − *Department of Biomedical Engineering, Oregon Health and Science University, Portland, Oregon 97239-3098, United States*

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

She Zhang − *OpenEye Scientific, Santa Fe, New Mexico 87508, United States;* orcid.org/0000-0001-9265-4498
Jeremy M. G. Leung − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States;* orcid.org/0000-0001-7021-4619
Anthony T. Bogetti − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States;* orcid.org/0000-0003-0610-2879
Jeff P. Thompson − *OpenEye Scientific, Santa Fe, New Mexico 87508, United States*
Alex J. DeGrave − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States;* Present Address: Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA
Paul A. Torrillo − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States;* Present Address: Massachusetts Institute of Technology, Cambridge, MA.
A. J. Pratt − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States*
Kim F. Wong − *Department of Chemistry, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States*
Junchao Xia − *OpenEye Scientific, Santa Fe, New Mexico 87508, United States*
Jeremy Copperman − *Department of Biomedical Engineering, Oregon Health and Science University, Portland, Oregon 97239-3098, United States;* orcid.org/0000-0002-5202-0690
Joshua L. Adelman − *Department of Biological Sciences, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States*
Matthew C. Zwier − *Department of Chemistry, Drake University, Des Moines, Iowa 50311-4505, United States;* orcid.org/0000-0002-0744-1146
David N. LeBard − *OpenEye Scientific, Santa Fe, New Mexico 87508, United States*
Daniel M. Zuckerman − *Department of Biomedical Engineering, Oregon Health and Science University, Portland, Oregon 97239-3098, United States;* orcid.org/0000-0001-7662-2031

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.1c01154

Author Contributions

[#]J.D.R., S.Z., J.M.G.L., and A.T.B. contributed equally.

Notes

The authors declare the following competing financial interest(s): L.T.C. is a current member of the Scientific Advisory Board of OpenEye Scientific and an Open Science Fellow with Roivant Sciences. S.Z., J.P.T., J.X., and D.N.L. are employees of OpenEye Scientific.

■ ACKNOWLEDGMENTS

This work was supported by an NIH grant (R01 GM115805) to L.T.C. and D.M.Z.; NSF grants (CHE-1807301 and MCB-2112871) to L.T.C.; a MolSSI Software Fellowship to J.D.R.; and a University of Pittsburgh Andrew Mellon Graduate Fellowship to A.T.B. Computational resources were provided by the University of Pittsburgh's Center for Research Computing, by OpenEye Scientific via compute instances sourced from Amazon Web Services, and by the Advanced Computing Center at Oregon Health and Science University. We thank David Aristoff, Gideon Simpson, Forrest York, Darian Yang, Surl-Hee Ahn and Alan Grossfield for helpful discussions.

■ REFERENCES

(1) McCammon, J. A.; Gelin, B. R.; Karplus, M. Dynamics of Folded Proteins. *Nature* **1977**, *267*, 585−590.
(2) Casalino, L.; Gaieb, Z.; Goldsmith, J. A.; Hjorth, C. K.; Dommer, A. C.; Harbison, A. M.; Fogarty, C. A.; Barros, E. P.; Taylor, B. C.; McLellan, J. S.; Fadda, E.; Amaro, R. E. Beyond Shielding: The Roles of Glycans in the SARS-CoV-2 Spike Protein. *ACS Cent. Sci.* **2020**, *6*, 1722−1734.
(3) Anandakrishnan, R.; Zhang, Z.; Donovan-Maiye, R.; Zuckerman, D. M. Biophysical Comparison of ATP Synthesis Mechanisms Shows a Kinetic Advantage for the Rotary Process. *Proc. Natl. Acad. Sci. U.S.A.* **2016**, *113*, 11220−11225.
(4) Zhao, G.; Perilla, J. R.; Yufenyuy, E. L.; Meng, X.; Chen, B.; Ning, J.; Ahn, J.; Gronenborn, A. M.; Schulten, K.; Aiken, C.; Zhang, P. Mature HIV-1 Capsid Structure by Cryo-Electron Microscopy and All-Atom Molecular Dynamics. *Nature* **2013**, *497*, 643−646.
(5) Perilla, J. R.; Schulten, K. Physical Properties of the HIV-1 Capsid from All-Atom Molecular Dynamics Simulations. *Nat. Commun.* **2017**, *8*, 15959.
(6) Casalino, L.; Dommer, A. C.; Gaieb, Z.; Barros, E. P.; Sztain, T.; Ahn, S.-H.; Trifan, A.; Brace, A.; Bogetti, A. T.; Clyde, A.; Ma, H.; Lee, H.; Turilli, M.; Khalid, S.; Chong, L. T.; Simmerling, C.; Hardy, D. J.; Maia, J. D.; Phillips, J. C.; Kurth, T.; Stern, A. C.; Huang, L.; McCalpin, J. D.; Tatineni, M.; Gibbs, T.; Stone, J. E.; Jha, S.; Ramanathan, A.; Amaro, R. E. AI-Driven Multiscale Simulations Illuminate Mechanisms of SARS-CoV-2 Spike Dynamics. *Int. J. High Perform. Comput. Appl.* **2021**, *35*, 432−451.
(7) Jung, J.; Nishima, W.; Daniels, M.; Bascom, G.; Kobayashi, C.; Adedoyin, A.; Wall, M.; Lappala, A.; Phillips, D.; Fischer, W.; Tung, C. S.; Schlick, T.; Sugita, Y.; Sanbonmatsu, K. Y. Scaling Molecular Dynamics beyond 100,000 Processor Cores for Large-Scale Biophysical Simulations. *J. Comput. Chem.* **2019**, *40*, 1919−1930.
(8) 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. Atomic-Level Characterization of the Structural Dynamics of Proteins. *Science* **2010**, *330*, 341−346.
(9) Zuckerman, D. M.; Woolf, T. B. Efficient Dynamic Importance Sampling of Rare Events in One Dimension. *Phys. Rev. E* **2000**, *63*, 016702.
(10) Faradjian, A. K.; Elber, R. Computing Time Scales from Reaction Coordinates by Milestoning. *J. Chem. Phys.* **2004**, *120*, 10880−10889.
(11) West, A. M. A.; Elber, R.; Shalloway, D. Extending Molecular Dynamics Time Scales with Milestoning: Example of Complex Kinetics in a Solvated Peptide. *J. Chem. Phys.* **2007**, *126*, 145104.
(12) van Erp, T. S.; Moroni, D.; Bolhuis, P. G. A Novel Path Sampling Method for the Calculation of Rate Constants. *J. Chem. Phys.* **2003**, *118*, 7762−7774.
(13) Allen, R. J.; Valeriani, C.; Rein ten Wolde, P. ten. Forward Flux Sampling for Rare Event Simulations. *J. Phys.: Condens. Matter* **2009**, *21*, 463102.
(14) Pratt, L. R. A Statistical Method for Identifying Transition States in High Dimensional Problems. *J. Chem. Phys.* **1986**, *85*, 5045−5048.
(15) Swenson, D. W. H.; Bolhuis, P. G. A Replica Exchange Transition Interface Sampling Method with Multiple Interface Sets for Investigating Networks of Rare Events. *J. Chem. Phys.* **2014**, *141*, 044101.
(16) DeFever, R. S.; Sarupria, S. Contour Forward Flux Sampling: Sampling Rare Events along Multiple Collective Variables. *J. Chem. Phys.* **2019**, *150*, 024103.

(17) Huber, G. A.; Kim, S. Weighted-Ensemble Brownian Dynamics Simulations for Protein Association Reactions. *Biophys. J.* **1996**, *70*, 97−110.
(18) Ray, D.; Stone, S. E.; Andricioaei, I. Markovian Weighted Ensemble Milestoning (M-WEM): Long-Time Kinetics from Short Trajectories. *J. Chem. Theor. Comput.* **2022**, *18* (1), 79−95.
(19) Zuckerman, D. M.; Chong, L. T. Weighted Ensemble Simulation: Review of Methodology, Applications, and Software. *Annu. Rev. Biophys.* **2017**, *46*, 43−57.
(20) Adhikari, U.; Mostofian, B.; Copperman, J.; Subramanian, S. R.; Petersen, A. A.; Zuckerman, D. M. Computational Estimation of Microsecond to Second Atomistic Folding Times. *J. Am. Chem. Soc.* **2019**, *141*, 6519−6526.
(21) Zwier, M. C.; Pratt, A. J.; Adelman, J. L.; Kaus, J. W.; Zuckerman, D. M.; Chong, L. T. Efficient Atomistic Simulation of Pathways and Calculation of Rate Constants for a Protein−Peptide Binding Process: Application to the MDM2 Protein and an Intrinsically Disordered P53 Peptide. *J. Phys. Chem. Lett.* **2016**, *7*, 3440−3445.
(22) Saglam, A. S.; Chong, L. T. Protein−Protein Binding Pathways and Calculations of Rate Constants Using Fully-Continuous, Explicit-Solvent Simulations. *Chem. Sci.* **2019**, *10*, 2360−2372.
(23) Lotz, S. D.; Dickson, A. Unbiased Molecular Dynamics of 11 Min Timescale Drug Unbinding Reveals Transition State Stabilizing Interactions. *J. Am. Chem. Soc.* **2018**, *140*, 618−628.
(24) Sztain, T.; Ahn, S.-H.; Bogetti, A. T.; Casalino, L.; Goldsmith, J. A.; Seitz, E.; McCool, R. S.; Kearns, F. L.; Acosta-Reyes, F.; Maji, S.; Mashayekhi, G.; McCammon, J. A.; Ourmazd, A.; Frank, J.; McLellan, J. S.; Chong, L. T.; Amaro, R. E. A Glycan Gate Controls Opening of the SARS-CoV-2 Spike Protein. *Nat. Chem.* **2021**, *13*, 963.
(25) Zwier, M. C.; Adelman, J. L.; Kaus, J. W.; Pratt, A. J.; Wong, K. F.; Rego, N. B.; Suárez, E.; Lettieri, S.; Wang, D. W.; Grabe, M.; Zuckerman, D. M.; Chong, L. T. WESTPA: An Interoperable, Highly Scalable Software Package for Weighted Ensemble Simulation and Analysis. *J. Chem. Theory Comput.* **2015**, *11*, 800−809.
(26) Zhang, B. W.; Jasnow, D.; Zuckerman, D. M. The "Weighted Ensemble" Path Sampling Method Is Statistically Exact for a Broad Class of Stochastic Processes and Binning Procedures. *J. Chem. Phys.* **2010**, *132*, 054107.
(27) Abdul-Wahid, B.; Feng, H.; Rajan, D.; Costaouec, R.; Darve, E.; Thain, D.; Izaguirre, J. A. AWE-WQ: Fast-Forwarding Molecular Dynamics Using the Accelerated Weighted Ensemble. *J. Chem. Inf. Model.* **2014**, *54*, 3033−3043.
(28) Lotz, S. D.; Dickson, A. Wepy: A Flexible Software Framework for Simulating Rare Events with Weighted Ensemble Resampling. *ACS Omega* **2020**, *5*, 31608−31623.
(29) Saglam, A. S.; Chong, L. T. Highly Efficient Computation of the Basal k_{on} Using Direct Simulation of Protein−Protein Association with Flexible Molecular Models. *J. Phys. Chem. B* **2016**, *120*, 117−122.
(30) Donovan, R. M.; Tapia, J.-J.; Sullivan, D. P.; Faeder, J. R.; Murphy, R. F.; Dittrich, M.; Zuckerman, D. M. Unbiased Rare Event Sampling in Spatial Stochastic Systems Biology Models Using a Weighted Ensemble of Trajectories. *PLoS Comput. Biol.* **2016**, *12*, No. e1004611.
(31) Tapia, J.-J.; Saglam, A. S.; Czech, J.; Kuczewski, R.; Bartol, T. M.; Sejnowski, T. J.; Faeder, J. R. MCell-R: A Particle-Resolution Network-Free Spatial Modeling Framework. *Methods Mol. Biol.* **2019**, *1945*, 203−229.
(32) Johnson, M. E.; Chen, A.; Faeder, J. R.; Henning, P.; Moraru, I. I.; Meier-Schellersheim, M.; Murphy, R. F.; Prüstel, T.; Theriot, J. A.; Uhrmacher, A. M. Quantifying the Roles of Space and Stochasticity in Computer Simulations for Cell Biology and Cellular Biochemistry. *MBoC* **2021**, *32*, 186−210.
(33) Donyapour, N.; Roussey, N. M.; Dickson, A. REVO: Resampling of Ensembles by Variation Optimization. *J. Chem. Phys.* **2019**, *150*, 244112.
(34) Bhatt, D.; Zhang, B. W.; Zuckerman, D. M. Steady-State Simulations Using Weighted Ensemble Path Sampling. *J. Chem. Phys.* **2010**, *133*, 014110.
(35) Suárez, E.; Lettieri, S.; Zwier, M. C.; Stringer, C. A.; Subramanian, S. R.; Chong, L. T.; Zuckerman, D. M. Simultaneous Computation of Dynamical and Equilibrium Information Using a Weighted Ensemble of Trajectories. *J. Chem. Theory Comput.* **2014**, *10*, 2658−2667.
(36) Bogetti, A. T.; Mostofian, B.; Dickson, A.; Pratt, A.; Saglam, A. S.; Harrison, P. O.; Adelman, J. L.; Dudek, M.; Torrillo, P. A.; DeGrave, A. J.; Adhikari, U.; Zwier, M. C.; Zuckerman, D. M.; Chong, L. T. A Suite of Tutorials for the WESTPA Rare-Events Sampling Software [Article v1.0]. *Living J. Comput. Mol. Sci.* **2019**, *1*, 10607.
(37) WESTPA. WESTPA Tutorials; Github Repository. 2021, https://github.com/westpa/westpa_tutorials (accessed October 25, 2021).
(38) Orion. *OpenEye Scientific Software*; Orion: Santa Fe, NM.
(39) Grebner, C.; Malmerberg, E.; Shewmaker, A.; Batista, J.; Nicholls, A.; Sadowski, J. Virtual Screening in the Cloud: How Big Is Big Enough? *J. Chem. Inf. Model.* **2020**, *60*, 4274−4282.
(40) WESTPA. WESTPA 2.0 Tutorials; Github Repository, 2021. https://github.com/westpa/westpa2_tutorials (accessed October 25, 2021).
(41) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics. *PLoS Comput. Biol.* **2017**, *13*, No. e1005659.
(42) Torrillo, P. A.; Bogetti, A. T.; Chong, L. T. A Minimal, Adaptive Binning Scheme for Weighted Ensemble Simulations. *J. Phys. Chem. A* **2021**, *125*, 1642−1649.
(43) Adelman, J. L.; Grabe, M. Simulating Rare Events Using a Weighted Ensemble-Based String Method. *J. Chem. Phys.* **2013**, *138*, 044105.
(44) Dickson, A.; Brooks, C. L. WExplore: Hierarchical Exploration of High-Dimensional Spaces Using the Weighted Ensemble Algorithm. *J. Phys. Chem. B* **2014**, *118*, 3532−3542.
(45) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophys. J.* **2015**, *109*, 1528−1532.
(46) Nguyen, H.; Case, D. A.; Rose, A. S. NGLview−Interactive Molecular Graphics for Jupyter Notebooks. *Bioinformatics* **2018**, *34*, 1241−1242.
(47) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. GROMACS 4:Algorithms for Highly Efficient ,Load -Balanced,and Scalable Molecular Simulation. *J. Chem. Theory Comput.* **2008**, *4*, 435−447.
(48) DeGrave, A. J.; Bogetti, A. T.; Chong, L. T. The RED Scheme: Rate-Constant Estimation from Pre-Steady State Weighted Ensemble Simulations. *J. Chem. Phys.* **2021**, *154*, 114111.
(49) Mostofian, B.; Zuckerman, D. M. Statistical Uncertainty Analysis for Small-Sample, High Log-Variance Data: Cautions for Bootstrapping and Bayesian Bootstrapping. *J. Chem. Theory Comput.* **2019**, *15*, 3499−3509.
(50) Copperman, J.; Zuckerman, D. M. Accelerated Estimation of Long-Timescale Kinetics from Weighted Ensemble Simulation via Non-Markovian "Microbin" Analysis. *J. Chem. Theory Comput.* **2020**, *16*, 6763−6775.
(51) Created with http://Biorender.com, 2021 (accessed October 25, 2021).
(52) Chodera, J. D.; Noé, F. Markov State Models of Biomolecular Conformational Dynamics. *Curr. Opin. Struct. Biol.* **2014**, *25*, 135−144.
(53) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An Overview of the Amber Biomolecular Simulation Package. *WIREs Comput. Mol. Sci.* **2013**, *3*, 198−210.

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

(54) Suárez, E.; Pratt, A. J.; Chong, L. T.; Zuckerman, D. M. Estimating First-Passage Time Distributions from Weighted Ensemble Simulations and Non-Markovian Analyses. *Protein Sci.* **2016**, *25*, 67−78.
(55) ZuckermanLab. NMpathAnalysis; Github Repository, 2021. https://github.com/ZuckermanLab/NMpathAnalysis (accessed October 25, 2021).
(56) Bhowmik, D.; Gao, S.; Young, M. T.; Ramanathan, A. Deep Clustering of Protein Folding Simulations. *BMC Bioinf.* **2018**, *19*, 484.
(57) Ribeiro, J. M. L.; Bravo, P.; Wang, Y.; Tiwary, P. Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE). *J. Chem. Phys.* **2018**, *149*, 072301.
(58) Swenson, D. W. H.; Prinz, J.-H.; Noe, F.; Chodera, J. D.; Bolhuis, P. G. OpenPathSampling: A Python Framework for Path Sampling Simulations. 1. Basics. *J. Chem. Theory Comput.* **2019**, *15*, 813−836.
(59) Swenson, D. W. H.; Prinz, J.-H.; Noe, F.; Chodera, J. D.; Bolhuis, P. G. OpenPathSampling: A Python Framework for Path Sampling Simulations. 2. Building and Customizing Path Ensembles and Sample Schemes. *J. Chem. Theory Comput.* **2019**, *15*, 837−856.
(60) DeFever, R. S.; Hanger, W.; Sarupria, S.; Kilgannon, J.; Apon, A. W.; Ngo, L. B. Building A Scalable Forward Flux Sampling Framework Using Big Data and HPC. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning); PEARC '19*; Association for Computing Machinery: New York, NY, USA, 2019; pp 1−8.
(61) Wei, W.; Elber, R. ScMile: A Script to Investigate Kinetics with Short Time Molecular Dynamics Trajectories and the Milestoning Theory. *J. Chem. Theory Comput.* **2020**, *16*, 860−874.
(62) Romo, T. D.; Grossfield, A. LOOS: An Extensible Platform for the Structural Analysis of Simulations. *In 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*; IEEE, 2009; pp 2332−2335.
(63) Romo, T. D.; Leioatts, N.; Grossfield, A. Lightweight Object Oriented Structure Analysis: Tools for Building Tools to Analyze Molecular Dynamics Simulations. *J. Comput. Chem.* **2014**, *35*, 2305−2318.
(64) Michaud-Agrawal, N.; Denning, E. J.; Woolf, T. B.; Beckstein, O. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.* **2011**, *32*, 2319−2327.
(65) Gowers, R. J.; Linke, M.; Barnoud, J.; Reddy, T. J. E.; Melo, M. N.; Seyler, S. L.; Domański, J.; Dotson, D. L.; Buchoux, S.; Kenney, I. M.; Beckstein, O. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. *Proceedings of the 15th Python in Science Conference*; Los Alamos National Lab., 2016; pp 98−105.
(66) Scherer, M. K.; Trendelkamp-Schroer, B.; Paul, F.; Pérez-Hernández, G.; Hoffmann, M.; Plattner, N.; Wehmeyer, C.; Prinz, J.-H.; Noé, F. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *J. Chem. Theory Comput.* **2015**, *11*, 5525−5542.

https://doi.org/10.1021/acs.jctc.1c01154
J. Chem. Theory Comput. XXXX, XXX, XXX−XXX