SSAGES: Software Suite for Advanced General Ensemble Simulations

Hythem Sidky, Yamil J. Colón, Julian Helfferich, Benjamin J. Sikora, Cody Bezik, Weiwei Chu, Federico Giberti, Ashley Z. Guo, Xikai Jiang, Joshua Lequieu, Jiyuan Li, Joshua Moller, Michael J. Quevillon, Mohammad Rahimi, Hadi Ramezani-Dakhel, Vikramjit S. Rathee, Daniel R. Reid, Emre Sevgen, Vikram Thapar, Michael A. Webb, Jonathan K. Whitmer, and Juan J. de Pablo

Citation: The Journal of Chemical Physics **148**, 044104 (2018);
View online: https://doi.org/10.1063/1.5008853
View Table of Contents: http://aip.scitation.org/toc/jcp/148/4
Published by the American Institute of Physics

SSAGES: Software Suite for Advanced General Ensemble Simulations

Hythem Sidky,[1,a)] Yamil J. Colón,[2,3,a)] Julian Helfferich,[2,4] Benjamin J. Sikora,[1] Cody Bezik,[2] Weiwei Chu,[2] Federico Giberti,[2] Ashley Z. Guo,[2] Xikai Jiang,[2] Joshua Lequieu,[2] Jiyuan Li,[2] Joshua Moller,[2] Michael J. Quevillon,[1] Mohammad Rahimi,[2] Hadi Ramezani-Dakhel,[2,5] Vikramjit S. Rathee,[1] Daniel R. Reid,[2] Emre Sevgen,[2] Vikram Thapar,[2] Michael A. Webb,[2,3] Jonathan K. Whitmer,[1] and Juan J. de Pablo[2,3]

[1]*Department of Chemical and Biomolecular Engineering, University of Notre Dame, Notre Dame, Indiana 46556, USA*
[2]*Institute for Molecular Engineering, University of Chicago, Chicago, Illinois 60637, USA*
[3]*Institute for Molecular Engineering and Materials Science Division, Argonne National Laboratory, Lemont, Illinois 60439, USA*
[4]*Steinbuch Center for Computing, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany*
[5]*Department of Biochemistry and Molecular Biology, University of Chicago, Chicago, Illinois 60637, USA*

(Received 10 October 2017; accepted 26 December 2017; published online 22 January 2018)

Molecular simulation has emerged as an essential tool for modern-day research, but obtaining proper results and making reliable conclusions from simulations requires adequate sampling of the system under consideration. To this end, a variety of methods exist in the literature that can enhance sampling considerably, and increasingly sophisticated, effective algorithms continue to be developed at a rapid pace. Implementation of these techniques, however, can be challenging for experts and non-experts alike. There is a clear need for software that provides rapid, reliable, and easy access to a wide range of advanced sampling methods and that facilitates implementation of new techniques as they emerge. Here we present SSAGES, a publicly available Software Suite for Advanced General Ensemble Simulations designed to interface with multiple widely used molecular dynamics simulations packages. SSAGES allows facile application of a variety of enhanced sampling techniques—including adaptive biasing force, string methods, and forward flux sampling—that extract meaningful free energy and transition path data from all-atom and coarse-grained simulations. A noteworthy feature of SSAGES is a user-friendly framework that facilitates further development and implementation of new methods and collective variables. In this work, the use of SSAGES is illustrated in the context of simple representative applications involving distinct methods and different collective variables that are available in the current release of the suite. The code may be found at: https://github.com/MICCoM/SSAGES-public. https://doi.org/10.1063/1.5008853

INTRODUCTION

The past several decades have seen molecular simulations emerge as an essential tool for molecular research.[1] Through the formalism provided by statistical mechanics, molecular simulations are able to relate phenomena at small length scales to experimental observables such as temperature, pressure, and other thermodynamic quantities. The validity of such calculations relies on proper sampling of the system under consideration, which is often achieved by running a simulation over a sufficiently long period of time. For many systems of interest, however, the time necessary to explore the phase space is prohibitively long, and one must resort to advanced sampling techniques.

Conventional molecular dynamics (MD) or Monte Carlo (MC) simulations generally encounter severe limitations in systems that exhibit rugged energy landscapes, which are characterized by multiple metastable states that are separated by large energy barriers. Examples include proteins,[2–8] small molecules in viscous solvents or membranes,[9–14] and phase transitions,[15–20] to name a few of the classes of simulation scenarios that one may encounter in a variety of disciplines.[21,22] Enhanced sampling techniques seek to eliminate or circumvent such barriers, thereby providing a means to recover thermodynamic quantities for systems that would be difficult to study through a direct MD approach.

A second limitation of traditional MD or MC simulations is that they only provide direct access to structure and "mechanical" properties such as the temperature or pressure. Thermodynamic quantities such as the entropy or the free energy can only be determined by relying on external methods that involve integrals of mechanical quantities over specific thermodynamic variables and pathways. The essence of an enhanced sampling method is to introduce an external bias, usually in the form of an external potential or an

[a)]H. Sidky and Y. J. Colón contributed equally to this work.

external force, which promotes the exploration of an activated process, described with a set of collective variables (CVs). Such collective variables are functions of the atomic coordinates of the system. Given a set of CVs and an enhanced sampling method, one can usually recover thermodynamic and kinetic quantities in the space that the CVs define, such as the free energy, entropy, or reaction rates. As explained in what follows, advanced biasing techniques and CVs provide information relating to the entropy or free energy as a byproduct of the simulation, thereby providing a strong incentive to implement such methods in order to fully exploit a simulation.

A variety of techniques have been developed to address these challenges. Examples include configurational bias sampling,[23–25] parallel tempering or replica exchange,[21,22,26–28] umbrella sampling,[29] Wang-Landau or density-of-states sampling, expanded ensemble methods,[30] chain-of-states methods,[31–33] adaptive biasing force (ABF),[34] metadynamics (MTD),[35–37] basis function sampling (BFS),[38,39] and others.[40–45] Similarly, typical CVs on which such techniques may be used include separation distance, dihedral angles, and coordination number.[46] Implementing these methods and CVs can be challenging and requires access to quantities defined in the MD software that are usually inaccessible to common users, e.g., Cartesian coordinates, momenta, forces, or energies. As a result, a majority of simulations to date have relied on standard, unbiased molecular dynamics simulations or have used the simplest (and easiest to implement) biasing techniques and CVs. Recent efforts have sought to remedy this situation by providing generalized, open-source software that can interface with multiple molecular dynamics engines and bias a trajectory through the use of various enhanced sampling methods and CVs. Examples include plug-ins such as PLUMED,[47,48] PyRETIS,[49] open path sampling,[50] and various collective variables modules.[46,51] While collectively these codes offer a wide variety of sampling algorithms, individual codes generally implement a few select methods. Path sampling and adaptive bias methods, for example, are offered separately. There is a need for a unified framework that allows for a seamless transition between types of methods and techniques, thereby enabling comparison between them. This capability will in turn allow users to make informed decisions about what sampling approach is most effective for a particular system of interest and facilitate both free energy and path sampling calculations.

In this work, we present SSAGES, a Software Suite for Advanced General Ensemble Simulations. SSAGES is a free, open-source software package written in C++ that allows users to easily apply enhanced sampling techniques to any molecular system of interest. SSAGES currently interfaces with LAMMPS,[52] GROMACS,[53,54] OpenMD,[55] Qbox,[56] and DASH.[57] SSAGES includes enhanced sampling techniques such as: umbrella sampling, steered MD, metadynamics, forward flux sampling (FFS), nudged elastic band (NEB), finite temperature string (FTS), swarms of trajectories (SoT), adaptive biasing force, basis function sampling, and artificial neural network (ANN) sampling. The collective variables available in SSAGES include: angle, dihedral, distance between centers of mass, Rouse modes, secondary structure, box volume, gyration tensor, and coordination number. Importantly, SSAGES has been created in a highly modular manner, allowing for easy implementation of new methods and collective variables or modification of existing ones. Herein we describe the overall structure of the code, as well as the methods that have been implemented and tested within the package. We also provide guidelines to implement new methods and collective variables using the SSAGES framework.

DESCRIPTION OF THE CODE

Figure 1 summarizes the workflow used in SSAGES to perform an enhanced sampling calculation. Input files are defined using the JSON (JavaScript Object Notation) syntax, which is a lightweight human and machine-readable file format with native support across many programming and scripting languages. This facilitates easy scripting and automated workflows without the burden of implementing custom parsers by the end user. Communication between SSAGES and the engine takes place through a lightweight adapter, termed "hook," which bridges engine-agnostic methods and CVs implemented within SSAGES and the integration of the equations of motion by the underlying MD engine.

Once a simulation is initialized by the MD engine of choice, the relevant atomic coordinates, forces and momenta, the box virial tensor, and other thermodynamic quantities are copied by the hook into a SSAGES "snapshot" object, which is then exposed to the SSAGES framework. The collective variables of interest are calculated, and then a user specified method makes the necessary changes to the snapshot based

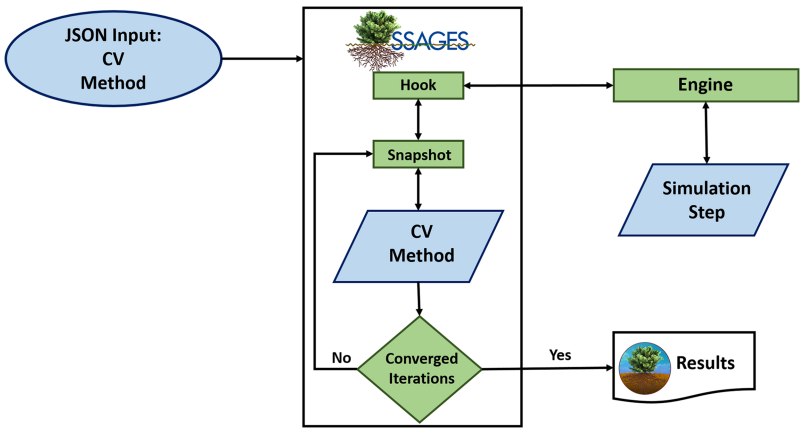

FIG. 1. Flowchart of running a simulation using SSAGES. A JSON input file defines collective variables and sampling methods to be used. SSAGES then starts the simulation with a given engine and updates quantities according to the method until convergence or a prescribed number of iterations are met.

on the state of the system and the collective variables. Other modules declared by the user that "observe" the simulation are also invoked. These may include logging or histogram modules to aid the user in post-processing or collect additional useful information. The final modified snapshot is then used by the hook to update the appropriate data within the engine, after which a simulation step is completed. This iterative process is repeated until a predefined number of steps are executed or a method converges.

As described above and illustrated in Fig. 1, SSAGES routines are engine-agnostic. The only engine-specific code that needs to be written is the hook, which must integrate into engine components that expose the necessary data structures. Strict compartmentalization ensures that the only objects and functions exposed to the remainder of the SSAGES code are independent of engine implementation. This setup allows for new methods to be developed using a single unified programming interface, fully decoupled from the intricacies of any particular engine. Furthermore, exposing all the methods available within SSAGES to a new engine simply requires the development of a single additional hook. This is particularly useful for users who may want to rely on their own engines. The open source nature of this project welcomes such public contributions to the base code, which we envision will rapidly accelerate the utilization of enhanced sampling techniques in the field. This construct also simplifies the task of maintaining support with new releases of the existing MD codes since any changes are reflected only within the hook.

Another important aspect of SSAGES is the straightforward implementation of new methods and collective variables (Fig. 2). The addition of a new method requires that a new class, which derives from the base "Method" class, be programmed. There are four functions that every method must implement: "PreSimulation," which is invoked after the construction of the simulation engine but before integration begins, "PostIntegration," the primary function call that is invoked every step after forces are calculated by the engine—but before atomic coordinates are updated, "PostSimulation," invoked after all steps have been completed prior to the shutdown of the engine, and "Build," which constructs a new method object from a parsed JSON input. The first three functions are provided with a pointer to a snapshot that may be manipulated and a reference to a CV manager that provides access to initialized CVs with selection rules.

Method	Collective Variable (CV)
Pre-Simulation	*Initialize*
Allocate memory	Declare variables
Initialize grid(s)	Pre-calculate reusable quantities
Open files	
	Evaluate
Post-Integration	
	Calculate CV
Get information from Snapshot	Calculate gradient of CV
Perform calculation	Calculate virial contribution
Update Snapshot	
Print data	*Build*
Post-Simulation	Get input parameters from JSON
	Semantic validation
Post-processing	Initialize CV
Close files	
Build	
Get input parameters from JSON	
Semantic validation	
Initialize method	

FIG. 2. Required functions for a method (left) and a collective variable (right). The pre-simulation section is used to initialize internal variables, and the post-simulation section is used for post-processing and to close any files that may have been used during the simulation. The post-integration section is where the heart of the method is located. This section is called after every step and is responsible for the method calculations. Both methods and CVs are constructed from JSON objects in the build call.

Although there are no strict requirements for what is contained within each of the aforementioned functions, it is customary that output file setup, memory allocation, and method-specific data structures be initialized during pre-simulation. The biasing or other system manipulation occurs during post-integration. This typically involves the addition of force to atoms based on CVs or the direct manipulation of atomic coordinates and summing the virial tensor contribution for constant pressure simulations. For methods that require additional post-processing, this task is usually reserved for the PostSimulation function.

Since different methods often rely on similar building blocks, SSAGES provides convenient access to implementations of various reusable components. For example, many methods either directly depend upon or are accelerated by grids that discretize CV space. SSAGES offers two efficient grid implementations that are templated to support arbitrary data types. One is tailored for use as a histogram, offering overflow bins, and the other is more general and better suited for storing forces. Both automatically handle periodic coordinates, support multidimensional interpolation, provide custom iterators, and include functions for saving/loading the grids to/from disk. Another example is related to the wide range of published string methods, which rely on slightly differing algorithms; a base string method class is available that implements a large portion of the necessary inter-process communication, greatly simplifying the implementation of other string techniques. Similarly, a reference forward flux sampling implementation is available that can be extended to support different "flavors" of forward flux sampling.

Like any method, a CV must derive from a base class and implement required functions. Because CVs are simply functions of system configurations, they are unable to manipulate snapshots and are given read-only access. Doing so limits the responsibility of a CV and in turn reduces the possibility of introducing undesirable errors. The "Initialize" function (Fig. 2) is used by a CV to ensure that all the participating atoms are available at runtime—this is the final part of a multi-tiered input validation process described later on. "Evaluate" is responsible for calculating the CV in addition to its gradient with respect to the 3N atomic Cartesian coordinates and 6 box degrees of freedom. For some CVs, calculating the latter term may be difficult and is only necessary if proper treatment in the isobaric ensemble is desired. The calculated CV value and gradient are passed on to the methods which may propagate forces along the CVs to the atoms using the chain rule. A CV may also optionally override a "GetDifference" function which allows a method to properly measure distances along periodic CVs.

An important consideration for the development of both methods and CVs is the parallel nature of modern molecular

dynamics codes. To support the large-scale simulation of thousands or millions of atoms, many engines rely on distributed computing strategies spanning multiple processors through domain and force decomposition. The necessary inter-process communication is often implemented using the Message Passing Interface (MPI) programming interface, which SSAGES handles seamlessly. Multi-threading using graphics processing units (GPUs) or OpenMP do not affect the availability of atomic information and do not alter the operation of SSAGES. In situations involving multiple processors, whether for a single system or multi-walker simulation, which SSAGES supports, global and intra-walker communicators are provided to methods and CVs. An additional MPI wrapper is also available which greatly simplifies inter-process communication. To minimize the need for the tedious MPI calls that are necessary to gather information distributed across multiple processors, the snapshot provides helper functions which map global atomic indices to local processor-specific IDs, gather the center of mass of an atom group across the necessary processors, and collect all atomic coordinates into a single array.

Performance is of utmost importance when performing advanced sampling simulations. Some amount of overhead is unavoidable, even with the most basic methods, simply due to the additional calculations that must be performed at every step. Additionally, elaborate CVs may involve computationally expensive gradients or require significant inter-process communication. While an arbitrary implementation of a method or CV is not guaranteed to be efficient, SSAGES was designed in such a way so as to minimize the fixed overhead and offer tools to accelerate calculations. The SSAGES hook and synchronization mechanism is very lightweight and does not rely on complex polymorphism or abstractions, thereby resulting in a measured overhead of 1%-2% across all tested engines. Thus, pairing a simple method and CV will not introduce a significant slowdown compared to an unbiased simulation. Furthermore, numerical arrays are built upon Eigen, a C++ numerical library, which provides linear algebra operations in a natural syntax and takes advantage of single instruction multiple data (SIMD) extensions on modern processors, leading to a substantial speedup over serial loops.[58] CV and method developers are encouraged to take advantage of SIMD parallelism when developing new classes.

To initialize a method or CV at runtime requires parsing of a JSON input file. SSAGES adopts a multi-tiered approach to validate the input. The integrated JSON parser is responsible for validating the overall structure and syntax of the input file. Gross syntax errors such as unclosed braces or unterminated quotes are caught immediately, and the user is informed. Basic data validation is provided for CVs and methods using the JSON schema draft 4 specification. This entails the specification of a JSON schema file which contains a "blueprint" or set of expectations by the developer for their object. Validation at this level includes data types, upper and lower bounds, string regular expression matching, required and optional fields, and enumerations. Once the schema is validated against the input file by the SSAGES engine, a C++ object containing the data is provided to the appropriate build function where a developer can perform semantic validation using advanced logic.

Input file

```
{
  "CVs":[
    {
      "type": "Distance",
      "group1": [1,2],
      "group2": [3,4]
    },
    {
      "type": "Distance",
      "group1": [5,6],
      "group2": [7,8]
    }
  ]
  "Methods":[
    {
      "type": "Umbrella",
      "cvs": [0,1],
      "ksprings" : [100,200],
      "centers" : [0.5,2]
    }
  ]
}
```

FIG. 3. JSON input file for performing umbrella sampling on two distance CVs.

However, the use of a schema greatly simplifies developers' efforts since they will have necessarily received the appropriate data and types. The object can be initialized using these data, and the final runtime validation, such as checks for the presence of atoms, can be performed at the pre-simulation stage.

Figure 3 shows a JSON input file to use a method (umbrella sampling) and a CV (distance between centers of mass). The input file specifies two distance CVs to be calculated. The first is the distance between the center of mass of particle 1 and 2 and the center of mass of particle 3 and 4. The second is the same but with particles 5 and 6 and particles 7 and 8. Then the input file identifies the method: umbrella sampling. It specifies that the method will operate on both CVs. The first will have a spring constant of 100 and a center of 0.5 while the second will have a spring constant of 200 and a center of 2. The units of these quantities will be consistent with the internal units of the engine of choice. Further instructions on how to implement new methods and CVs can be found in the documentation of the software.

OVERVIEW OF METHODS

In what follows, we only provide a brief description of the methods that are currently included in SSAGES. For more detailed explanations of the techniques, we refer the reader to the references contained within this section. We also note that the list of methods available in SSAGES continues to grow, and users interested in the latest release are referred to the code's website (https://github.com/MICCoM/SSAGES-public).

A wide range of techniques have been developed to improve sampling efficiency and allow the extraction of quantities of interest: free energies, transition pathways, and/or rates. One technique implemented in SSAGES is umbrella sampling. It helps to overcome energetic barriers and improve

sampling by applying a bias along a specified collective variable.[59] The bias often takes the form of a harmonic potential and is typically constant throughout a simulation. Usually, a series of umbrella-sampled simulations are performed and analyzed together using the weighted histogram analysis method (WHAM)[60,61] or umbrella integration.[62] In SSAGES, the target value of umbrella sampling can be made time-dependent, which is equivalent to the steered molecular dynamics method.[63–66]

To calculate free energies, SSAGES makes use of flat-histogram or density-of-states methods. These are powerful methods that calculate free energies by developing uniform distributions along expanded ensemble partition functions. More details on these methods can be found in a recent review by Singh, Chopra, and de Pablo.[30] Of the flat-histogram methods, SSAGES currently includes metadynamics,[67] basis function sampling (BFS),[38] and the recently proposed artificial neural network (ANN) sampling.[68] The main difference between these methods is how the bias is applied to achieve a uniform distribution. Within metadynamics, a history-dependent bias is accrued through the sequential addition of Gaussian biases to the collective variables (CVs) of interest, which acts to push the system away from previously visited states. As the bias accrues to the height of nearby features in the free energy surface (FES), the system's trajectory can escape a local free energy minimum and begin to explore other regions of CV space. After a sufficient number of biases have been applied, the total applied bias within a given region begins to oscillate around the negative of the free energy in that region. There are many variants of metadynamics: well-tempered, flux-tempered, transition-tempered, and adaptive Gaussians. Currently, SSAGES includes only standard metadynamics with fixed-shape hills.

Instead of dropping Gaussians, basis function sampling (BFS) relies on the summation of Kronecker deltas to bias a simulation. The Kronecker delta is approximated by the projection of a locally biased histogram to a truncated set of orthogonal basis functions. By projecting a basis set relative to a weight function, the method resolves the same properties as the Kronecker deltas but in a continuous and differentiable manner that lends itself to MD simulations. SSAGES currently constructs the basis set using Legendre and Chebyshev polynomials and Fourier series. Convergence of the method is based on the updates of the basis set coefficients. ANN sampling uses Bayesian-regularized neural networks to develop an ideal bias that is free from ringing and boundary artifacts associated with basis functions and metadynamics.[68] It is capable of rapidly conforming to varied free energy landscapes by interpolating undersampled regions and extrapolating bias into new unexplored areas.

A variant of the flat-histogram methods is adaptive biasing force, or ABF, which also seeks to sample uniformly over CV space. Unlike other methods, however, ABF does not obtain a direct estimate of the free energy surface but rather the derivative of the free energy in the CV direction: the generalized force on that CV by the system. In practice, this translates to binning coordinates in CV space with an instantaneous estimation of the force. This instantaneous estimate fluctuates around the true, global free energy derivative at that point, but the average quickly converges to the real value. Then, the free energy derivatives can be integrated to determine the free energy surface. More detailed explanations of the method can be found in Refs. 69 and 70. SSAGES follows the implementation of Darve and co-workers.[69]

In addition to free energy methods, SSAGES provides methods to identify transition pathways between metastable states. These chain-of-states methods discretize a proposed pathway that connects two states into images or nodes. As sampling proceeds, the pathway is iteratively evolved by adjusting the position of these nodes. There are many variants, and SSAGES currently includes three: nudged elastic band (NEB), finite temperature string (FTS), and swarm of trajectories (SoT). The NEB method involves the evolution of a series of images connected by a spring interaction (hence the "elastic" nature of the band).[71–73] The "nudged" nature refers to a force projection that ensures the images do not all fall into the metastable states as they evolve. This projection is accomplished by using the parallel portion of the spring force and the perpendicular portion of the true force. In this way, the spring forces act similarly to reparameterization schemes common to string methods. When the images are in Cartesian space, NEB converges to the minimum energy path (MEP). When they are in CV space, which requires the introduction of an M-tensor, NEB converges to the minimum free energy path (MFEP). FTS converges to the principal curve, which intersects each of the perpendicular hyperplanes that it passes through at the expected value of each hyperplane. Instead of sampling along each hyperplane belonging to each node along the string, SSAGES employs the Voronoi approximation introduced by Vanden-Eijnden and Venturoli.[74] Each image is free to explore CV space within its own Voronoi cell: the region in CV space where any point is closer to its node than any other node along the string. The string is then evolved toward the running averages in CV space for each image along the string. When the string is updated, the nodes are redistributed such that they are an equal arc length apart and a cubic spline is used to ensure that the curve is smooth. Finally, SoT launches a large number (a swarm) of unrestrained trajectories from each image on the string and estimates the average drift of the collective variables over the swarm. The images along the string are then evolved along the average drift. As a result, SoT yields the most probable transition pathway (MPTP). Similar to FTS, the string is reparameterized such that the nodes along the string are an equal arc length apart. Additional details on SoT may be found in the literature.[75]

Lastly, another type of transition pathway method available in SSAGES is forward flux sampling (FFS). FFS produces as output the rate and the trajectories for a system that goes from an initial state A to a final state B. This is done by choosing several intermediate states or interfaces between the states such that the energy barriers between adjacent interfaces are readily surmountable in a typical simulation. Using configurations at each interface, attempts are made to arrive at the next interface in the forward direction. A large number of trajectories are launched at each interface, and successful trials are stored and used as checkpoints in the next interface. From successful and failed trials, conditional probabilities of interface crossing may be calculated and hence the transition rate from

TABLE I. Summary of methods available in SSAGES and other open-source software.

Method	Software
Replica exchange	PLUMED, Colvars, GROMACS, LAMMPS
Umbrella sampling	SSAGES, PLUMED, Colvars
Steered MD	SSAGES, PLUMED, Colvars
Metadynamics	SSAGES, PLUMED, Colvars
Well-tempered metadynamics	SSAGES, PLUMED, Colvars
Basis function sampling	SSAGES
Adaptive biasing force	SSAGES, PLUMED, Colvars
Artificial neural network sampling	SSAGES
Nudged elastic band	SSAGES
Finite temperature string	SSAGES
Swarm of trajectories	SSAGES
Forward flux sampling	SSAGES, PyRETIS
Transition path sampling	Open path sampling

state A to state B. Different flavors of the forward flux method use their unique protocol to select checkpoints to initiate trials at a given interface, compute final probabilities, create transitions paths, and analyze additional statistics.[76–78] SSAGES currently implements direct forward flux sampling (DFFS).

Table I summarizes the available methods in SSAGES and, if applicable, other software packages.

OVERVIEW OF COLLECTIVE VARIABLES

Collective variables (CVs) are functions of the atomic coordinates in a molecular simulation. Table II shows the CVs included in SSAGES along with a short description. We have implemented several CVs highlighted by Fiorin and co-workers[46] and added others relevant to multiple research endeavors. Although SSAGES currently does not include as many collective variables as other sampling packages (e.g., PLUMED and Colvars), it still includes the most commonly used CVs[46] and provides the capability to utilize these CVs with the range of methods in Table I. Importantly, as described above, the implementation of new CVs is straightforward, and the project invites user contributions that will help expand the list of options.

REPRESENTATIVE EXAMPLES: METHODS

To highlight the current capabilities of SSAGES, several representative examples are provided in what follows. A feature of SSAGES is the ability to easily change between methods to study the same system. We compare the results obtained using ABF, BFS, metadynamics (MTD), and ANN sampling for the dissociation of NaCl in explicit water. NaCl was modeled using parameters reported in the literature[84,85] and extended simple point charge (SPC/E)[86] water model was used. An NaCl pair along with 1498 water molecules was

TABLE II. Collective variables available in SSAGES.

Collective variable	Description
Angle	Angle between three particles.
Box volume	Volume of simulation box.
Gyration tensor	Components of gyration tensor. The user can specify what component of the tensor to calculate: principal moment, radius of gyration, asphericity, acylindricity, or shape anisotropy.
Pairwise	Calculate pairwise distances between two groups of particles. The distances are then passed to a Gaussian or switching function. The resulting quantities can be used to distinguish between clusters[79] or calculate coordination number.[80]
Particle coordinate	Cartesian coordinates of a group of particles (x, y, or z).
Particle position	Distance of a group of particles from a specified Cartesian coordinate (x, y, or z).
Particle separation	Distance between centers of mass of two groups of particles.
Rouse mode	Rouse mode of a polymer chain.
Alpha helix	Calculate alpha-helix character over a range of residues as the number of six-residue segments which match an ideal alpha-helical configuration.[81]
Parallel beta-sheet	Calculate parallel beta-sheet character over a range of residues as the number of six-residue segments which match an ideal parallel beta-sheet configuration.[81]
Anti-parallel beta-sheet	Calculate anti-parallel beta-sheet character over a range of residues as the number of six-residue segments which match an ideal anti-parallel beta-sheet configuration.[81]
Torsional	Proper dihedral between four atoms.[82,83]

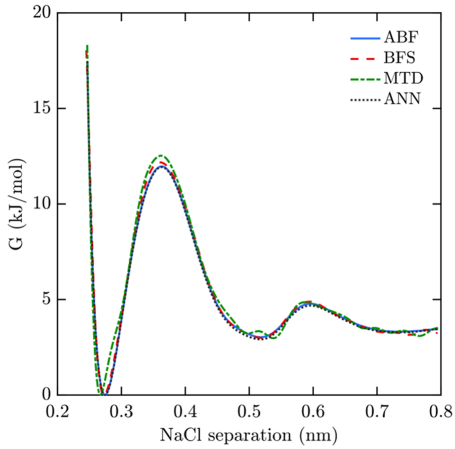

FIG. 4. Free energy profile for NaCl dissociation in water using ABF, BFS, MTD, and ANN sampling. All three methods produce very comparable results, which serves as a means to validate the methods within SSAGES.

simulated at a constant temperature and pressure of 298.15 K and 1.0 bar using the stochastic velocity rescale thermostat and Parrinello-Rahman barostat, respectively. Electrostatic interactions were calculated using particle-mesh Ewald method with a 10^{-4} accuracy. The simulations were run for 30 ns with a 2 fs time step using GROMACS 2016.3. Hydrogen bonds were held fixed using the LINCS algorithm.[87] Method-specific parameters were chosen optimally to achieve rapid convergence.

As can be seen from Fig. 4, all three methods yield very similar free energy estimates. Metadynamics, in particular, displays oscillations not present in the other methods. This is a known shortcoming of non-tempered metadynamics and need not be discussed here. The summed hills were averaged once a level bias began accruing with a stride of 100. Longer simulation times are certain to yield a smoother curve. However, the main point that we wish to emphasize is that having access to software that offers distinct methods for sampling and calculation of free energies allows one to compare the performance of different algorithms for a specific application and identify an optimal choice.

Similarly, we calculate the free energy surface (FES) for the conformational energies of alanine dipeptide (ADP) in vacuum and compare the results of ABF, BFS, metadynamics, and ANN sampling (Fig. 5). As with NaCl, we used GROMACS for the simulations. ADP was modeled using the Amber99sb forcefield[88] and hydrogen bonds were held fixed using the LINCS algorithm.[87] Simulations were run in the NVT ensemble at 298.15 K using the stochastic velocity rescale thermostat. Electrostatic interactions were calculated using particle-mesh Ewald method with a 10^{-4} accuracy. The simulations were run for 100 ns with a 2 fs time step using GROMACS 2016.3. The free energy surfaces were calculated using two torsional angles along the backbone as collective variables. We find the three configurations with the lowest free energy at approximately (−2.5, 2.7), (−1.4, 1.1), and (1, −1.1). These structures have also been found and reported previously in the literature.[47,67]

Next, we present a comparison of the nudged elastic band and the string methods implemented in SSAGES. As previously discussed, three methods are available: the nudged elastic band method, the finite temperature string method, and the string method with swarms of trajectories. In Fig. 6, an initial chain of 22 images is overlaid on a free energy surface for the alanine dipeptide configurations, obtained via ABF. The three converged pathways obtained from the three chain-of-states methods using this initial chain are also plotted over the surface in Fig. 6. All simulations, including ABF, were performed using LAMMPS with the same force field without holding hydrogen bonds rigid.

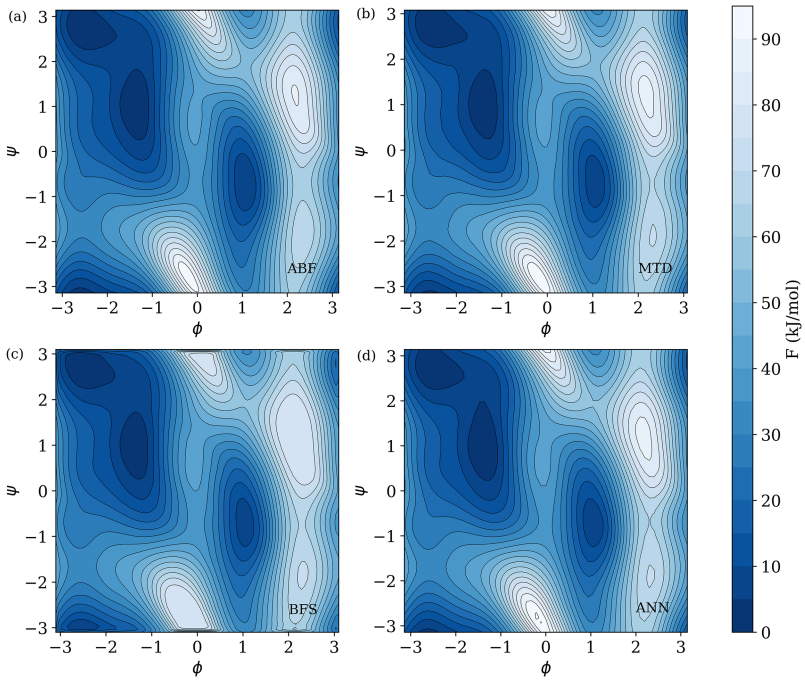

FIG. 5. Free energy surface of ADP configurations in vacuum calculated with ABF (a), BFS (b), Metadynamics (c), and ANN sampling (d). All methods find the same minima and identify the same barriers, though the explicit height and shape of high free energy barriers varies.

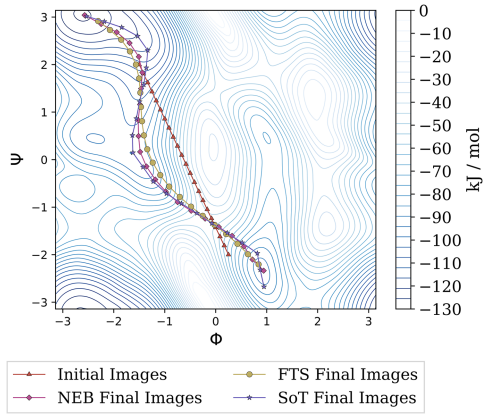

FIG. 6. Free energy landscape with chain-of-states results for ADP calculated using nudged elastic band (diamonds), finite temperature string (circles), and swarm of trajectories (stars). The contour map was obtained using ABF. We observe excellent agreement between nudged elastic band and the string methods, as well as with ABF.

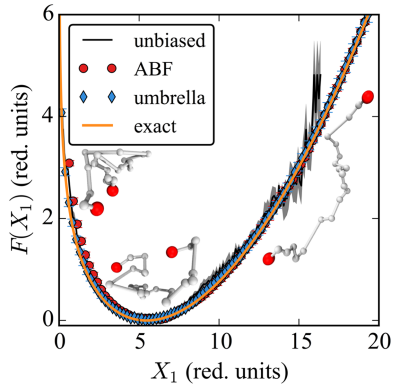

FIG. 7. Free energy profiles along the first Rouse mode amplitude X_1, using reduced units, for an ideal Gaussian chain obtained from unbiased, brute-force MD simulation (black line with gray error region), the ABF method in SSAGES (red circles), the umbrella sampling method in SSAGES (blue diamonds), and Eq. (2) (orange line). Insets are three representative configurations of the Gaussian chain for $X_1 \approx 1.0$, 5.0, and 15.0 from left to right. Error bars and/or error regions represent the standard error of the mean.

The estimated location of the transition barrier is in excellent agreement across all three methods, as well as the location of the two minima. However, the three paths differ slightly from one another outside the vicinity of the transition barrier and the two minima. This is likely a manifestation of several effects, including the fact that each of these three methods theoretically converges to a different type of pathway: NEB to the minimum free energy pathway (MFEP), FTS to the principal curve, and SoT to the most probable transition pathway (MPTP). Additionally, as discussed above, the metric tensor is assumed here to be identity for NEB and FTS for computational efficiency, which is not necessarily always the case.

REPRESENTATIVE EXAMPLES: COLLECTIVE VARIABLE

We next illustrate the application of SSAGES with a new CV. We consider the free energy profile for the Rouse modes of a polymer chain, which can describe both the internal structure and dynamics of various polymer systems.[89,90] For a polymer comprised of N segments, the Rouse mode coordinates are computed as follows:

$$\boldsymbol{X}_p = \sqrt{\frac{c_p}{2}} \sum_{i=0}^{N-1} \boldsymbol{R}_i \cos\left[\frac{p\pi}{N}\left(i+\frac{1}{2}\right)\right], \tag{1}$$

where $p = 0, \ldots, N-1$ indicates the mode index, $\boldsymbol{R}_i$ is the position of the ith segment in the polymer chain, and c_p is a constant equal to 1 for $p = 0$ and equal to 2 for all other modes. In Eq. (1), the zeroth mode roughly maps to the polymer's center of mass, and the remaining modes can be thought as collective variables corresponding to sub-chains of $(N-1)/p$ segments. For example, the relaxation at long times of the end-to-end vector $\boldsymbol{R} = \boldsymbol{R}_{N-1} - \boldsymbol{R}_0$ corresponds to the relaxation of $\boldsymbol{X}_1$.[91]

Here, we compute the free energy profile along $X_1 = \sqrt{\boldsymbol{X}_1 \cdot \boldsymbol{X}_1}$ of an ideal, Gaussian chain with SSAGES using both umbrella sampling and ABF, with unbiased, brute-force molecular dynamics and with the exact result given by

$$F\left(X_p\right) = -k_\mathrm{B}T\left[2\ln X_p - \frac{6\sin^2\frac{p\pi}{2N}}{\pi b^2}X_p^2 + \ln\left(24\sqrt{\frac{6}{\pi}}\frac{\sin^3\frac{p\pi}{2N}}{b^3}\right)\right], \tag{2}$$

where b^2 is the mean-squared bond length between adjacent segments of the polymer chain. Figure 7 compares the results of the different methods for $F(X_1)$ using reduced units with $b = 1$ and $k_\mathrm{B}T = 2/3$. All MD simulations employed the LAMMPS simulation engine. The figure shows that both umbrella sampling and ABF as implemented in SSAGES capably reconstruct the correct free energy profile. For this toy example, brute-force, unbiased MD also generates the correct free energy profile via Boltzmann inversion, albeit over a more limited range (there is no sampling for $X_1 > 16.50$) and with additional statistical error. Although the results for this system are trivially obtained, this first application, accelerated by the SSAGES framework, demonstrates an intriguing strategy to use specific Rouse modes as biasing for more novel future applications.

The umbrella sampling results were obtained using 21 simulation windows with harmonic biasing potentials given by $\frac{1}{2}k\left(X_1 - X_1^{(l)}\right)^2$, with $k = 1.0$ and $X_1^{(l)} = l$, for $l = 0.0, \ldots, 20.0$. Each simulation ran for 2.1×10^6 time steps with X_1 sampled every 50 time steps after the first 10^5 time steps used as equilibration; the free energy profile was reconstructed using WHAM. The ABF results were obtained using four independent simulations, and each simulation ran for 4×10^6 time steps. The unbiased, brute-force MD results were obtained from a single simulation of 4.1×10^6 time steps.

OUTLOOK

We present an open-source, free, and modular software for the general use of enhanced sampling techniques: SSAGES (Software Suite for Advanced General Ensemble Simulations). The software currently interfaces with LAMMPS, GROMACS, OpenMD, QBox, and DASH and includes the

following methods: umbrella sampling, metadynamics, forward flux sampling, nudged elastic band, finite temperature string, swarms of trajectories, adaptive biasing force, artificial neural network sampling, and basis function sampling. We demonstrate SSAGES' versatility by comparing various systems across different methods and collective variables and showing their agreement. Further, SSAGES provides a flexible framework for the modification and implementation of new methods and CVs. We believe that SSAGES will be used and developed by the scientific community, promoting the development and exchange of new methods and CVs. The code is open-source and can be downloaded from GitHub at https://github.com/MICCoM/SSAGES-public.

ACKNOWLEDGMENTS

This work was supported by the Department of Energy (DOE), Basic Energy Sciences, Materials Research Division, through the Midwest Integrated Center for Computational Materials (MICCoM). We would also like to acknowledge computational resources from Argonne National Laboratory (LCRC), the University of Notre Dame (CRC), and the University of Chicago (RCC). H.S. acknowledges support from the National Science Foundation (NSF) Graduate Research Fellowship Program (GRFP). J.H. acknowledges funding by the High-Performance Computing II program of Baden-Württember Stiftung.

[1] Nobelprize.org, "The Nobel Prize in Chemistry 2013," http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2013/, accessed 5 September 2017.
[2] N. Rathore, T. A. Knotts, and J. J. de Pablo, Biophys. J. **85**, 3963–3968 (2003).
[3] Y. Sugita and Y. Okamoto, Chem. Phys. Lett. **314**, 141–151 (1999).
[4] P. R. Batista, G. Pandey, P. G. Pascutti, P. M. Bisch, D. Perahia, and C. H. Robert, J. Chem. Theory Comput. **7**, 2348–2352 (2011).
[5] C. Liao and J. Zhou, J. Phys. Chem. B **118**, 5843–5852 (2014).
[6] J.-M. Depaepe, J.-P. Ryckaert, E. Paci, and G. Ciccotti, Mol. Phys. **79**, 515–522 (1993).
[7] R. C. Bernardi, M. C. R. Melo, and K. Schulten, Biochim. Biophys. Acta, Gen. Subj. **1850**, 872–877 (2015).
[8] V. Spiwok, Z. Sucur, and P. Hosek, Biotechnol. Adv. **33**, 1130–1140 (2015).
[9] M. H. Cheng and R. D. Coalson, Biophys. J. **102**, 1363–1371 (2012).
[10] M. A. Wilson, C. Wei, P. Bjelkmar, B. A. Wallace, and A. Pohorille, Biophys. J. **100**, 2394–2402 (2011).
[11] R. Sa, W. Zhu, J. Shen, Z. Gong, J. Cheng, K. Chen, and H. Jiang, J. Phys. Chem. B **110**, 5094–5098 (2006).
[12] E. Paci and M. Marchi, Mol. Simul. **14**, 1–10 (1994).
[13] S. Chunsrivirot and B. L. Trout, Langmuir **27**, 6910–6919 (2011).
[14] N. Bhatnagar, G. Kamath, I. Chelst, and J. J. Potoff, J. Chem. Phys. **137**, 014502 (2012).
[15] F. Calvo and C. Mottet, Phys. Rev. B **84**, 035409 (2011).
[16] S. Sharma and P. G. Debenedetti, J. Phys. Chem. B **116**, 13282–13289 (2012).
[17] R. Martoňák, A. Laio, and M. Parrinello, Phys. Rev. Lett. **90**, 075503 (2003).
[18] R. Martoňák, D. Donadio, A. R. Oganov, and M. Parrinello, Nat. Mater. **5**, 623–626 (2006).
[19] F. Trudu, D. Donadio, and M. Parrinello, Phys. Rev. Lett. **97**, 105701 (2006).
[20] A. G. Stack, P. Raiteri, and J. D. Gale, J. Am. Chem. Soc. **134**, 11–14 (2011).
[21] Q. L. Yan and J. J. de Pablo, J. Chem. Phys. **113**, 1276–1282 (2000).
[22] D. J. Earl and M. W. Deem, Phys. Chem. Chem. Phys. **7**, 3910–3916 (2005).
[23] T. S. Jain and J. J. de Pablo, J. Chem. Phys. **116**, 7238–7243 (2002).
[24] J. I. Siepmann and D. Frenkel, Mol. Phys. **75**, 59–70 (1992).
[25] J. J. de Pablo, M. Laso, and U. W. Suter, J. Chem. Phys. **96**, 2395–2403 (1992).
[26] Y. Sugita and Y. Okamoto, Chem. Phys. Lett. **329**, 261–270 (2000).
[27] Q. L. Yan and J. J. de Pablo, J. Chem. Phys. **111**, 9509–9516 (1999).
[28] N. Rathore, M. Chopra, and J. J. d. Pablo, J. Chem. Phys. **122**, 024111 (2005).
[29] G. M. Torrie and J. P. Valleau, J. Comput. Phys. **23**, 187–199 (1977).
[30] S. Singh, M. Chopra, and J. J. de Pablo, Annu. Rev. Chem. Biomol. Eng. **3**, 369–394 (2012).
[31] W. E, W. Ren, and E. Vanden-Eijnden, Phys. Rev. B **66**, 052301 (2002).
[32] W. E, W. Ren, and E. Vanden-Eijnden, J. Phys. Chem. B **109**, 6688–6693 (2005).
[33] V. Ovchinnikov, M. Karplus, and E. Vanden-Eijnden, J. Chem. Phys. **134**, 085103 (2011).
[34] E. Darve and A. Pohorille, J. Chem. Phys. **115**, 9169–9183 (2001).
[35] A. Barducci, M. Bonomi, and M. Parrinello, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **1**, 826–843 (2011).
[36] A. Laio and M. Parrinello, Proc. Natl. Acad. Sci. U. S. A. **99**, 12562–12566 (2002).
[37] L. Sutto, S. Marsili, and F. L. Gervasio, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **2**, 771–779 (2012).
[38] J. K. Whitmer, C.-c. Chiu, A. A. Joshi, and J. J. de Pablo, Phys. Rev. Lett. **113**, 190602 (2014).
[39] J. K. Whitmer, A. M. Fluitt, L. Antony, J. Qin, M. McGovern, and J. J. de Pablo, J. Chem. Phys. **143**, 044101 (2015).
[40] C. Dellago and P. G. Bolhuis, Adv. Polym. Sci. **221**, 167–233 (2009).
[41] C. Abrams and G. Bussi, Entropy **16**, 163 (2014).
[42] P. Kollman, Chem. Rev. **93**, 2395 (1993).
[43] D. Trzesniak, A. P. E. Kunz, and W. F. van Gunsteren, ChemPhysChem **8**, 162–169 (2007).
[44] E. Vanden-Eijnden, J. Comput. Chem. **30**, 1737–1747 (2009).
[45] C. D. Christ, A. E. Mark, and W. F. Van Gunsteren, J. Comput. Chem. **31**, 1569–1582 (2010).
[46] G. Fiorin, M. L. Klein, and J. Hénin, Mol. Phys. **111**, 3345–3362 (2013).
[47] M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R. A. Broglia, and M. Parrinello, Comput. Phys. Commun. **180**, 1961–1972 (2009).
[48] G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, Comput. Phys. Commun. **185**, 604–613 (2014).
[49] A. Lervik, E. Riccardi, and T. S. van Erp, J. Comput. Chem. **38**(28), 2439–2451 (2017).
[50] J.-H. Prinz, D. W. H. Swenson, J. Chodera, and P. G. Bolhuis, Open Path Sampling, http://openpathsampling.org/latest/, 2017.
[51] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten, J. Comput. Chem. **26**, 1781–1802 (2005).
[52] S. Plimpton, J. Comput. Phys. **117**, 1–19 (1995).
[53] H. J. C. Berendsen, D. Vanderspoel, and R. Vandrunen, Comput. Phys. Commun. **91**, 43–56 (1995).
[54] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, J. Chem. Theory Comput. **4**, 435–447 (2008).
[55] J. Gezelter, S. Kuang, J. Marr, K. Stocker, C. Li, C. Vardeman, T. Lin, C. Fennell, X. Sun, and K. Daily, OPENMD, an open source engine for molecular dynamics, available at http://openmd.net.
[56] F. Gygi, IBM J. Res. Dev. **52**, 137–144 (2008).
[57] D. R. Reid and J. J. de Pablo (2017). Zenodo. https://doi.org/10.5281/zenodo.886545
[58] G. Guennebaud and B. Jacob, Eigen v3, 2010, http://eigen.tuxfamily.org.
[59] J. Kästner, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **1**, 932–942 (2011).
[60] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman, J. Comput. Chem. **13**, 1011–1021 (1992).
[61] M. Souaille and B. Roux, Comput. Phys. Commun. **135**, 40–57 (2001).
[62] J. Kästner and W. Thiel, J. Chem. Phys. **123**, 144104 (2005).
[63] H. Grubmüller, B. Heymann, and P. Tavan, Science **271**, 997–999 (1996).
[64] S. Izrailev, S. Stepaniants, M. Balsera, Y. Oono, and K. Schulten, Biophys. J. **72**, 1568–1581 (1997).
[65] E. Evans and K. Ritchie, Biophys. J. **72**, 1541–1555 (1997).
[66] M. Balsera, S. Stepaniants, S. Izrailev, Y. Oono, and K. Schulten, Biophys. J. **73**, 1281–1287 (1997).
[67] L. Alessandro and L. G. Francesco, Rep. Prog. Phys. **71**, 126601 (2008).
[68] H. Sidky and J. K. Whitmer, e-print arXiv:1712.02840 (2017).
[69] E. Darve, D. Rodríguez-Gómez, and A. Pohorille, J. Chem. Phys. **128**, 144120 (2008).
[70] J. Comer, J. C. Gumbart, J. Hénin, T. Lelièvre, A. Pohorille, and C. Chipot, J. Phys. Chem. B **119**, 1129–1151 (2015).
[71] G. Henkelman, B. P. Uberuaga, and H. Jónsson, J. Chem. Phys. **113**, 9901–9904 (2000).

[72]H. Jónsson, G. Mills, and K. W. Jacobsen, in *Classical and Quantum Dynamics in Condensed Phase Simulations*, edited by B. J. Berne, G. Ciccotti, and D. F. Coker (World Scientific, Singapore, 1998), p. 385.
[73]G. Henkelman and H. Jónsson, J. Chem. Phys. **113**, 9978–9985 (2000).
[74]E. Vanden-Eijnden and M. Venturoli, J. Chem. Phys. **130**, 194103 (2009).
[75]A. C. Pan, D. Sezer, and B. Roux, J. Phys. Chem. B **112**, 3432–3440 (2008).
[76]R. J. Allen, C. Valeriani, and P. R. ten Wolde, J. Phys.: Condens. Matter **21**, 463102 (2009).
[77]F. A. Escobedo, E. E. Borrero, and J. C. Araque, J. Phys.: Condens. Matter **21**, 333101 (2009).
[78]R. J. Allen, D. Frenkel, and P. R. ten Wolde, J. Chem. Phys. **124**, 194111 (2006).
[79]G. A. Tribello, J. Cuny, H. Eshet, and M. Parrinello, J. Chem. Phys. **135**, 114109 (2011).
[80]M. Iannuzzi, A. Laio, and M. Parrinello, Phys. Rev. Lett. **90**, 238302 (2003).
[81]F. Pietrucci and A. Laio, J. Chem. Theory Comput. **5**, 2197–2201 (2009).
[82]R. C. van Schaik, H. J. C. Berendsen, A. E. Torda, and W. F. van Gunsteren, J. Mol. Biol. **234**, 751–762 (1993).
[83]A. Blondel and M. Karplus, J. Comput. Chem. **17**, 1132–1141 (1996).
[84]D. Beglov and B. Roux, J. Chem. Phys. **100**, 9050–9063 (1994).
[85]B. Roux, Biophys. J. **71**, 3177–3185 (1996).
[86]H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma, J. Phys. Chem. **91**, 6269–6271 (1987).
[87]B. Hess, H. Bekker, H. J. C. Berendsen, and J. Fraaije, J. Comput. Chem. **18**, 1463–1472 (1997).
[88]V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling, Proteins: Struct., Funct., Bioinf. **65**, 712–725 (2006).
[89]M. Doi and S. F. Edwards, *The Theory of Polymer Dynamics* (Oxford University Press, 1988).
[90]P. E. Rouse, Jr., J. Chem. Phys. **21**, 1272–1280 (1953).
[91]A. Kopf, B. Dünweg, and W. Paul, J. Chem. Phys. **107**, 6945–6955 (1997).