# WESTPA: An Interoperable, Highly Scalable Software Package for Weighted Ensemble Simulation and Analysis

Matthew C. Zwier,[†] Joshua L. Adelman,[‡] Joseph W. Kaus,[¶] Adam J. Pratt,[¶] Kim F. Wong,[§] Nicholas B. Rego,[¶] Ernesto Suárez,[‖] Steven Lettieri,[‖] David W. Wang,[¶] Michael Grabe,[⊥] Daniel M. Zuckerman,[‖] and Lillian T. Chong*[,¶]

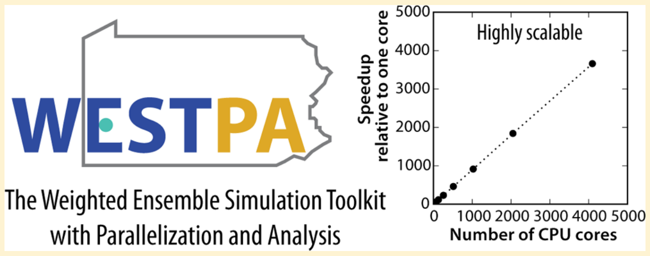
[†]Department of Chemistry, Drake University, Des Moines, Iowa 50311, United States

[‡]Department of Biological Sciences, [¶]Department of Chemistry, [§]Center for Simulation and Modeling, and [‖]Department of Computational and Systems Biology, University of Pittsburgh, Pittsburgh, Pennsylvania 15206, United States

[⊥]Cardiovascular Research Institute, Department of Pharmaceutical Chemistry, University of California, San Francisco, San Francisco, California 94158, United States

*Supporting Information*

**ABSTRACT:** The weighted ensemble (WE) path sampling approach orchestrates an ensemble of parallel calculations with intermittent communication to enhance the sampling of rare events, such as molecular associations or conformational changes in proteins or peptides. Trajectories are replicated and pruned in a way that focuses computational effort on underexplored regions of configuration space while maintaining rigorous kinetics. To enable the simulation of rare events at any scale (e.g., atomistic, cellular), we have developed an open-source, interoperable, and highly scalable software package for the execution and analysis of WE simulations: WESTPA (The Weighted Ensemble Simulation Toolkit with Parallelization and Analysis). WESTPA scales to thousands of CPU cores and includes a suite of analysis tools that have been implemented in a massively parallel fashion. The software has been designed to interface conveniently with any dynamics engine and has already been used with a variety of molecular dynamics (e.g., GROMACS, NAMD, OpenMM, AMBER) and cell-modeling packages (e.g., BioNetGen, MCell). WESTPA has been in production use for over a year, and its utility has been demonstrated for a broad set of problems, ranging from atomically detailed host−guest associations to nonspatial chemical kinetics of cellular signaling networks. The following describes the design and features of WESTPA, including the facilities it provides for running WE simulations and storing and analyzing WE simulation data, as well as examples of input and output.



## 1. INTRODUCTION

Despite advances in computer hardware, many rare events including molecular associations or large-scale conformational transitions in proteins are beyond the reach of dynamical simulations. A number of software packages have therefore been developed for rare-events simulations that enhance sampling without introducing bias in the dynamics. These fall into three categories: (a) packages that integrate enhanced sampling into the dynamics engine (e.g., Milestoning[1] in MOIL[2] or transition path sampling[3] in CHARMM[4]), (b) framework software that interfaces with multiple dynamics engines (e.g., FRESHS,[5] which employs—among others—the forward flux sampling approach[6]); and (c) analysis packages that provide external tools to extract long-time scale information from already completed simulations (e.g., MSMBuilder[7] or EMMA,[8] which construct Markov state models[9] that can in turn be used to enhance sampling in an iterative manner[10,11]).

Here, we report on a computational framework for studying rare events that employs the weighted ensemble (WE) path sampling approach. WE is a proven methodology for accelerating simulations of rare events while maintaining rigorous kinetics.[12−26] In some cases, both pathways and rate constants can be generated with orders of magnitude greater efficiency than standard simulations[12,15,18,19,23] in terms of total computing time. The approach is rigorous for any type of stochastic simulation method, including molecular dynamics (MD) and Monte Carlo simulations. WE sampling can yield equilibrium as well as nonequilibrium observables (state populations and rate constants, respectively) and is applicable to equilibrium or nonequilibrium steady-state processes and simulations of relaxation toward a steady state.[16] The efficiency of WE simulations can be further increased, particularly in the presence of metastable intermediates, by the use of reweighting procedures that determine global thermodynamic properties in terms of local kinetic properties.[17,27]

**Received:** November 26, 2014
**Published:** January 13, 2015

We have developed an open-source, interoperable, highly scalable software package that embodies the full range of WE's capabilities. Called WESTPA (The Weighted Ensemble Simulation Toolkit with Parallelization and Analysis, https://chong.chem.pitt.edu/WESTPA), this package has been available for over a year as (to our knowledge) the first highly scalable and freely available WE software package. We note that since the release of WESTPA, another WE package has become available — AWE-WQ[25] — which was designed for use with the Work Queue task distribution system and has been used in conjunction with the GROMACS molecular dynamics (MD) engine.[28] Additionally, several in-house implementations of the WE algorithm exist, including WExplore,[24] which has been used with the CHARMM simulation package.[4] Our WESTPA package appears to be unique in its demonstrated ability to interface with a large number of dynamics engines and in providing a comprehensive framework for data generation and storage, along with integral support for the novel analysis and simulation protocols we describe below. The software has been designed to interface conveniently with any dynamics engine at any system size or level of detail, including atomistic, coarse-grained, and whole-cell. The WESTPA software package can be used with any typical scientific computing platform, including desktop workstations, commodity clusters, and supercomputers, and can automatically take advantage of accelerator technologies like graphics processing units (GPUs) or other coprocessors.

To date, WESTPA has been used with MD engines (AMBER,[29] GROMACS,[28] NAMD,[30] and OpenMM[31]), a Brownian dynamics engine (UIOWA_BD[32,33]), and systems biology engines (BioNetGen[34] and MCell[35]). Interfacing WESTPA with a new dynamics engine is straightforward, as no modifications to the source code of the dynamics engine are necessary. The WESTPA software package has enabled efficient atomistic simulations of host–guest associations in explicit solvent,[19] atomistic conformational sampling of peptides in implicit solvent,[27] coarse-grained simulations of large-scale protein conformational transitions,[20,23] and nonspatial simulations of chemical cellular signaling networks in the context of systems biology.[22]

In the present report, after a brief review of WE algorithms, we describe the design and features of WESTPA that have enabled its broad applicability.

## 2. OVERVIEW OF WE SAMPLING

WE sampling uses an ensemble of independent simulation trajectories, each carrying a statistical weight, to explore space and promote relatively even simulation coverage along a (possibly multidimensional) progress coordinate describing a rare event. This progress coordinate is divided into bins, and the relatively even coverage of WE simulations is attained by periodically replicating trajectories in bins with too few trajectories and pruning trajectories in bins containing too many trajectories (Figure 1).[12] There is great flexibility in bin construction, as detailed below, and—notably—the choice of bins affects the efficiency but not the correctness of WE simulations. Careful management of the statistical weights associated with each trajectory ensures that no bias is introduced by this procedure, which amounts to a statistical resampling among trajectories.[16] When using MD simulations to propagate the dynamics, a stochastic thermostat is required since the dynamics of the WE trajectories must diverge when the trajectories are replicated. Although the choice of

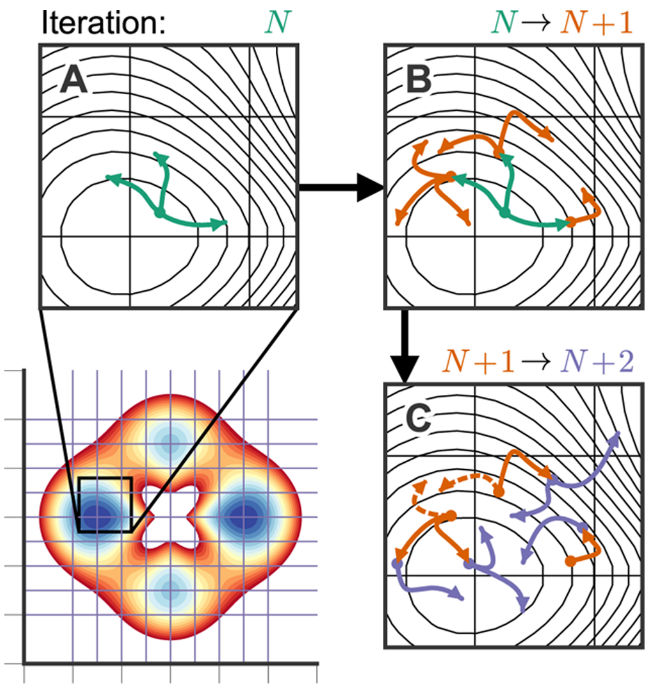


**Figure 1.** Basic weighted ensemble protocol. A two-dimensional energy landscape is partitioned into bins using a rectilinear grid with a target of three trajectories per bin (lower left panel). Trajectories, which are each assigned a weight, are initiated in one of the dominant metastable basins and are propagated for a short interval, $\tau$. A magnified view of the bin containing these trajectories is shown in (A), and the paths that each trajectory segment follow are shown in green for iteration $N$. The trajectories are then resampled according to the WE protocol. (B) One of the three trajectories reaches an unoccupied bin, and its weight is evenly distributed among three copies. Similarly one of the two trajectory segments that remained in the initial bin is replicated to maintain the target number of trajectories per bin. This collection of six trajectories of the system are propagated again for an interval $\tau$ during iteration $N + 1$ (red paths). (C) When a bin contains more than the target number of trajectories, one or more are terminated (dashed paths), and their weight is assigned statistically to the remaining trajectories in the bin. Trajectory segments for the subsequent iteration ($N + 2$) are shown as purple paths.

thermostat can generally be expected to perturb dynamics,[36,37] we note that certain stochastic thermostats can be run with sufficiently weak coupling constants such that the dynamical properties of simple test systems are minimally perturbed relative to those of the microcanonical (NVE) ensemble.[37] Thermodynamic information such as free energy landscapes (the spatial distribution of weights) and kinetic information such as rates of transition between states (the fluxes of weight across arbitrary surfaces in progress coordinate space) are simultaneously generated and readily available from WE simulations. The efficiency of WE sampling compared to standard (brute-force) simulation has been shown to increase dramatically with increasing barrier height (or, equivalently, increasingly rare events).[15,19,23]

WE is naturally suited to simulating a variety of initial conditions and dynamical ensembles including equilibrium, nonequilibrium steady states (both with and without driving forces), and relaxation from an arbitrary initial condition to a chosen steady state.[16] Most basically, WE can be used to study relaxation from an initial condition, specified by the initial set of trajectories and their weights, toward equilibrium. With only slight additional complexity, WE can be used to estimate rates

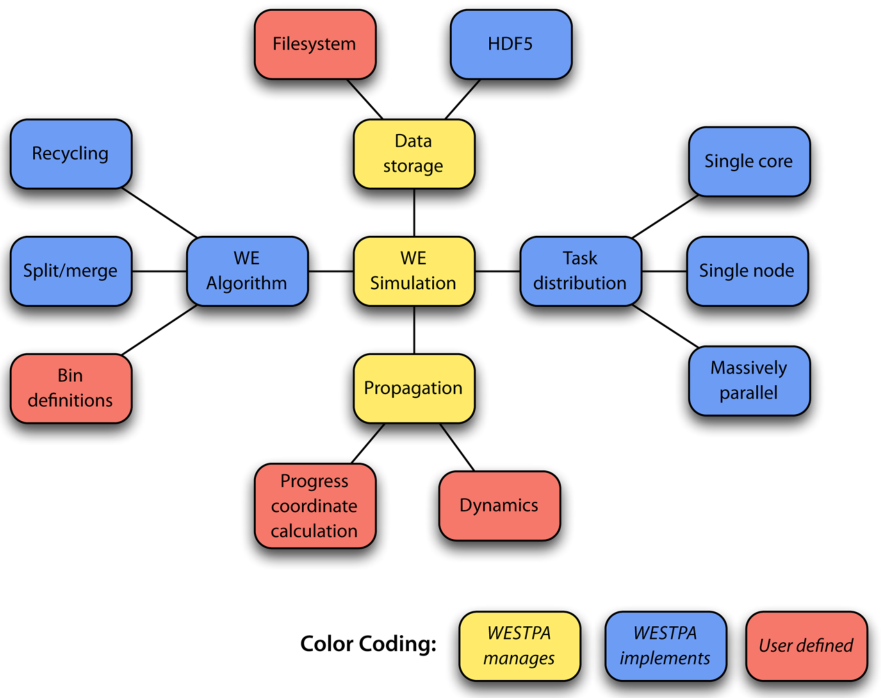


**Figure 2.** Logical structure of the WESTPA software package. Components shown in blue are implemented in WESTPA and may be replaced, if desired, by custom code. Components shown in yellow indicate functionality that WESTPA coordinates, which can generally be customized by configuration options. Components shown in red must be provided by the user, typically as Python modules, shell scripts, or compiled executables.

of unidirectional (nonequilibrium) steady states, as implemented in the original WE paper;[12] a steady state is simulated by removing trajectories reaching a specified target state and reinitiating them from a chosen start-state distribution. These procedures track the unbiased evolution of an initial distribution toward a steady state, but the relaxation process may be very slow.

To enable WE simulations to reach steady states of interest (including equilibrium) despite slow relaxation times, enhanced WE procedures have been developed.[17,27] These newer approaches take advantage of the fact that all WE trajectories are unbiased (no artificial forces are applied) and hence can be used to estimate conditional probabilities, or effective rate constants, for bin-to-bin transitions. These rate constants, in turn, can be used to estimate steady-state bin populations which would hold at long times.[17] Two separate approaches for using the bin-to-bin rates have been developed. In the first, trajectories are reweighted on-the-fly to ensure conformance with estimated steady-state bin probabilities.[17] In the second, a post-simulation analysis protocol is used to estimate kinetic and equilibrium properties for arbitrary states selected after the simulation has been run.[27] These procedures can be quite important for enhancing the efficiency of WE simulation,[17,20,21,23,25−27] particularly in the presence of metastable intermediate states that would otherwise make converged determination of thermodynamic and kinetic observables (populations and rates) prohibitively slow.

The basic WE procedure of running independent trajectories for fixed time intervals with intermittent communication lends itself naturally to parallelization and interoperability. The intermittency of communication leads almost immediately to excellent parallel scaling, with high performance whenever the length of trajectory segments exceeds the "overhead" time for non-dynamics operations such as writing trajectories or assessing bin occupancy.

Facile interoperability arises from WE's use of fixed-length trajectory segments. The only requirements for interfacing with a dynamics engine are the ability to start and stop dynamics and query progress coordinates at fixed time points. There is no need to modify the dynamics software itself because WE does not "catch" trajectories in the act of crossing from one bin to another; rather, the theory and implementation are built on examining the ensemble at fixed time points. This permits the use of a variety of dynamics engines in conjunction with WE sampling, and, importantly, optimizations for these dynamics packages designed to increase simulation throughput can remain in place when used in WE simulations.

Implementation parameters for a WE simulation can be changed on-the-fly without biasing the results.[16] In particular, the progress coordinate(s) and the binning can be updated during the course of a simulation without needing to discard existing simulation data, allowing the computational cost of a WE simulation to be tuned on-the-fly to maximize its effectiveness.

WESTPA is designed to run WE simulations efficiently while exploiting all of the above strengths of WE sampling, particularly supporting a variety of dynamics engines, various initial and boundary conditions, online tuning of progress coordinate and binning, massively parallel dynamics propagation, and (via analysis tools and the plugin architecture described below) the enhanced WE procedures described previously.

## 3. ORGANIZATION AND FEATURES OF WESTPA

**3.1. Software Design.** The design of the WESTPA code is highly modular (see Figure 2), enabling straightforward customization of how simulations can be performed. New or

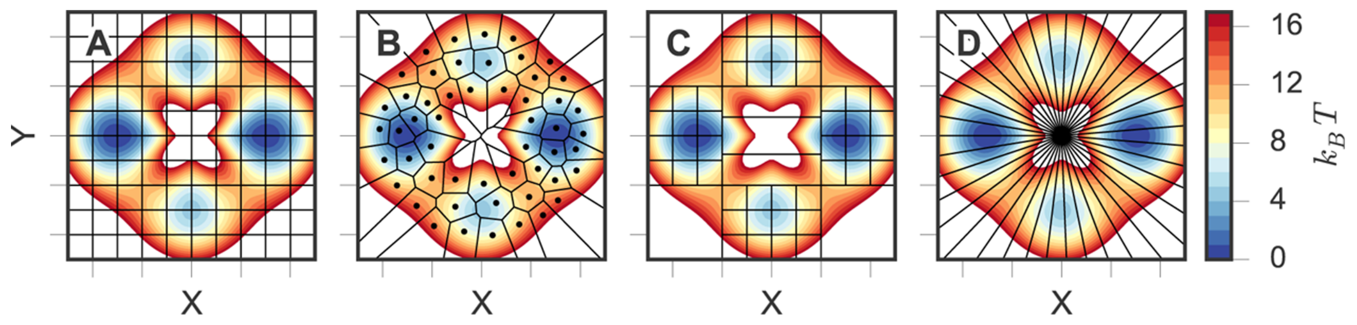


**Figure 3.** Discretization of progress coordinate space into bins. The conformational space of a simple two-dimensional energy with multiple metastable basins[23] is discretized into bins using different schemes available in WESTPA. In (A) the system is divided into bins using a grid, while in (B) a set of points covering the landscape defines a tessellation of Voronoi cells. Bins may be defined hierarchically as shown in (C), where a coarse grid is further subdivided into finer sets of bins in some regions. In (D), bin boundaries extend radially from the center of the potential and are implemented using a user-defined function that, given a set of coordinates, returns the appropriate bin assignments. Examples of how each binning scheme is defined in WESTPA are provided in the Supporting Information.

altered modules can easily be substituted into the WESTPA framework at runtime. In particular, WESTPA provides a number of hooks that allow plugins (small pieces of code written by the user) to perform custom processing at several points through the main simulation loop without the need to modify the WESTPA code itself. For example, such plugins have been used to implement a finite-temperature string method[23] without requiring modifications to the “core” WESTPA code. For more complex customizations, most components of the WESTPA software (such as those responsible for propagating dynamics or effecting the binning, replication, and pruning of the WE sampling scheme) can also be swapped for custom versions at run time by modification of a configuration file.

WESTPA is written in the Python programming language,[38] allowing WESTPA to leverage the diverse, high performance, and rapidly expanding Python scientific computing ecosystem.[39] WESTPA achieves native Fortran or C performance by its use of Numpy arrays[40] for numeric data and the optimization of critical routines through the Cython extension language for Python.[41] This combination of Python, Numpy, and Cython enables rapid development of readable source code without a substantial sacrifice of run-time performance.

**3.2. Compatibility with Existing Dynamics Software.** As the WE approach is rigorously correct for any stochastic simulation,[16] WESTPA is designed to interface with any existing simulation package. This includes traditional standalone MD packages like GROMACS,[28] AMBER,[29] or NAMD.[30] For increased flexibility and efficiency, WESTPA is also capable of interfacing with simulation toolkits like OpenMM[31] or fully user-programmed routines via a relatively simple interface; a customized dynamics propagator need only define three relatively simple Python functions (“propagate dynamics”, “generate initial state for a new trajectory”, and “get progress coordinate”). Notably, because propagation of dynamics is typically handled by programs or routines external to WESTPA, any optimizations that are already in place for propagation of dynamics (such as use of optimized linear algebra libraries or offloading of computational work to GPUs or other coprocessors) are automatically used by WESTPA simulations, as previously discussed. Configuring WESTPA to use a new dynamics engine (i.e., one that has not been used with WESTPA previously) usually only requires hours or a few days of effort, since only shell scripting — and notably no “under-the-hood” modifications of the dynamics engine source code — is required. Instructions for interfacing with various software packages are available through the WESTPA Web site (https://chong.chem.pitt.edu/WESTPA/).

**3.3. Flexibility in Tuning WE Cost/Payoff Balance.** The WE approach is highly flexible among enhanced sampling techniques, and WESTPA provides users with the ability to exploit this flexibility to tune the balance between computational cost and level of sampling. As discussed in Section 2, WE naturally supports multidimensional progress coordinates and may be used to perform simulations under equilibrium conditions or nonequilibrium steady states. The enhanced WE procedures discussed previously may be used to accelerate convergence in both thermodynamic and kinetic observables. Progress coordinates can represent history-dependent quantities (e.g., the last state visited by a trajectory), which can be used by these enhanced WE procedures in attaining proper steady states and in subsequent analysis of thermodynamics and kinetics. WESTPA includes support for equilibrium, nonequilibrium, or steady-state boundary conditions, flexible (possibly history-dependent) binning schemes, and the enhanced WE procedures described above.

Progress coordinates in WESTPA may be single- or multidimensional and may be divided into bins by boundaries on grids, Voronoi cells, or any user-defined function that can map progress coordinate values to integers (bin numbers). Sets of bins may be arbitrarily nested, allowing the rapid construction of complex binning schemes. Examples of these binning strategies are shown in Figure 3, and their implementation in WESTPA is described in the Supporting Information. The target number of trajectories can vary from bin to bin, thus allowing a direct specification of the extent of computational resources to devote to each region of progress coordinate space. Binning, including the ideal number of trajectories in each bin, may change at any point in the simulation without needing to discard existing simulation data. Combined, these features allow a user of WESTPA to, for example, assign different progress coordinates to different regions of configuration space, with each region (or bin) having a different number of trajectories. This allows the sampling and computational cost of a WESTPA simulation to be tuned to make maximally efficient use of computer resources; further, this balance can be adjusted on-the-fly as a simulation evolves.

**3.4. Data Storage.** WE simulations consist of a great number of small dynamics segments (millions or more), and WESTPA orchestrates efficient storage of this simulation data.

Foremost, subsets of trajectories formally share history, resulting from the fact that trajectories may be replicated by the WE approach prior to continuation. For maximum efficiency, this shared history is stored only once, along with a directed graph describing the connectivity of trajectory segments. The type of data needed to drive or extract useful results from WE simulations will necessarily vary from problem to problem, so WESTPA provides users with facilities for storing data either in the native trajectory formats of the underlying dynamics engines or in a WESTPA-managed data store. Trajectories themselves (e.g., the time-ordered configurations of molecular dynamics trajectories) are typically stored in the native format of the underlying dynamics engine, which is usually highly optimized for space or speed (for instance, XTC files for GROMACS, NetCDF files for AMBER, or DCD files for NAMD). In addition to the (required) progress coordinate data, arbitrary data sets associated with each trajectory segment may be calculated and stored directly by WESTPA during the simulation for efficient access during subsequent analysis. These data sets may be stored at an arbitrarily high time resolution. WESTPA stores all data in the cross-platform, language-neutral, highly efficient HDF5 file format.[42] The HDF5 library provides optimized storage and retrieval of numeric data. Data produced by WESTPA may be accessed from any programming language which has HDF5 bindings (Python, C, C++, Fortran, Java, R, MATLAB, and Julia, among many others).

**3.5. Parallelization and Scaling.** WE simulations are naturally parallel, as trajectories are very loosely coupled; the propagation of *N* trajectories can always be distributed over up to *N* cores, and WESTPA provides facilities to accomplish this. As shown in Figure 4, perfectly linear scaling has been achieved over thousands of cores with modest overhead. For large-scale simulations (e.g., condensed-phase molecular dynamics simulations), performance is generally not limited by the speed of communication between processors but rather how quickly external programs (e.g., for propagating dynamics or extracting progress coordinates) can be spawned or how quickly analysis tools can read data from disk. These operations appear as overhead which reduces the slope of the scaling plot below unity, but the performance impact of these operations can be mitigated by tightly integrating WESTPA with the underlying dynamics code. Importantly, even without such integration, the speedup observed in WE simulations can potentially more than compensate for the overhead imposed by segment startup/shutdown or disk I/O relative to a lower-overhead but lower-efficiency brute force simulation.

WESTPA includes facilities to distribute tasks such as dynamics propagation or subsequent analysis over multiple cores within a node and multiple nodes within a network. Different communication patterns are available for different combinations of core count and the amount of data generated for each segment of dynamics. WESTPA includes facilities for thread-based (OpenMP-like) and process-based (MPI-like) parallelization within a single node and custom TCP/IP parallelization between nodes. An interface to traditional MPI facilities for either intranode or internode communication is also provided for sites where a suitable MPI implementation exists. Due to the modular design of WESTPA, other task distribution protocols can be added as needed. For typical simulations where (as discussed above) the computational cost of dynamics propagation far outstrips the cost of communication among processors, no substantial difference in scalability or efficiency is observed among the various communication patterns available in WESTPA. The internode communication systems fully support heterogeneous clusters (those having nodes of differing speeds), including implicit load balancing via asynchronous dispatch of tasks to idle cores. Though designed for WESTPA, the task distribution module (called *wwmgr*) is completely general and may be used for writing parallel programs in Python independently of the rest of the WESTPA framework. To facilitate its use in this way, wwmgr is made available for download separately from WESTPA (in addition to being included with WESTPA).

**3.6. Analysis Tools.** WESTPA includes a suite of tools to analyze WE simulations. (See Table 1 for brief descriptions.) Each tool focuses on performing one task (e.g., “assign trajectories to bins”,“calculate the probability distribution of progress coordinate values”, or “plot the time evolution of a probability distribution”), so that tools can be chained together to perform complicated analysis tasks. Additionally, common analysis tasks are packaged as a Python programming framework for ease of reuse in customized analysis scripts. Most tools can run on multiple cores to increase analysis throughput. Both input for and output from these analysis tools are stored in the flexible and efficient HDF5 file format. Analyses not directly addressed by the tools packaged with WESTPA are relatively simple to implement in custom programs, which use the HDF5 library to read data recorded by WESTPA. The structure and meaning of data produced by WESTPA and its tools are documented on the WESTPA Web site or in the output of the online help for analysis tools (the “--help” command line option).

**3.7. Extensibility and Plugins.** WE sampling has not reached its full potential, as shown by the continued development of new algorithms for improving the WE

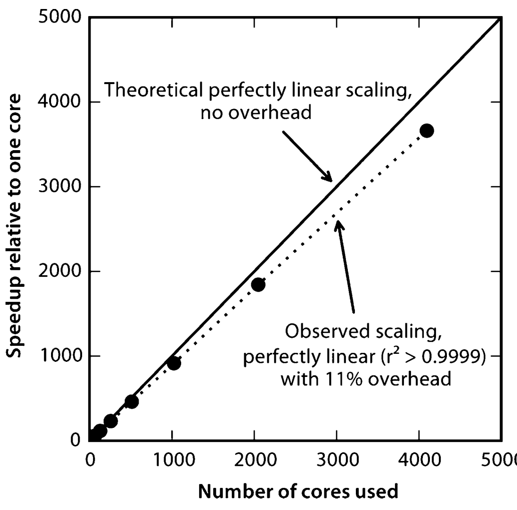


**Figure 4.** WESTPA scales linearly with minimal overhead. The solid line represents theoretically perfect (linear) scaling with no overhead; the dotted line represents the observed scaling of WESTPA, which is perfectly linear with 11% overhead (due to the cost of input/output associated with each trajectory segment). These timings are from a WE atomistic molecular dynamics simulation of the association of the MDM2 protein and a 13-residue fragment of p53 in GB/SA implicit solvent using GROMACS. This WE simulation employed up to 10,000 trajectories per iteration, each running on a single core. Individual trajectory segments were $\tau = 50$ ps in duration, requiring about 10 min of wallclock time; the total simulation required about a month on up to 4096 cores.

Table 1. Partial List of Simulation and Analysis Tools Packaged with WESTPA

| tool | description |
|---|---|
| *w_init* | Initialize a new weighted ensemble simulation. Populate the new simulation with trajectories drawn from any of several possible initial states. |
| *w_run* | Run a WESTPA simulation in serial or parallel. |
| *w_bins* | Describe or alter the bin space, including trajectories per bin, for a WESTPA simulation. |
| *w_states* | Add or remove initial (source) or target (sink) states for a WESTPA simulation. |
| *w_crawl* | Perform a custom analysis on data from an entire WESTPA simulation (e.g., to calculate a new simulation observable from existing trajectory data). |
| *w_pdist* | Calculate probability distributions of simulation observables (such as progress coordinate values or spatial distributions). |
| *w_eddist* | Calculate transition event duration (transit time) distributions, useful for characterizing the diversity of transition pathways[43] in a WESTPA simulation. |
| *plothist* | Plot histograms (probability distributions) for data generated with *w_pdist* or *w_eddist*. Supports one- or two-dimensional histograms either of instantaneous distributions or distributions averaged over a specified simulation time window. Also supports displaying the evolution of a one-dimensional probability distribution as a function of simulation time (useful for gauging simulation progress). |
| *w_assign* | Assign trajectories to modified bins, progress coordinates, and/or macrostates for the purposes of analysis. |
| *w_ntop* | Select a number of trajectories from a given bin (e.g., for visualization of structures in atomistic simulations). |
| *w_trace* | Trace the history of an individual trajectory, optionally recording time-resolved simulation observables along the trajectory. |
| *w_stateprobs* | Calculate the average population of macrostates previously defined by *w_assign*, including time-correlated confidence intervals. |
| *w_kinetics* | Analyze the kinetics of WESTPA simulations, including bin-to-bin and state-to-state fluxes and transition rates. |
| *w_kinavg* | Perform averaging of kinetics observables previously calculated with *w_kinetics*, taking time correlation into account. |
| *ploterr* | Plot the time evolution of simulation observables, including confidence intervals. |

scheme,[17,21,23,24,27] and WESTPA is designed to easily facilitate changes and extensions to the WE approach. In addition to WESTPA's modular design that allows a user to replace individual components of the software package, a simulation can be modified in-progress via WESTPA's plugin system. A plugin is a piece of code that is registered to run at a specific execution point in the main simulation loop. After activating the plugin in the configuration file, WESTPA automatically executes it at runtime, giving it full access to all of the underlying data structures and—importantly—the ability to modify them. Currently, WESTPA allows plugins to run during the initial startup of a WE simulation and during final shutdown, as well as before and after the WE resampling step, trajectory propagation, and individual iterations. Multiple plugins can be registered at the same execution point and run in a specific order to allow complex behaviors to be encoded as a series of small and discrete steps.

As an example, the plugin system was used in ref 23 to validate a WE-based string method, in which the bin space (consisting of a one-dimensional path through a high dimensional phase space) was dynamically updated based on the accumulated sampling of the WE trajectories. Additionally, the weights of trajectories were adjusted on-the-fly to hasten convergence of the simulations using a reweighting protocol that uses bin-to-bin fluxes to solve for the global steady-state of the system.[17] This string method plugin is bundled with WESTPA.

## 4. RESOURCES FOR USERS AND DEVELOPERS

**4.1. Resources for Users.** To help users get started quickly with the WESTPA software, we provide tutorials, example simulations, and tools to facilitate communication among WESTPA users. The WESTPA Wiki (https://chong.chem.pitt.edu/wewiki/) provides a collaboratively edited source of documentation on WESTPA, including both detailed documentation about the WESTPA software itself and general documentation on how to construct and run WE simulations using WESTPA. A number of tutorials describe how to use WESTPA with the popular GROMACS, AMBER, and NAMD molecular dynamics engines and the BioNetGen systems biology engine, along with how to construct a custom WE simulation using the OpenMM toolkit. The files necessary for running most of these tutorials are packaged as examples distributed with the WESTPA source code.

The WESTPA command-line tools themselves (see Table 1) are constructed with usability in mind. Each tool has been designed to be modular and optimized for a specific analysis task, allowing users to construct relatively complex analyses from discrete and comprehensible analysis steps. Input and output data for these analysis tools are stored in HDF5 files, allowing users to insert their own analysis programs, written in their programming language(s) of choice, into the analysis chain provided by WESTPA tools in the event that greater flexibility is required in analyzing WESTPA simulations. Each tool has brief but complete online help, accessible by providing the "--help" option on the command line, which describes the purpose, use, input, and output of the tool. The output format descriptions are particularly notable, as they provide enough information to allow users to take the output from a WESTPA analysis tool and use it as input (via the HDF5 library) for their own analysis scripts and programs, which are often necessary for answering specific scientific questions or preparing publication quality figures.

Finally, we provide a number of mechanisms to foster communication among the WESTPA community to ensure that users can employ WESTPA in as effective a manner as possible in their research. We have created an e-mail mailing list for WESTPA users to provide a forum where questions about how to run WE simulations can be asked and addressed; instructions for joining this list are posted on the WESTPA Web site. Further, the Github site which hosts the WESTPA code (described below) provides a mechanism for reporting potential problems with the WESTPA software and tracking their resolution. The same mechanisms also track requests for improvements to the software, allowing WESTPA users to give feedback to developers.

**4.2. Resources for Developers.** WESTPA has been designed from its inception to support collaborative community development of both the WE method and the WESTPA software that implements it. In addition to the use of an expressive and accessible programming language (Python), the strictly modular design of WESTPA (Figure 2) separates different aspects of the WE algorithm (and supporting code) into discrete units which are more readily understood and modified. These units of code are described and linked by well-defined application programming interfaces (APIs), which allow their reuse both in modifying the WESTPA code and in customizing it for a particular simulation. Documentation for these APIs is generally included in the code itself, ensuring that developers always have in-context access to the documentation they need to understand and use the WESTPA code.

WESTPA source code development is tracked using the Git version control system,[44] which encourages decentralized collaborative development while maintaining a record of prior versions of the code (of critical importance to scientific reproducibility). The WESTPA Git repositories are hosted publicly on the enormously popular and productive Github collaborative development site, which provides not only space to store the code but also mechanisms for tracking bugs and feature requests and an easily accessible forum for discussing proposed modifications to the code prior to their incorporation. WESTPA includes unit tests (currently coordinated with the *nose* testing framework for Python) which demonstrate the correctness of key algorithms; additional regression tests provide a means of ensuring that continued development does not "break" discrete portions of the code that formerly worked correctly. Further, the example simulations distributed with WESTPA (described above) function as integration (functional) tests, ensuring that the WESTPA package works correctly as a whole.

Of particular note for developers of WE methods are the modules and APIs for parallelization and creating command-line tools for working with WESTPA simulations. The parallelization API is fully general and maps cleanly onto the Python programming language, encouraging its use not only in WESTPA but also in any Python script or application needing a framework for multiprocessing and parallel task distribution. The command-line tool API facilitates the creation of WESTPA analysis programs and includes mechanisms for easily incorporating parallelization, access to WESTPA HDF5 files, and key analysis patterns into a user-created script. The command-line tool API also provides facilities for seamless code profiling, allowing a developer to determine where (and why) an analysis script or simulation routine is running most slowly and therefore could be optimized for enhanced performance.

Development of software does not usually proceed in isolation, as programmers should (and frequently must) confer with each other to make informed choices about how best to fix a bug or implement a new feature. In addition to the issue tracking and commenting facilities provided by Github, we provide a WESTPA developers' mailing list (currently hosted on Google Groups) to encourage discussion among developers. Further, information useful for those modifying (or interested in modifying) the WESTPA code is provided on the WESTPA Wiki (https://chong.chem.pitt.edu/wewiki/); this includes guidelines for how best to write code for eventual inclusion in WESTPA along with instructions for contributing changes back to the WESTPA community.

We note in passing that several of the tools and techniques discussed above (including use of version control, code modularization and reuse, unit testing, profiling, and embedded documentation) have recently been discussed by Wilson et al. as "best practices" for scientific software development.[45]

## 5. EXAMPLE: BISTABLE DIMER IN A DENSE WCA FLUID

During the development of WESTPA, the software package was validated for a number of different systems of varying complexity, in which direct comparisons were made between WE simulations and results derived from long-time scale brute force trajectories.[19,23,27] Here we consider in more explicit detail how WESTPA can be used to calculate dynamic and distributional quantities from a molecular system, focusing on key elements of a typical workflow and highlighting important capabilities provided by the software package.

To that end, we examine a bistable dimer immersed in a dense bath of particles[46−48] that interact via the Weeks−Chandler−Andersen (WCA) soft repulsive potential.[49] The dimer consists of a pair of particles interacting via a double-well potential with minima corresponding to a compact and extended conformation, separated by a 5 $k_BT$ barrier (Figure 5), described by the potential

$$U_{\text{bond}}(r) = h\left[1 - \frac{(r - r_0 - s)^2}{s^2}\right]^2 \tag{1}$$

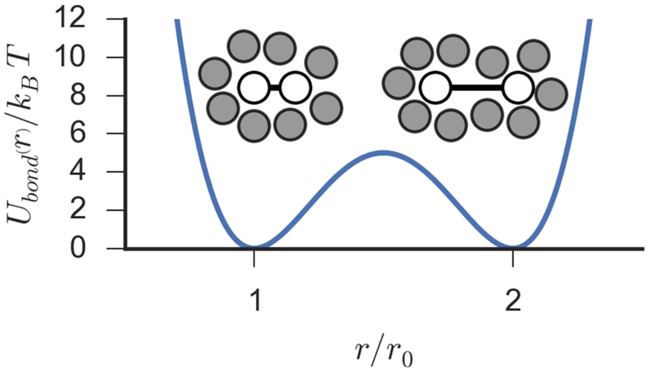


**Figure 5.** Bistable dimer potential. The bond potential between the two particles comprising the dimer (eq 1) is shown with compact ($r/r_0$ = 1) and extended ($r/r_0$ = 2) metastable states separated by a 5 $k_BT$ barrier. A cartoon of each state is shown above its respective potential minima, with the solvating WCA fluid depicted as gray circles.

where $r$ is the interatomic distance between particles, $h = 5k_BT$, $r_0 = r_{\text{WCA}}$, and $s = r_{\text{WCA}}/2$, where $r_{\text{WCA}} = 2^{1/6}\sigma$. The dimer is solvated in a dense bath of particles that interact with each other and the dimer via the WCA potential[49]

$$U_{\text{WCA}}(r) = \begin{cases} 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] + \varepsilon, & r < r_{\text{WCA}} \\ 0, & r \geq r_{\text{WCA}} \end{cases} \tag{2}$$

where $m$ = 39.9 amu, $\sigma$ = 3.4 Å, and $\varepsilon$ = 120 $k_BT$.

The system considered here contains 216 particles prepared at a reduced density of $\rho\sigma^3$ = 0.96 and was prepared using the custom scripts provided in ref 48. All trajectories were propagated using OpenMM 6.1[31] using Langevin dynamics at a reduced temperature of $k_BT/\varepsilon$ = 0.824 with a time step of $\delta t$ = 0.002$\tau$ (4.3 fs) and a collision rate of $\tau^{-1}$, where $\tau = (\sigma^2 m/\varepsilon)^{1/2}$. A long conventional brute force simulation, 2.5 × $10^8$ $\delta t$ in length, was generated as a reference, with coordinates recorded every 250 time steps.

The WE simulation was initiated from two conformations, one with a compact and the other with an extended dimer. Each was given a nonequilibrium weight of 0.8 and 0.2, respectively, requiring that the system relax to the correct equilibrium distribution during the simulation. The distance separating the two particles in the dimer, $r$, was used as the progress coordinate to enhance sampling of transitions between the two states. Bins 0.1 Å in width, spanning the interval $r$ = [3,8] Å, with a target number of 12 trajectories per bin were specified in the WESTPA system driver file. The WE resampling frequency was set to 250 $\delta t$.

In the accompanying source code for this example (*lib/examples/wca-dimer_openmm* in the WESTPA Github repository), we provide two implementations of how a user can specify the method by which WESTPA propagates the segments of each trajectory. In the first, we use OpenMM's internal API via the provided Python wrapper to write a custom propagator that makes direct calls to the low-level instructions that step the system forward in time and then extract the updated coordinates directly into memory without an intermediate file I/O call. In the second, we use the generic *executable* propagator provided by WESTPA to call a user defined shell script that runs an OpenMM executable to propagate dynamics with the appropriate initial coordinates and velocities. The same script then parses the outputted trajectory files to assign the progress coordinate. WESTPA defines and makes accessible a number of environment variables on a per-segment basis that can be used to configure the run and return required results to the WESTPA master process. This latter mode is more typical for programs that provide an executable (e.g., *pmemd* in Amber or *mdrun* in Gromacs). Both approaches give equivalent results, and only results from the custom propagator are shown here. For this system, which contains only 216 atoms, the dynamics are inexpensive to propagate relative to the initialization and file I/O. In this regime, the custom driver incurs a much smaller overhead since the system is persistent in memory for each worker process and only needs to be initiated once at the start of the simulation rather than at the start of every trajectory segment. Additionally, the progress coordinate is calculated using data in-memory for the custom propagator, rather than by processing a file on disk. For the hardware configuration we used, the custom propagator required approximately 15 s per iteration (wallclock time), whereas the executable propagator required nearly 3 min. The discrepancy between the two approaches will be negligible for larger or more detailed systems where the cost of propagating the dynamics dominates the overall simulation time.

After defining the bin space, propagation protocol and necessary input files, the *w_init* program initializes the WE simulation from the predefined set of initial states and prepares the first set of segments to be run. The *w_run* program is then used to perform the WE sampling. This program coordinates running the trajectory segments at each iteration, resampling the trajectory ensemble, and coordinating the distributed computational resources when run with a parallel work manager. Here, the calculation was distributed over 4 NVIDIA GPUs on a single node using the *processes* work manager. Progress of the WE simulation was monitored both by examining the log file, which provides per-iteration metrics such as the number of occupied bins, elapsed wallclock time, and statistics related to the distribution of weight on a per-segment and per-bin basis. Additionally the HDF5 output file generated by WESTPA along with the files generated specifically by the dynamics engine were periodically analyzed using programs included in the WESTPA analysis suite.

Here we used the *w_pdist* tool to calculate the probability distribution for finding the system with a dimer extension length of $r$. This distribution is shown in Figure 6 and is directly

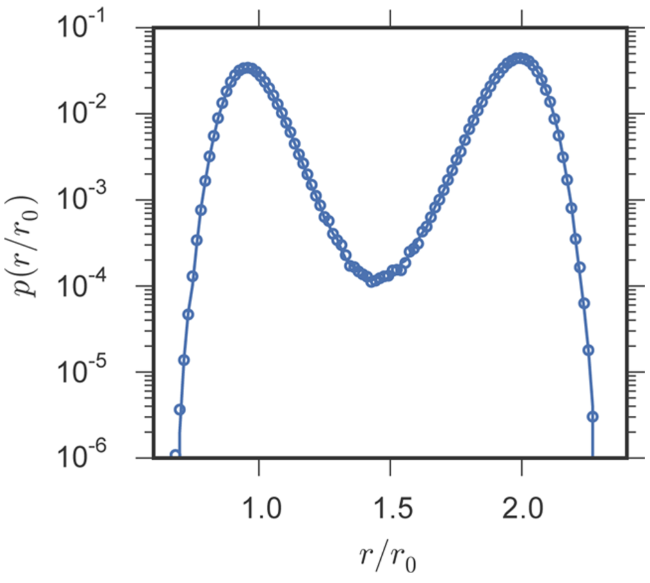


**Figure 6.** Probability distribution of the bistable dimer separation distribution obtained from a long brute force trajectory (line) compared to the same quantity calculated from a WE simulation (open circles).

compared to the distribution obtained from a long brute force simulation. The logarithmic scale highlights the close agreement both in the metastable basins at $r/r_0$ = 1 and 2 but also in the energetically less accessible extension distances in the barrier region and in the tails of the distribution. Although here we are examining the distribution along the progress coordinate used in the simulation, *w_pdist* can also be simply configured to use other data sets stored in the main WESTPA HDF5 file or constructed data sets built on-the-fly from any auxiliary data generated during the simulation. The *w_pdist* tool can be paired with *plothist*, to quickly visualize any calculated distribution or the raw data it generates can be used to compose custom plots like Figure 6.

The *w_kinetics* and *w_kinavg* tools are used to calculate the rates of transitioning between the compact and extended conformations of the dimer. The former tool determines kinetic quantities by tracing the pathways of individual trajectories as they move between user-defined states, and the latter tool calculates the average rates and associated errors. Here we define the compact state as any conformation of the system with $r/r_0 < 1.1$ and the extended as $r/r_0 > 1.83$ The evolution of the rates over the course of the WE simulation is shown in Figure 7, and the rates are again compared to the same quantities calculated from a long brute force trajectory.

The complete analysis shown in Figures 6 and 7, along with the generation of rudimentary plots for fast inspection can be accomplished with the set of commands shown in Table 2.

In Table 2, `$WEST_ROOT` represents the directory which contains the WESTPA software. The figures generated by *plothist* and *ploterr* are shown as Supporting Information Figures S1−S5.

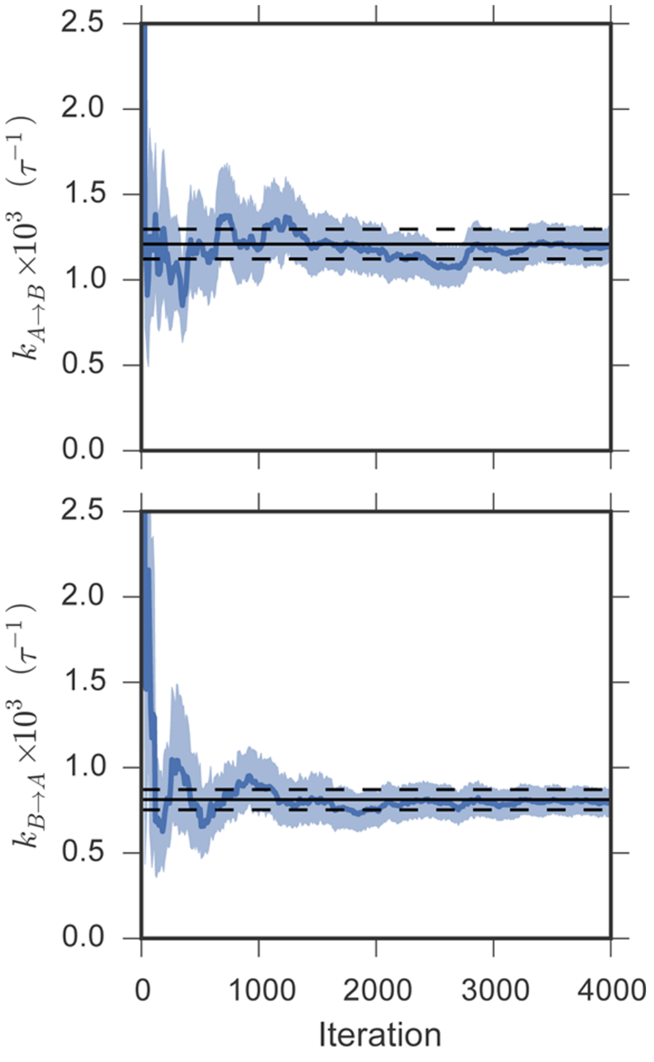


**Figure 7.** Evolution of the rate of interconversion from the compact to extended conformations ($k_{A\to B}$) of the bistable dimer (upper panel) and from the extended to compact conformation ($k_{B\to A}$) (lower panel) calculated from a WE simulation with the w_kinetics and w_kinavg tools. The solid blue line shows the mean rate calculated from the last 50% of the WE iterations at a given time point, with the region filled in blue delineating the 95% confidence interval. The rates calculated from a long brute force trajectory are shown as solid black lines, with the brute force 95% confidence interval falling between the black dashed lines.

## 6. SUMMARY

WESTPA is an open-source, interoperable, high-performance software implementation of the weighted ensemble (WE) path sampling approach, which increases the efficiency of simulating rare events while maintaining rigorous kinetics. The WE approach permits simulation of equilibrium, nonequilibrium steady-state, and relaxation processes, and WESTPA supports all of these simulation types. The WESTPA software includes facilities for running simulations in serial and parallel and scales from single-core workstations to the thousands of cores available on supercomputing clusters. WESTPA can be used with any stochastic dynamics engines (including existing simulation packages or custom dynamics propagation routines) at any scale (e.g., atomistic, molecular, or cellular models). The WESTPA package also provides a toolkit of optimized programs for various types of simulation analysis, including assessment of convergence and error estimation for key observables like state populations and rate constants. Extensive resources (currently listed at https://chong.chem.pitt.edu/WESTPA) are available for researchers to facilitate the incorporation of WE simulations into their work, including searchable online user and developer forums as well as tutorials focused on problems and/or types of simulations that are of wide interest. The portability of WESTPA, combined with its interoperability and optimized scaling, are essential for advancing the ability of researchers to address challenging problems requiring rare event sampling.

## ■ ASSOCIATED CONTENT

### Ⓢ Supporting Information

Description of the analysis of the WCA dimer simulation (including example input and output from the WESTPA analysis tools) and detailed descriptions of the progress coordinate binning facilities (including example code). This material is available free of charge via the Internet at http://pubs.acs.org.

## ■ AUTHOR INFORMATION

**Corresponding Author**

*E-mail: ltchong@pitt.edu.

**Notes**

The authors declare no competing financial interest.

**Table 2**

| command |
|---|

```
$WEST_ROOT/bin/w_assign --states-from-function system.gen_state_labels
$WEST_ROOT/bin/w_pdist -o pdist.h5
$WEST_ROOT/bin/plothist average --first-iter 1000 --log10 pdist.h5

$WEST_ROOT/bin/w_kinetics trace
$WEST_ROOT/bin/w_kinavg trace -e cumulative --window-frac .5 --step-iter 10
$WEST_ROOT/bin/ploterr kinavg
```

## ■ ACKNOWLEDGMENTS

This work was supported by NSF CAREER award MCB-0845216 to L.T.C.; the Office of the Provost at Drake University (M.C.Z.); National Institutes of Health (NIH) Grant No. R01-GM089740 (M.G.); NIH Grant No. T32-DK061296 (J.L.A.); NIH Grant No. P41 GM103712 and NSF Grant No. MCB-1119091 (D.M.Z.); University of Pittsburgh Arts & Sciences Graduate, Department of Chemistry Graduate Excellence, and University of Pittsburgh Andrew Mellon Fellowships (M.C.Z.); and a University of Pittsburgh Brackenridge Fellowship underwritten by the United States Steel Foundation (J.W.K.). Computational resources were provided by NSF XSEDE award MCB-100109, NSF MRI award CNS-1229064, and by the Center for Simulation and Modeling at the University of Pittsburgh.

## ■ REFERENCES

(1) Faradjian, A. K.; Elber, R. *J. Chem. Phys.* **2004**, *120*, 10880.
(2) Elber, R.; Roitberg, A.; Simmerling, C.; Goldstein, R.; Li, H.; Verkhivker, G.; Keasar, C.; Zhang, J.; Ulitsky, A. *Comput. Phys. Commun.* **1995**, *91*, 159.
(3) Dellago, C.; Bolhuis, P. G.; Csajka, F.; Chandler, D. *J. Chem. Phys.* **1998**, *108*, 1964.
(4) Brooks, B. R.; Brooks, C. L., III; Mackerell, A. D., Jr.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. *J. Comput. Chem.* **2009**, *30*, 1545.
(5) Kratzer, K.; Berryman, J. T.; Taudt, A.; Zeman, J.; Arnold, A. *Comput. Phys. Commun.* **2014**, *185*, 1875.
(6) Allen, R. J.; Warren, P.; ten Wolde, P. R. *Phys. Rev. Lett.* **2005**, *94*, 018104.
(7) Beauchamp, K. A.; Bowman, G. R.; Lane, T. J.; Maibaum, L.; Haque, I. S.; Pande, V. S. *J. Chem. Theory Comput.* **2011**, *7*, 3412.
(8) Senne, M.; Trendelkamp-Schroer, B.; Mey, A. S. J. S.; Schütte, C.; Noé, F. *J. Chem. Theory Comput.* **2012**, *8*, 2223.
(9) Chodera, J. D.; Swope, W. C.; Pitera, J. W.; Dill, K. A. *Multiscale Model. Simul.* **2006**, *5*, 1214.
(10) Bowman, G. R.; Ensign, D. L.; Pande, V. S. *J. Chem. Theory Comput.* **2010**, *6*, 787.
(11) Doerr, S.; De Fabritiis, G. *J. Chem. Theory Comput.* **2014**, *10*, 2064.
(12) Huber, G. A.; Kim, S. *Biophys. J.* **1996**, *70*, 97.
(13) Rojnuckarin, A.; Kim, S.; Subramaniam, S. *Proc. Natl. Acad. Sci. U.S.A.* **1998**, *95*, 4288.
(14) Rojnuckarin, A.; Livesay, D. R.; Subramaniam, S. *Biophys. J.* **2000**, *79*, 686.
(15) Zhang, B. W.; Jasnow, D.; Zuckerman, D. M. *Proc. Natl. Acad. Sci. U.S.A.* **2007**, *104*, 18043.
(16) Zhang, B. W.; Jasnow, D.; Zuckerman, D. M. *J. Chem. Phys.* **2010**, *132*, 054107.
(17) Bhatt, D.; Zhang, B. W.; Zuckerman, D. M. *J. Chem. Phys.* **2010**, *133*, 014110.
(18) Bhatt, D.; Zuckerman, D. M. *J. Chem. Theory Comput.* **2010**, *6*, 3527.
(19) Zwier, M. C.; Kaus, J. W.; Chong, L. T. *J. Chem. Theory Comput.* **2011**, *7*, 1189.
(20) Adelman, J. L.; Dale, A. L.; Zwier, M. C.; Bhatt, D.; Chong, L. T.; Zuckerman, D. M.; Grabe, M. *Biophys. J.* **2011**, *101*, 2399.
(21) Bhatt, D.; Bahar, I. *J. Chem. Phys.* **2012**, *137*, 104101.
(22) Donovan, R. M.; Sedgewick, A. J.; Faeder, J. R.; Zuckerman, D. M. *J. Chem. Phys.* **2013**, *139*, 115105.
(23) Adelman, J. L.; Grabe, M. *J. Chem. Phys.* **2013**, *138*, 044105.
(24) Dickson, A.; Brooks, C. L., III *J. Phys. Chem. B* **2014**, *118*, 3532.
(25) Abdul-Wahid, B.; Feng, H.; Rajan, D.; Costaouec, R.; Darve, E.; Thain, D.; Izaguirre, J. A. *J. Chem. Inf. Model.* **2014**, 3033.
(26) Dickson, A.; Mustoe, A. M.; Salmon, L.; Brooks, C. L. *Nucleic Acids Res.* **2014**, *42*, 12126.
(27) Suárez, E.; Lettieri, S.; Zwier, M. C.; Stringer, C. A.; Subramanian, S. R.; Chong, L. T.; Zuckerman, D. M. *J. Chem. Theory Comput.* **2014**, *10*, 2658.
(28) Pronk, S.; Pall, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; van der Spoel, D.; Hess, B.; Lindahl, E. *Bioinformatics* **2013**, *29*, 845.
(29) Case, D. A.; Cheatham, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. *J. Comput. Chem.* **2005**, *26*, 1668.
(30) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781.
(31) Eastman, P.; Friedrichs, M. S.; Chodera, J. D.; Radmer, R. J.; Bruns, C. M.; Ku, J. P.; Beauchamp, K. A.; Lane, T. J.; Wang, L.-P.; Shukla, D.; Tye, T.; Houston, M.; Stich, T.; Klein, C.; Shirts, M. R.; Pande, V. S. *J. Chem. Theory Comput.* **2013**, *9*, 461.
(32) Elcock, A. H. *PLoS Comput. Biol.* **2006**, *2*, e98.
(33) Frembgen-Kesner, T.; Elcock, A. H. *J. Chem. Theory Comput.* **2009**, *5*, 242.
(34) Blinov, M. L.; Faeder, J. R.; Goldstein, B.; Hlavacek, W. S. *Bioinformatics* **2004**, *20*, 3289.
(35) Kerr, R. A.; Bartol, T. M.; Kaminsky, B.; Dittrich, M.; Chang, J.-C. J.; Baden, S. B.; Sejnowski, T. J.; Stiles, J. R. *SIAM J. Sci. Comput.* **2008**, *30*, 3126.
(36) Frenkel, D.; Smit, B. *Understanding molecular simulation: from algorithms to applications*, 2nd ed.; Academic Press: San Diego, 2002.
(37) Basconi, J. E.; Shirts, M. R. *J. Chem. Theory Comput.* **2013**, *9*, 2887.
(38) Oliphant, T. E. *Comput. Sci. Eng.* **2007**, *9*, 10.
(39) Perez, F.; Granger, B. E.; Hunter, J. D. *Comput. Sci. Eng.* **2011**, *13*, 13.
(40) van der Walt, S.; Colbert, S. C.; Varoquaux, G. *Comput. Sci. Eng.* **2011**, *13*, 22.
(41) Behnel, S.; Bradshaw, R.; Citro, C.; Dalcin, L.; Seljebotn, D. S.; Smith, K. *Comput. Sci. Eng.* **2011**, *13*, 31.
(42) Collette, A. *Python and HDF5*; O'Reilly: Sebastopol, CA, 2014.
(43) Zhang, B. W.; Jasnow, D.; Zuckerman, D. M. *J. Chem. Phys.* **2007**, *126*, 074504.
(44) Loeliger, J.; McCullough, M. *Version Control Using Git*, 2nd ed.; O'Reilly: Sebastopol, CA, 2012.
(45) Wilson, G.; Aruliah, D. A.; Brown, C. T.; Chue Hong, N. P.; Davis, M.; Guy, R. T.; Haddock, S. H. D.; Huff, K. D.; Mitchell, I. M.; Plumbley, M. D.; Waugh, B.; White, E. P.; Wilson, P. *PLoS Biol.* **2014**, *12*, e1001745.
(46) Dellago, C.; Bolhuis, P. G.; Chandler, D. *J. Chem. Phys.* **1999**, *110*, 6617.
(47) Straub, J. E.; Borkovec, M.; Berne, B. J. *J. Chem. Phys.* **1988**, *89*, 4833.
(48) Nilmeier, J. P.; Crooks, G. E.; Minh, D. D. L.; Chodera, J. D. *Proc. Natl. Acad. Sci. U.S.A.* **2011**, *108*, E1009.
(49) Weeks, J. D.; Chandler, D.; Andersen, H. C. *J. Chem. Phys.* **1971**, *54*, 5237.