Expanded functionality and portability for the Colvars library

Giacomo Fiorin, Fabrizio Marinelli, Lucy Forrest, Haochuan Chen, Christophe Chipot, Axel Kohlmeyer, Hubert Santuz, Jérôme Hénin

Expanded functionality and portability for the Colvars library

Giacomo Fiorin,[*,†,‡] Fabrizio Marinelli,[¶,‡] Lucy R. Forrest,[†] Haochuan Chen,[§] Christophe Chipot,[‖,§,⊥,#] Axel Kohlmeyer,[@] Hubert Santuz,[△] and Jérôme Hénin[*,△]

†*National Institute of Neurological Disorders and Stroke, Bethesda, MD, 20814, USA*

‡*National Heart, Lung and Blood Institute, Bethesda, MD, 20892, USA*

¶*Department of Biophysics and Data Science Institute, Medical College of Wisconsin, Milwaukee, WI 53226-3548, USA*

§*Theoretical and Computational Biophysics Group, Beckman Institute, and Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois 61820, United States*

‖*Laboratoire International Associé CNRS et University of Illinois at Urbana-Champaign, UMR 7019, Université de Lorraine, 54506 Vandœuvre-lès-Nancy, France*

⊥*Department of Biochemistry and Molecular Biology, The University of Chicago, 929 E. 57th Street W225, Chicago, Illinois 60637, United States*

#*Department of Chemistry, The University of Hawai'i at Manoa, 2545 McCarthy Mall, Honolulu, Hawaii 96822, United States*

@*Institute for Computational Molecular Science, Temple University, Philadelphia, PA, 19122, USA*

△*Université Paris Cité, CNRS, Laboratoire de Biochimie Théorique UPR 9080, 75005, Paris, France*

E-mail: giacomo.fiorin@nih.gov; jerome.henin@cnrs.fr

Abstract

Colvars is an open-source C++ library that provides a modular toolkit for collective-variable-based molecular simulations. It allows practitioners to easily create and implement descriptors that best fit a process of interest, and to apply a wide range of biasing algorithms in collective variable space. This paper reviews several features and improvements to Colvars that were added since its original introduction. Special attention is given to contributions that significantly expanded the capabilities of this software or its distribution with major MD simulation packages. Collective variables can now be optimized either manually or by machine-learning methods, and the space of descriptors can be explored interactively using the graphical interface included in VMD. Beyond the spatial coordinates of individual molecules, Colvars can now apply biasing forces to mesoscale structures and alchemical degrees of freedom, and perform simulations guided by experimental data within ensemble averages or probability distributions. It also features advanced computational schemes to boost the accuracy, robustness, and general applicability of simulation methods, including extended-system and multiple-walker Adaptive Biasing Force, boundary conditions for metadynamics, replica exchange with biasing potentials, and Adiabatic Bias Molecular Dynamics. The library is made available directly within the main distributions of the academic software GROMACS, LAMMPS, NAMD, Tinker-HP, and VMD. The robustness of the software and the reliability of the results are ensured through the use of continuous integration, with a test suite within the source repository.

Introduction

In the field of molecular simulations, projecting high-dimensional configurations into a low-dimension space of collective variables (CVs) is a common way to perform enhanced sampling,[1] and a nearly universal tool for analyzing simulation data to extract physical, chemical, and biological insight.

Computational techniques, including CV-based ones, are generally useful insofar as a

reliable and efficient software implementation exists for them. Accordingly, the most commonly used simulation methods in molecular simulation (interatomic potentials, thermostats, barostats, multiple time-stepping) have gained wide distribution via software applications that could be installed directly by their intended users. However, methods for CV-based enhanced sampling come in a broader variety compared to those for conventional MD simulations, a fact that hinders their bottom-up re-implementation in each software package. To address this issue, we have previously introduced the collective variables module (Colvars), a software library designed for distribution with multiple simulation and analysis software packages.[2]

The close integration between Colvars and each package has greatly simplified many computational protocols, to the extent that Colvars has been perceived in a few cases to be an exclusive feature of NAMD,[3] the package to which Colvars was first added. It is more accurate instead to consider Colvars as distinct from the CV implementations specific to each MD package, such as the pull code of GROMACS[4] or the CustomCVForce feature of OpenMM,[5] as well as from independently distributed plugins such as PLUMED.[6]

Further customization is available through a scripting interface, or a graphical user interface (GUI)[7] within VMD.[8] Additionally, the Colvars library is under continuous development, with many of its features being introduced over multiple major releases of the packages with which it is distributed. Therefore, it is useful to review the main extensions and new features brought to the library since the original publication.[2]

Following a brief summary of the underlying theoretical concepts, this manuscript reviews new or improved collective variables to reconstruct the energy landscape of multiple macromolecules in contact with each other as well as their own conformational changes. Also reviewed are the improvements to established sampling schemes such as adaptive biasing force (ABF) and metadynamics, and the introduction of new schemes to bias the simulated ensemble towards experimental data. Lastly, the enhanced interfaces to the NAMD and LAMMPS[9] packages, as well as the interfaces to VMD, Tinker-HP and GROMACS are also

discussed.

General architecture of Colvars

A collective variable, often abbreviated as 'colvar' or CV, can be any function $\xi(\mathbf{X}, \mathbf{u})$ of the Cartesian coordinates of multiple atoms $\mathbf{X}$ and the periodic cell parameters $\mathbf{u}$, such that the values of ξ map physically relevant regions of configurational space. To enhance the sampling of rare states in MD simulations, external forces are added in colvar space and propagated to the equations of motion of the individual atoms in Cartesian space. Owing to generality of this scheme, any mathematical function can define of a CV, as long as it i continuous and differentiable in all its arguments.

The most common use of Colvars is a continuous simulation of a single copy of the model system, under the effect of external forces whose effects are measured and included in the estimate of a-free energy profile for the system at equilibrium. It is, however, also straightforward to use Colvars with methods that introduce artificial discontinuities in the atomic trajectory[10] or rely on selecting simulation snapshots without applying external biasing forces,[11–14] or other multi-copy simulations.[15]

As with conventional MD simulations, computation using methods implemented by Colvars requires the user to define two entities:

- At least one collective variable, by selecting atoms and a function of those atoms' coordinates: this is a generalization of interatomic variables such as bonds, angles, and dihedrals.

- A scheme to govern the dynamics of the colvar just defined: this is a generalization of the interatomic potentials; however, because the typical use is to achieve biased sampling, this scheme is called a '*bias*'.

Multiple variables and biases can be simultaneously defined. The Colvars library offers many choices of definition for each, ranging from massively parallel, compiled code to scripted

functions defined by the user at run time; both are described below. This fully modular architecture gives practitioners access to a broad space of possible collective variables and types of biased dynamics.

Notable or new coordinates

Euler and polar angles

To facilitate the description of the spatial arrangement of a ligand and a protein as one-dimensional variables in binding free-energy calculations,[16] the Colvars library has introduced *(i)* the polar (θ) and azimuthal (ϕ) angles that describe the position of an atom group in spherical coordinates,[17] and *(ii)* the "roll" (Φ), "pitch" (Θ) and "yaw" (Ψ) angles that depict the rotation of an atom group with respect to a reference frame.[18] These angles are calculated as

$$\begin{cases} \theta & = \arccos\left(z\right) \\ \phi & = \text{arctan2}\left(y, x\right) \\ \Phi & = \text{arctan2}\ \left(2\left(q_0 q_1 - q_3 q_2\right), 1 - 2\left(q_1^2 + q_2^2\right)\right) \\ \Theta & = \arcsin\ \left(2(q_0 q_2 - q_3 q_1)\right) \\ \Psi & = \text{arctan2}\ \left(2(q_0 q_3 + q_1 q_2), 1 - 2\left(q_2^2 + q_3^2\right)\right) \end{cases}$$

where (x, y, z) is the unit vector from the origin to the center-of-mass of the molecule, and (q_0, q_1, q_2, q_3) is the quaternion describing the rotation that minimizes the root-mean-square-deviation (RMSD) of the molecule with respect to the reference.[2,19] A schematic representation of these angles is shown in Figure 1. The radial distance r, the angles θ and ϕ form a complete set of polar coordinates, which is defined in Colvars using the `distance`, `polarTheta`, and `polarPhi` keywords. Conversely, the angles Φ, Θ and Ψ are defined using the `eulerPhi`, `eulerPsi` and `eulerTheta` keywords, which define internally the computation of a quaternion $\mathbf{q}$ from which the relevant projection is carried out. The set of functions

described above is extensively used by the Binding Free Energy Estimator.[20,21] Like any other collective variable, these angles can be computed in a moving frame of reference tied to a specified group of atoms (see section below).

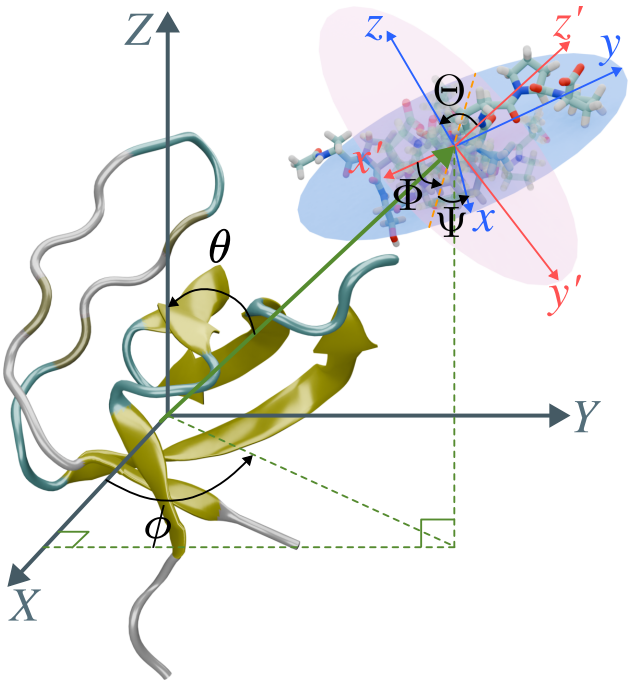

Figure 1: Schematic representation of the polar angles (ϕ, θ) and the Euler angles (Φ, Θ, Ψ) in protein-ligand binding. The polar angles (ϕ, θ) describe how the ligand revolves around the protein. The Euler angles (Φ, Θ, Ψ) depict the rotation of the ligand around its origin.

Path collective variables

When considering a pathway that connects two metastable states, A and B, of the free-energy landscape underlying a geometric transformation—e.g., the conformational transition of a protein—it becomes advantageous to associate a progress variable[22,23] to this pathway, which can be supplied, for instance, by path-optimization strategies like the string method and its variants,[24–26] or, more recently, by machine-learning strategies aimed at discovering the committor, i.e., the probability that, starting from a given configuration, the target state, B, will be reached before returning to the reference state, A.[27–29] Such an approach facilitates the calculation of the free-energy change between the reference and the target states. The

concept of a progress variable also offers a robust framework for reducing dimensionality,[30–32] while providing a concise description of potentially complex geometric transformations by means of a one-dimensional free-energy profile, or "potential of mean force". The latter ideally captures the dynamics of the molecular processes at play in the transition between the two end-states.

Computing a progress variable necessitates the projection of the collective variable, or alternatively, the Cartesian-coordinate space onto the path representing the average transition, resulting in differentiable expressions. These expressions characterize a continuous pathway along which a free-energy change can be precisely determined. An example of such expressions is furnished by the so-called path-collective variables (PCVs), which are introduced here in a variant of their original arithmetic formulation,[33]

$$\begin{cases} s(\mathbf{z}) &= \dfrac{1}{N-1}\dfrac{\displaystyle\sum_{i=0}^{N-1} i \ \exp\left(-\lambda \sum_{j=1}^{M} c_j \left(z_j - z_j^{(i)}\right)^2\right)}{\displaystyle\sum_{i=0}^{N-1} \exp\left(-\lambda \sum_{j=1}^{M} c_j \left(z_j - z_j^{(i)}\right)^2\right)} \\ \zeta(\mathbf{z}) &= -\dfrac{1}{\lambda} \ln\left(\displaystyle\sum_{i=0}^{N-1} \exp\left(-\lambda \sum_{j=1}^{M} c_j \left(z_j - z_j^{(i)}\right)^2\right)\right) \end{cases} \tag{1}$$

where c_j is the weight of the j–th CV, $z_j^{(i)}$, the value of this CV for the i–th reference frame, or node of the pathway, and z_j, the value of j–th CV for the current frame. M and N are the number of CVs and the number of reference frames, respectively. λ serves as a smoothing parameter that relates to the inverse of the mean-squared displacement between consecutive images. Variable $s(\mathbf{z})$ acts as a progress parameter along the pathway, ranging from 0 (state A) to 1 (state B), while the ancillary variable $\zeta(\mathbf{z})$ can be interpreted as the radius of a tube enveloping the pathway, confining the sampling within its vicinity. This conceptual framework aids in determining the underlying free energy profile, most notably from a coarse approximation of the pathway, by mapping the free energy in the two dimensions, $s(\mathbf{z})$ and $\zeta(\mathbf{z})$, denoted `aspathCV` and `azpathCV` in the Colvars library. The one-dimensional free-

energy profile is then derived from the marginal distribution of $s(\mathbf{z})$.

An alternative to the previously stated definition of the PCV is provided by the original arithmetic expression of Branduardi et al,[33] resting on mean-squared displacements between the current position in Cartesian-coordinate space and that of the images of the string. These PCV are called `aspath` and `azpath` in the Colvars library, and have proven useful when it is not possible to infer, primarily from human intuition, the subspace of CV needed to optimize the pathway.

While the free energy should ideally remain immune to changes in λ for straightforward linear abscissas, the complexity of the CV or Cartesian space, along with the non-linearity of the string, complicates its choice and is prone to induce instabilities in the trajectory due to the singularity of the mean-squared displacement as it approaches zero. This ailment can be alleviated by means of a normalized exponential function (Softmax). Still, even minor deviations from the optimal value of λ may result not only in sampling inefficiencies but also in discernible artifacts that compromise the physical accuracy of sampling. These limitations have led to the development of alternative geometric expressions that are more robust to parameter selection. For example, path-metadynamics variables (PMVs) generate unique values for $s(\mathbf{z})$ and $\zeta(\mathbf{z})$, unlike the original arithmetic PCV formulation, and are defined as follows,[34]

$$\begin{cases} s(\mathbf{z}) &= \dfrac{n}{N} \pm \dfrac{1}{2N} \left\{ \dfrac{\left[(\mathbf{v}_1 \cdot \mathbf{v}_3)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)\right]^{1/2} - (\mathbf{v}_1 \cdot \mathbf{v}_3)}{|\mathbf{v}_3|^2} - 1 \right\} \\ \zeta(\mathbf{z}) &= \left| \mathbf{v}_1 + \dfrac{1}{2} \left\{ \dfrac{\left[(\mathbf{v}_1 \cdot \mathbf{v}_3)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)\right]^{1/2} - (\mathbf{v}_1 \cdot \mathbf{v}_3)}{|\mathbf{v}_3|^2} - 1 \right\} \mathbf{v}_4 \right| \end{cases} \tag{2}$$

Here, $\mathbf{v}_1 = \mathbf{s}_n - \mathbf{z}$ is a vector pointing from the current position to the nearest image, $\mathbf{v}_2 = \mathbf{z} - \mathbf{s}_{n-1}$, a vector pointing fro the second nearest image to the current position, $\mathbf{v}_3 = \mathbf{s}_{n+1} - \mathbf{s}_n$, a vector pointing from the nearest to the third nearest image, and $\mathbf{v}_4 = \mathbf{s}_n - \mathbf{s}_{n-1}$,

a vector pointing from the second nearest to the nearest image. Here, n is the index of the nearest image, and N is the total number of images. If the current position is to the left of the nearest reference image, the sign in the expression of $s(\mathbf{z})$ is positive—otherwise, it is negative. The above PMV, $s(\mathbf{z})$ and $\zeta(\mathbf{z})$, are denoted `gspathCV` and `gzpathCV` in the Colvars library, and, just like the PCV, possess a related definition in Cartesian-coordinate space, referred to as `gspath` and `gzpath`.

Moving frame of reference

Many applications of Colvars are focused on studying changes in structure within one macromolecule, or the interactions between multiple macromolecules. However, in some of these cases, the CVs employed are sensitive to translations or rotations of those macromolecules, making the associated free-energy landscape unnecessarily complex. To address this issue, Colvars allows for any CVs to be defined in an invariant frame of reference, whose axes and point of origin follow the motion of a chosen group of atoms. The same methodology[19] and implementation[2] used in the computation of optimal RMSDs also underlie the definition of such a frame of reference.

A typical example of this feature is a protein/ligand complex, where the protein defines the moving frame of reference, and the ligand's movements are parameterized by specific CVs expressed in that frame. These CVs may include, for example, the translations and Euler angles used in the Binding Free-Energy Estimator (BFEE) protocol,[18,35] or the RMSD from a reference structure in the Distance from Bound Configuration (DBC) coordinate used in the SAFEP approach.[36,37] Notably, the implementation places no restrictions on the number of frames of reference that may be defined concurrently and is sufficiently modular to support arbitrary types of CVs. For example, a moving frame of reference may be combined with a symmetry-invariant RMSD coordinate for efficient restraining of symmetric ligands.[38]

A moving frame of reference is defined by the `fittingGroup` keyword, which selects the set of atoms whose frame of reference a CV may be defined on. The set of fitting atoms

is defined separately from the atoms explicitly used in defining a CV, and the coordinates of the latter set of atoms are transformed into the moving frame whenever the CV is being recomputed. In mathematical terms any function $\zeta(\mathbf{X})$ may be defined in a moving frame as:

$$\zeta^{(\text{fitted})}(\mathbf{X};\, \mathbf{X}^{(\text{fit})}) = \zeta(\mathbf{R}(\mathbf{X} - \bar{\mathbf{x}}) + \bar{\mathbf{x}}^{(\text{fit})}) \tag{3}$$

where $\bar{\mathbf{x}}$ is the center of geometry of $\mathbf{X}$, $\mathbf{R}$ is the optimal rotation matrix, $\mathbf{X}^{(\text{fit})}$ are the Cartesian coordinates of the fitting atoms, $\bar{\mathbf{x}}^{(\text{fit})}$ their center of the geometry. Because R is a function of $\mathbf{X}^{(\text{fit})}$, any biasing force applied to $\zeta^{(\text{fitted})}$ is also propagated to the fitting atoms by implicitly calculating the gradients of $\zeta^{(\text{fitted})}$ with respect to $\mathbf{X}^{(\text{fit})}$ as detailed in the Supporting Information.

While this treatment allows to consistently bias all degrees of freedom associated to $\zeta^{(\text{fitted})}$, the additional computation requires nested loops over both $\mathbf{X}$ and $\mathbf{X}^{(\text{fit})}$, which may be computationally costly. Such consideration applies to all moving frames of reference, with the exception of the special case when $\mathbf{X}$ and $\mathbf{X}^{(\text{fit})}$ are the same coordinates and ζ is the RMSD function, for which the fitting gradients are zero by definition and are therefore not computed. In all other cases, it is good practice to define a CV and a moving frame of reference such that the atoms $\mathbf{X}$ selected for the CV are significantly fewer than the fitting atoms $\mathbf{X}^{(\text{fit})}$.

DEER CVs

A primary objective of molecular dynamics simulations is to provide a molecular interpretation of complex experimental signals, particularly those highlighting large-scale conformational changes in biomolecules, like double electron-electron resonance (DEER) measurements. This technique measures long distances (up to 16 nm[43]) between spin labels on a biomolecule (Fig. 2). To compute the DEER signal of a biomolecule with two attached spin

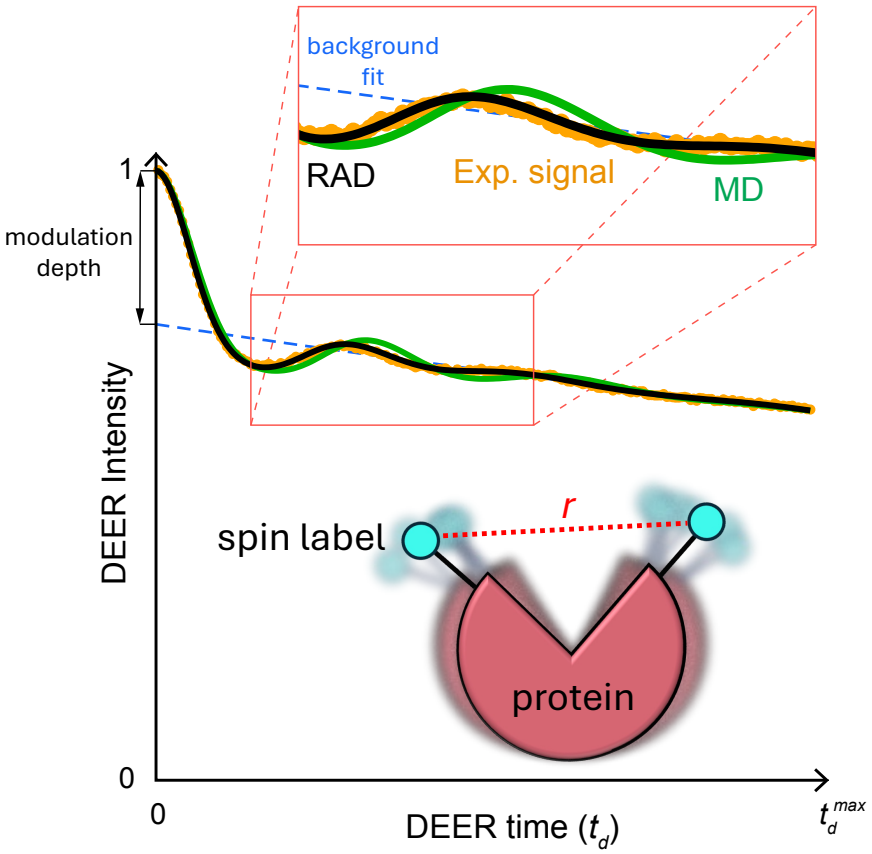

Figure 2: Example of experimental and calculated DEER time-domain signals. Below the DEER traces is a schematic of the protein with attached spin labels. The experimental signal (orange circles) is from DEER measurements on spin-labeled T4 Lysozyme at positions 62 and 109[39,40,41]. DEER traces calculated using conventional MD and the restrained average dynamics (RAD) technique[42] are shown as green and black lines, respectively, representing the time average of the `deer` CV (see Eq. 4). The blue dashed line represents the fit of the background function, $(1-\Lambda_1)\exp-(\Lambda_2|t_d|)$, at large times when oscillations are minimal.

labels, we implemented a vector-valued CV in Colvars, named `deer`. This CV assesses the contribution of an individual biomolecular configuration to the signal.

The implementation of the `deer` CV is based on the standard approximation of nearly isolated spin pairs in a dilute solution,[44,45] enabling the `deer` CV to be expressed as a function of the spin labels' distance, r (the distance between nitroxide groups in MTSSL spin labels):

$$F_{t_d}(r) = [(1-\Lambda_1) + \Lambda_1 k(t_d, r)] \exp -(\Lambda_2 |t_d|)^{D/3} \tag{4}$$

where t_d denotes the time of the DEER signal, spanning an interval with length provided as an input parameter. Each t_d value corresponds to a specific vector component of the `deer` CV. The experimental parameters Λ_1 and Λ_2 characterize the signal modulation depth and the background contribution from spin-spin interactions between different biomolecules in the solution, respectively (Fig. 2). The parameter D describes the system dimensionality (3 for a solvated biomolecule in a homogeneous sample and 2 for a membrane-embedded biomolecular system). The function $k(t_d, r)$ (DEER kernel) represents the intramolecular spin-spin contribution to the signal and is the only term in Eq. 4 that depends on the molecular configuration via the spin labels' distance, r. The analytical expression for this term, based on the same approximations and assuming ideal pulses, is given by:[46–48]

$$k(t_d, r) = \sqrt{\frac{\pi \left[C(z)^2 + S(z)^2\right]}{6\omega_d t_d}} \cos\left[\omega_d t_d - \arctan\frac{S(z)}{C(z)}\right] \tag{5}$$

where $\omega_d = g^2 \mu_B^2 \mu_0 / 4\pi\hbar r^3$ denotes the dipolar frequency. $C(z) = \int_0^z \cos\left(\frac{\pi}{2}x^2\right) dx$ and $S(z) = \int_0^z \sin\left(\frac{\pi}{2}x^2\right) dx$ are the cosine and sine Fresnel integrals ($z = \sqrt{6\omega_d t_d/\pi}$).

Once the `deer` CV is evaluated along the MD trajectory, the predicted DEER signal can be calculated as the time average of the CV: $\langle F_{t_d}(r)\rangle = [(1-\Lambda_1) + \Lambda_1 \langle k(t_d, r)\rangle] \exp -(\Lambda_2 |t_d|)^{D/3}$ (where $\langle ... \rangle$ denotes ensemble average). This can be computed, for example, using the running average functionality in Colvars or a custom script. The experimental parameters, Λ_1 and Λ_2, can be provided in input if they are known a priori (e.g. through background fitting

as shown in Fig. 2). Alternatively, we also implemented routines in Colvars to evaluate them automatically by best fitting predicted and experimental signals.[42] The current implementation also supports the use of only the DEER kernel function in Eq. 5, referred to as the `deerkernel` CV. This can provide a prediction of the background-corrected and shifted DEER signal ($\langle k\left(t_d, r\right)\rangle$), which can be inferred from the measured signal if Λ_1 and Λ_2 are known.

Overall, the `deer` and `deerKernel` CVs enable comparison of predicted and experimental DEER signals for rigorous molecular interpretation. They can also be used in combination with refinement methods like Restrained Average Dynamics (RAD)[42] to align simulation ensembles with experimental data (see the section on biasing methods below). At the time of writing, these feature are available in a Colvars development branch and are slated for integration into the standard release.

Machine-learned CVs

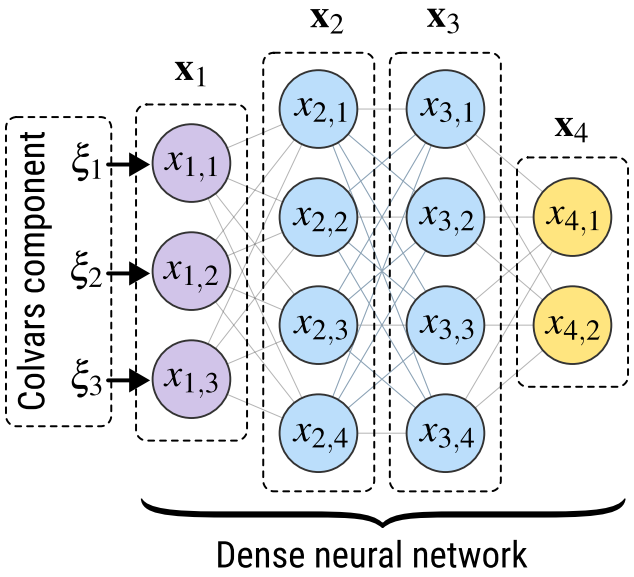

Figure 3: Schematic representation of a dense neural network. Each circle represents a node, and nodes within the input, hidden, and output layers are marked in purple, blue, and yellow, respectively.

In principle, according to the universal approximation theorem,[49] a neural network (NN)

with hidden layers can be used to approximate any continuous function. Thanks to this advantage, machine learning using NNs has become a widely used strategy for dimensionality reduction and discovering CVs.[50–60] To make use of these CVs in molecular simulations, we have implemented in Colvars the machine-learned CV (MLCV) component,[58] referred to as `neuralNetwork`. This function can use any of the available CVs available in Colvars as inputs, and forwards them into a user-defined dense neural network (NN): the output of the network's last layer provides the value for the CV. In a dense NN, the output of the j-th node at the k-th layer that has N_k nodes is computed as

$$x_{k,j} = f_k \left(\sum_{i=1}^{N_{k-1}} w_{(k,j),(k-1,i)} x_{k-1,i} + b_{k,j} \right) \tag{6}$$

where f_k is the activation function of the k-th layer, $w_{(k,j),(k-1,i)}$ is the weight of the j-th node at the k-th layer with respect to the i-th node at the $(k-1)$-th layer, $b_{k,j}$ is the bias of the j-th node at the k-th layer, and N_{k-1} is the number of nodes at the $(k-1)$-th layer. The computation can be also written in the vector form $\mathbf{x}_k = \mathbf{f}_k(\mathbf{W}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{b}_k)$, where $\mathbf{f}_k$ is an N_k-dimension activation function, $\mathbf{W}_{k,k-1}$ is an $N_k \times N_{k-1}$ weight matrix, $\mathbf{x}_{k-1}$ is the output of the previous layer, and $\mathbf{b}_k$ is an N_k-dimension bias vector. A schematic representation of the structure of a neural network can be found in Figure 3. To make MLCV agnostic of the underlying deep-learning frameworks, the weights and biases of the dense NN are provided in plain text files, and the activation functions can be defined using the Lepton library.[61] The MLCV component has been used, for instance, to implement the CVs discovered by autoencoders for describing the folding of proteins.[62]

In addition to the `neuralNetwork` feature, which is available directly within the Colvars library, more general NN coordinates can be computed through an interface to the PyTorch library. This functionality, available through the `torchANN` keyword, allows for computing colvars as functions of existing variables defined by a NN in PyTorch format. The ability to use the `torchANN` feature is conditional upon the presence of a Torch library linked to the

MD engine, or dynamically loaded at runtime.

Volumetric map-based CVs

Traditionally, biased MD simulations approaches have often been used to simulate changes in conformation of few individual molecules, such as proteins or nucleic acids. However, supra-molecular aggregates such as water clusters, surfactant micelles and lipid membranes, are more accurately described by the spatial distribution of their constituent molecules. Therefore, suitable CVs for these systems could be defined using that distribution directly, for example in the form of a density map evaluated on a dense grid over a given volume.

Collective variables based on volumetric maps were recently introduced in two variants[63]

1. as variables based on a single map, $\phi(\mathbf{X})$ (`mapTotal` keyword), to bias the number of water molecules in a given region of space; among the applications of this approach is, for instance, the dewetting of ion channel pores;[64]

2. as multiple maps combined together, $\sum_i \xi_i \phi_i(\mathbf{X})$, to define a continuous pathway that connects different morphological states of a lipid membrane.

The latter approach, referred to as *Multi-Map*, has been applied to computing the free-energy cost of membrane deformation by embedded transporter proteins,[65] obtain coarse models of protein conformational changes from density maps,[66] and to describe the mechanism of curvature generation in cholesterol-rich membranes.[67]

The current implementation is specific to NAMD and VMD, because it leverages their implementations of volumetric maps[8,68] as well as the functionality of the Colvars Dashboard VMD plugin.[7] Extension to other engines and future improvements will be described elsewhere.

Alchemical variable and lambda-dynamics

Colvars now implements an alchemical collective variable `alchLambda`, which is communicated to compatible back-ends – currently, Tinker-HP or NAMD – if their alchemical simulation mode is active.[69] When the extended-Lagrangian feature is enabled for such a variable, the alchemical simulation becomes driven by Colvars, resulting in a λ-dynamics trajectory. λ-dynamics has existed for several decades[70,71] but has only been practically accessible to a small fraction of the community so far, thus the Colvars implementation aims at expanding its reach. This feature benefits from improvements in the extended-Lagrangian implementation, especially a new Langevin integrator (BAOA[72]), and reflecting boundary conditions for the extended variable, which is therefore strictly contained within the physically relevant interval $[0, 1]$.

The alchemical variable can then be used in biasing and free energy calculation methods such as ABF. This combination gives rise to lambda-ABF, a general and simple method for alchemical free energy estimation.[69] Stochastic diffusion in the alchemical space was found to enhance relaxation in configuration space compared to traditional, fixed-lambda simulations. Lambda-ABF is particularly useful when run over multiple walkers in the mwABF scheme. The alchemical coordinate is another tool in the Colvars toolbox and can be freely combined with other coordinates to create multidimensional biases and other custom methods. As an illustration, the decoupling free energy of the anti-inflammatory drug ketoprofen in TIP3P water was estimated efficiently using short simulations (4 replicas, 1 ns per replica) using Colvars in NAMD. Simulation details can be found in Supporting Information. The results are presented in Figure 4, and discussed in the ABF section below.

The Tinker-HP version of the alchemical variable is compatible with the CPU and GPU implementations. The NAMD implementation is compatible with both the GPU offloading and GPU-resident alchemical simulations[73] of NAMD 3,[3] although it does not yet support multiple time-stepping.

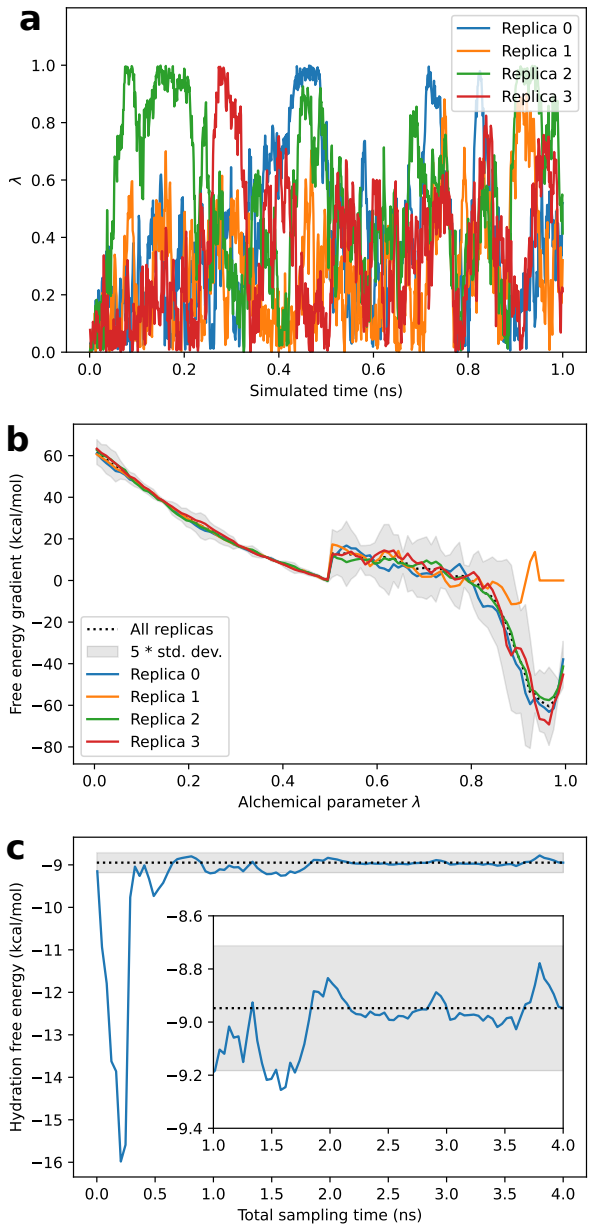

Figure 4: Analysis of an alchemical, multiple-walker lambda-ABF simulation of ketoprofen performed using Colvars and NAMD. a) Time trajectory of the alchemical parameter λ for the four replicas ("walkers"). b) Free-energy gradient as a function of the alchemical parameter λ, estimated based on the combined data of four replicas (black) and from each individual replica. The shaded interval is plus or minus the standard deviation between the replicas, weighted by the local sampling of each replica, and multiplied by a factor of 5 for visibility. Replica 1 does not affect the common estimate due to a lack of samples for λ close to 1. The discontinuous change at $\lambda = 0.5$ reflects the transition from electrostatic to Lennard-Jones decoupling. c) Convergence of the estimated free energy of hydration as a function of total sampling time combining the 4 replicas. Inset: vertical zoom on the final segment of the simulation. The shaded area denotes the average plus or minus the error estimate based on the dispersion of the gradient, represented in panel b).

New and improved biasing methods

Data-Driven Biasing Methods

In addition to the adaptive biasing techniques described in the previous Colvars article,[2] a range of sophisticated techniques have emerged for integrating experimental measurements and other types of data into simulations. These methods aim to produce structural ensembles consistent with target data while minimizing bias, following the maximum-entropy principle.[74] They require a forward model to predict target data from the MD ensemble and use a bias potential to align predictions with targets. Here, we describe methods in Colvars based on these concepts, categorizing them into techniques that focus on matching the mean value of an observable and those that target the probability distributions of CVs (which can be viewed as a continuous set of mean values).

Methods to restrain ensemble averages

The methods described in this section assume that the experimental data (or other data source) reflects an average of an observable, $\xi_i(\mathbf{X}, \boldsymbol{\Lambda_i})$ (forward model), over an ensemble of molecular configurations, $\mathbf{X}$, with $\boldsymbol{\Lambda_i}$ denoting model parameters. To align the simulated mean of $\xi_i(\mathbf{X}, \boldsymbol{\Lambda_i})$ with the target value, these methods employ a machine-learning bias potential, $V_t(\xi_i)$ (where t denotes time), during molecular dynamics simulations. The general expression of $V_t(\xi_i)$ follows the principle of maximum entropy:[74]

$$V_t(\xi_i) = -k_B T \sum_i \lambda_i(t)\, \xi_i(\mathbf{X}, \boldsymbol{\Lambda_i}) \tag{7}$$

where the parameter λ_i controls the mean value of $\xi_i(\mathbf{X}, \boldsymbol{\Lambda_i})$. To align the latter mean value with the desired target, the parameter λ_i is adjusted over time according to a gradient

descent optimization:

$$\lambda_i\left(t+dt\right)=\lambda_i\left(t\right)-\frac{\xi_i\left(\mathbf{X}\left[t\right],\mathbf{\Lambda_i}\right)-\xi_i^{tar}}{c\left(t\right)}dt \tag{8}$$

where ξ_i^{tar} is the target mean value and $c\left(t\right)$ controls the learning rate. The latter rate depends on the observable and can be tailored to ensure efficient convergence.[42,75]

Colvars offers two formulations of these methods: Adaptive Linear Bias (ALB, called `alb` in Colvars)[75] and Restrained-Average Dynamics (RAD, called `rad` in Colvars).[42] The key distinction is that RAD explicitly accounts for experimental and model uncertainty, thereby reducing the risk of over-fitting when handling multiple experimental datasets. In ALB, the target mean value is set precisely to the input experimental value ($\xi_i^{tar}=\xi_i^{exp}$ in eq. 8). In contrast, RAD optimizes ξ_i^{tar} to minimize bias while ensuring it remains within acceptable error limits. For Gaussian errors the time evolution of ξ_i^{tar} is given by:

$$\xi_i^{tar}\left[t\right]=\xi_i^{exp}-\frac{\eta_i^2}{\gamma_i\left[t\right]}\lambda_i\left(t\right) \tag{9}$$

where η_i is the estimated experimental error and γ_i is evolved to set an overall level of agreement between predicted and experimental data (e.g. $\sum_i|\xi_i^{tar}-\xi_i^{esp}|/N\eta_i\approx 1$, where N is the size of the dataset). Inputting both experimental errors and desired agreement levels helps prevent over-fitting and excessive bias. Besides uncertainties, RAD explicitly accounts for the model parameters, Λ_i, which, if unknown, can be optimized to reduce bias. However, the current implementation in Colvars only supports observables that are linear functions of the parameters Λ_i, such as the `deer` CV in the previous section on new coordinates. An example of RAD applied to DEER data is shown in Fig. 2. RAD is presently available in a separate branch of Colvars and is in the process of being integrated into the main branch.

Methods targeting CVs probability distributions

Probability distributions of molecular observables, such as label distances from DEER or Förster resonance energy transfer (FRET), and pair distances from radiation scattering, are alternative key targets for simulations. Focusing on these derived distributions rather than raw experimental signals (via methods described above) provides a unified framework for integrating these experiments (albeit an indirect one), avoiding the need for experiment-specific formulations.

Probability distributions of CVs can be represented as the ensemble average of density kernel functions $h_i\left[\xi\left(X\right)\right]$ (e.g. rectangular or Gaussian), centered on a specific CV value ξ_i, and covering the relevant CV range.[40,76] Thus, these distributions can be enforced in MD simulations using methods similar to those described above. Roux and coworkers[39,76] used this formulation to target the spin labels' distance distributions derived from DEER measurements. In this method, the mean values of the kernel functions are targeted using the restrained-ensemble technique. Namely, N simulation replicas are carried out and harmonic potentials, V_i, are applied on the mean value of the kernel functions across replicas:

$$V_i\left(X_1,...X_N\right)=\frac{1}{2}K\left(\frac{1}{N}\sum_{j=1}^{N}h_i\left[\xi\left(X_j\right)\right]-\bar{h_i}\right)^2 \tag{10}$$

where X_i denotes the molecular coordinates of replica i, K is the force constant and $\bar{h_i}$ is the target mean value. For large K and many replicas, the restrained-ensemble technique becomes statistically equivalent to the maximum entropy methods discussed in the previous section.[77] In practice, Roux et al.'s approach[76] employs a multiple-copy algorithm with replicas of spin-label moieties only, maintaining a single replica for the protein. Additionally, it applies a mean field approximation to scale the replica average statistics by N^2. This method is available in Colvars via the keyword `histogramRestraint` and is currently applicable only to one-dimensional probability distributions.

Besides representing kernel function mean values, probability distributions of CVs are

also linked to free energies via logarithmic scaling. This connection has been exploited by Marinelli,[40] White,[78] and coworkers to adapt free energy methods, such as metadynamics,[79] for targeting CVs distributions with an efficient single replica approach. Multiple replicas can be used to speed up convergence as in the multiple-walker metadynamics method,[80] but are not strictly required. Colvars provides an implementation of this methodology, called ensemble-biased metadynamics (EBMetaD) or `ebMeta` in Colvars, which can be applied to distributions of arbitrary dimensions. This approach is based on scaling the Gaussian functions in the metadynamics bias potential by the inverse of the target distribution function, referred to as ρ_{exp} (provided in input as a grid function):

$$V_{\mathrm{EBMetaD}}\left(\boldsymbol{\xi}, t\right) \;=\; \sum_{t'=\delta t, 2\delta t, \ldots}^{t'<t} \frac{W}{\exp\left(S_\rho\right) \rho_{exp}(\boldsymbol{\xi}(t'))} \prod_{i=1}^{N_{\mathrm{cv}}} \exp\left(-\frac{(\xi_i - \xi_i(t'))^2}{2\sigma_{\xi_i}^2}\right), \tag{11}$$

where S_ρ represents the target distribution differential entropy (W and σ_{ξ_i} are Gaussians' height and width respectively). At convergence, the EBMetaD bias potential is related to the free energy, $A(\boldsymbol{\xi})$, via:

$$A(\boldsymbol{\xi}) \;\simeq\; -V_{\mathrm{EBMetaD}}(\boldsymbol{\xi}) - \kappa_{\mathrm{B}} T \log \rho_{exp}(\boldsymbol{\xi}) \tag{12}$$

Namely, EBMetaD is statistically equivalent to the maximum entropy approaches discussed in the previous section when the observables are kernel density functions.[40] Like the aforementioned methods, EBMetaD has been extended to account for experimental errors[41] (which are also input as a grid function), using the same formulation as RAD. The extended version of EBMetaD is available in a development version of Colvars and is currently being implemented into the main branch.

Improvements to the ABF implementation

The Adaptive Biasing Force method (ABF) is a modified dynamics that enhances the sampling of a low-dimension space of collective variables based on an on-the-fly estimate of the free-energy gradient in that space.[81–83] Colvars has always included an implementation of ABF.[84] Since the time of the previous report,[2] the Colvars/ABF implementation was enriched with extended-system ABF method (eABF),[85] which can be coupled to an umbrella-integration free-energy estimator.[86] The eABF implementation in Colvars has formed the basis for developing methods that combine eABF and metadynamics[87,88] or well-tempered metadynamics,[88] as well as Gaussian-accelerated MD.[89] A Poisson integration algorithm has been added, enabling the code to directly produce integrated free-energy surfaces in dimensions 2 and 3.[90]

Here we report recent improvements to the multiple-walker, "shared ABF" variant, or mwABF.[15,91] In shared ABF, different copies of an ABF simulation run in parallel and share their free energy derivative data at finite time intervals, so that they apply nearly the same biasing force, and asymptotically so at long times. In the reworked implementation, each walker keeps a separate copy of its own samples and writes it to a separate set of output files, whereas only the first walker writes the collective free energy and gradient estimates. This lets users analyze the data from individual walkers, and the dispersion between them can be used as a convergence metric. This is illustrated in Figure 4, describing the alchemical decoupling of ketoprofen from water. Figure 4 displays the time dependence of λ. The λ values for the four replicas, all initialized at 0, cover most of the [0, 1] range. Of note, replica 1 never reaches the value of 1 in this short simulation: this does not affect the reliability of the overall estimate, which combines data from all replicas.

The free energy estimate converges rapidly over time, with fluctuations of 0.1 kcal/mol in the second half of the simulations. The dispersion between the data collected by all the replicas can now easily be analyzed to yield an error estimator.

The free-energy gradient estimated based on each replica's data is now output separately,

enabling the dispersion analysis of panel b. As visible in Figure 4a, the replicas' dynamics in λ space decorrelate rapidly. The benefit of mwABF is that the enhanced diffusion along the CV increases the rate at which replicas become decorrelated in the orthogonal (here, Cartesian) space.[69] Each replica visits each λ point several times, with a different history. Assuming that this decorrelation has happened, the errors in different replicas are independent, and the error in the common gradient estimate can be computed as the weighted standard deviation between the replicas. The error on the integrated free energy difference is estimated by assuming independence between the errors in different bins.[92] In this case, the decoupling free energy is estimated as -8.9±0.2 kcal/mol.

The compatibility of mwABF with eABF has been made complete, with improved handling of the data needed by the CZAR free energy estimator[85] across replicas.

Finally, the mwABF selection mechanism whereby walkers could be deleted and spawned in regions of lower sampling density has been extended to two-dimensional collective variable spaces. Note that this selection mechanism is implemented as a NAMD-specific Tcl script, whereas the general mwABF implementation is compatible with LAMMPS, NAMD, and Tinker-HP. This is not a major drawback because in our experience, the selection mechanism is not critical in ensuring diversity in the orthogonal space, which is the main benefit of multiple-walker sampling.[69]

Improvements in metadynamics-based methods

The metadynamics implementation initially detailed in ref.[2] included both the original method,[79] which aims to achieve uniform exploration of CV space, and its well-tempered variant,[93] which achieves Boltzmann sampling of the free energy in the CV space at a higher, specified temperature. Since then, the metadynamics approach has been extended to target any specified distribution in the CV space via the ensemble-biased metadynamics method,[40,41] detailed above. Additional improvements to metadynamics have focused on enhancing numerical accuracy and ease of use, which we discuss further here.

Boundary correction for metadynamics

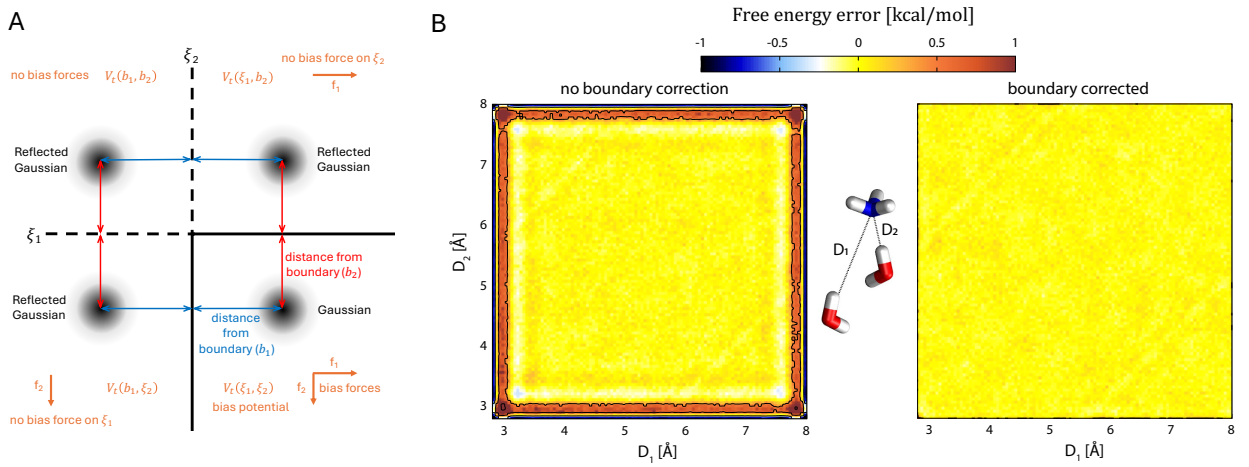

Figure 5: CVs boundary correction for metadynamics and its variants. (**A**) 2D schematic of the boundary correction method in Colvars. (**B**) Comparison of metadynamics simulations with (right) and without (left) boundary correction for ammonia coordinated by two water molecules in vacuum. Metadynamics simulations used two CVs (D_1 and D_2) representing distances between water oxygen atoms and ammonia nitrogen, with steep confining potentials outside [2.8Å, 8Å] and Gaussians added every ps (of height 0.006 kcal/mol and sigma 0.2Å). The panel compares free energy errors ($-k_BT \ln \rho(D_1, D_2)$, where ρ is the 2D histogram from the metadynamics sampling) from the last 250 ns of simulation (300K, 1 fs timestep). The boundary correction effectively removes errors from steep walls, as shown in the plot.

Regardless of the metadynamics variant employed, the finite spatial resolution of the Gaussian bias leads to systematic errors near steep walls or mathematical boundaries of CVs. These errors accumulate over the course of the simulation, hindering correct sampling in these regions, which can also cause simulation instabilities during extended runs. As a result, the accuracy of the calculated free energy in these areas can be compromised[94,95] (Fig. 5**B**; left plot). To address this issue, we implemented a boundary correction in Colvars across arbitrary dimensions, based on concepts from previous research.[94,96,97] The approach consists in applying reflecting conditions at the boundaries[94,97] and removing biasing force components along CVs, when they exceed their specified boundary limits[96] (i.e. when they are not mathematical limits). This boundary condition can be enabled in Colvars with the keyword `useHillsReflection`. When activated, additional Gaussians are placed outside the boundaries at locations reflected relative to the original ones and with the same height

and width of the original Gaussians (Fig. 5**A**). To improve efficiency, this condition is applied only to Gaussians that are within a cutoff distance from the boundary (by default 6 times the Gaussians sigma, but can be modified with the keyword `reflectionRange`).

To prevent artifacts when CVs cross boundaries, this also includes Gaussians that are simultaneously reflected across multiple CVs, considering all possible combinations of single, double, or multiple concurrent reflections (Fig. 5**A**). This approach produces a smooth distribution of additional Gaussians on a shell surrounding the exterior of the CV space enclosed by the boundaries. Although the reflected Gaussians are placed outside the boundaries, potentially beyond the mathematically valid region of the CV space, their tails still affect the bias potential and forces within the boundaries. This effect helps reaching stationary conditions and prevents the accumulation of systematic errors. Due to the symmetry of the reflecting condition, biasing force components vanish at the boundaries that are perpendicular to them. Therefore, the aforementioned force components can be safely set to zero outside these boundaries (Fig. 5**A**) if they are not strict mathematical limits (and may be exceeded during the simulation). This avoids discontinuities when crossing boundaries and further reduces systematic errors. Consistently, outside the boundaries we maintain a constant bias potential in the direction of the removed force components, matching the value at the nearest exceeded boundary (Fig. 5**A**).

In summary, the boundary correction method implemented is effective and accurate as illustrated here on a model system (Fig. 5**B**). It is compatible with any metadynamics variant in Colvars and requires no additional parameters or significant computational overhead. Additionally, it can be used with both intrinsically limited CVs and those confined by flat-bottom potentials.

When the metadynamics bias is introduced by projecting Gaussians onto a grid (enabled by default), but the boundary correction above is not employed, it is recommended to either:

- extend the grid boundaries to enclose all wall restraints, as detailed in the Colvars documentation, or

- place the grid boundaries at the true mathematical boundaries of each CV whenever possible.

Alternatively, to derive the free energy using reweighting approachs[98–100] or a force-based method.[101] These methods can effectively employ free-energy estimators other than the biasing potential itself, and thus may also be useful for assessing the quality of sampling.

Usability improvements in metadynamics

One of the usability improvements in metadynamics is the ability to specify the width of Gaussian hills along each dimension, in units of the corresponding CVs, using the `gaussianSigmas` keyword; this option is mutually exclusive with the pre-existing `hillWidth` option, which specifies the width along all dimensions as a single number of grid points. Lastly, when running multiple-walker metadynamics[80] it is now possible to use the same Colvars configuration file for all walkers, provided that the MD engine is launched in a multiple-replicas configuration. This is achieved by setting automatically the default value for the `replicaID` keyword whenever the `-multidir` option is used in GROMACS, or `-partition` flag in LAMMPS, the `+replicas` flag in NAMD, or the `replicas` keyword in Tinker-HP.

On-the-fly Probability Enhanced Sampling (OPES)

An implementation of On-the-fly Probability Enhanced Sampling (OPES),[102] a generalization of metadynamics, is being included into Colvars at the time of writing this report. The implementation is based on the implementation of OPES already distributed with PLUMED,[6] and therefore retains many of its features save for some changes for the new platform. For example, checkpointing an OPES simulation does not rely on accessory files, but rather on the state file written by Colvars (NAMD and Tinker-HP) or on the MD engine's own checkpoint file (GROMACS and LAMMPS).

Bias-exchange schemes

Colvars implicitly supports the temperature-exchange method,[10] by communicating the value of its current biasing potential to the MD engine, which uses it to compute the probability of exchanges between replicas. Additionally, Colvars provides a direct implementation for other replica-exchange schemes, where the property being exchanged is an umbrella-sampling restraint[103] or a metadynamics bias.[104] Both schemes are currently supported using the Colvars scripting interface described below, using scripts available from the NAMD website (replica-exchange umbrella sampling) or from the Colvars website (bias-exchange metadynamics). As is common with most implementations of replica-exchange schemes, synchronicity and fast communication between replicas are also required. Therefore, this feature is currently limited to NAMD, where script-driven communication between replicas[105] is available in most typical cluster installations of NAMD.

A noteworthy aspect of the aforementioned implementation of bias-exchange metadynamics is the use of a single configuration file to define the CVs and their metadynamics biases on all replicas simultaneously. During a simulation, only one metadynamics bias is kept active by each replica, with no forces being applied to the CVs that are not involved in that bias. When an exchange is attempted between two replicas, the information accumulated by the two biases is mutually communicated between the respective replicas. If the exchange is successful, the two biases will be swapped and their permutation will persist until the next exchange attempt and be restored when continuing a simulation from a checkpoint. This approach has two advantages:

- by performing all exchanges internally within Colvars, all other features of the MD engine remain compatible with the sampling scheme;
- because all CVs are always recorded, regardless of whether forces are applied to them, analyzing the simulated trajectories does not require recomputing from Cartesian coordinates, potentially saving significant storage space.

Lastly, the bias-exchange implementation is easily generalizable to sampling schemes other than the two mentioned above,[103,104] providing a basis for developing new methods.

Adiabatic Bias MD

Adiabatic Bias MD (ABMD) is a time-dependent, non-equilibrium biasing method that produces reactive trajectories for rare events by enhancing the forward motion of a progress coordinate ξ_t.[106] Forward fluctuations of the coordinate are selected by applying a history-dependent harmonic potential $V_t(\xi_t)$ centered on the highest value reached by ξ_t over the past trajectory (high-water mark).

$$V_t(\xi_t) = \begin{cases} \frac{1}{2}k\left(\xi_t - \xi_t^{\mathrm{ref}}\right)^2 & \text{if } \xi_t < \xi_t^{\mathrm{ref}} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where ξ_t^{ref} is the high-water mark at time t, bounded by a user-defined stopping value ξ^{stop}:

$$\xi_t^{\mathrm{ref}} = \min\left(\max_{s=0}^{t} \xi_s, \xi^{\mathrm{stop}}\right). \tag{14}$$

ABMD (also called ratchet-and-pawl MD) has been successfully applied in particular to protein folding processes.[106–108] Recently, it has proved efficient at producing diverse binding pathways and poses for a nucleotide ligand inside the cavity of Uncoupling Protein 1, a proton transporter from the mitochondrial inner membrane.[109]

Previously, ABMD had been implemented as a simple scripted bias, showcasing the efficiency of the scripting interface for rapidly implementing new methods. Because the biases work in low-dimension colvar space, the overhead associated with a scripting language is negligible and prototype implementations are efficient enough to be used for production. However, the new, native C++ implementation of ABMD in Colvars makes it more flexible, and portable to all MD back-ends, including GROMACS and LAMMPS.

The C++ implementation adds the `abmd` bias keyword, with the following options: the

name of the collective variable undergoing the bias, the harmonic force constant k, the stopping value ξ^{stop}, and an optional boolean value indicating that the values should decrease over time rather than increase.

Custom-defined collective variables and biases

Extending the functionality of the Colvars library without changing its source code is supported through multiple mechanisms, which are presented here. This section discusses the goal of defining a collective variable $\xi(\mathbf{X})$ such as:

$$\xi(\mathbf{X}) = f\left(\zeta_1(\mathbf{X}), \zeta_2(\mathbf{X}), \ldots\right) \tag{15}$$

where $\zeta_1(\mathbf{X})$, $\zeta_2(\mathbf{X})$, ... are functions implemented by Colvars, and $f(\ldots)$ is a function defined at runtime by the user. The following sections describe three different approaches to implement $f(\ldots)$.

Linear and polynomial superposition

Since its first release, Colvars allows defining the function $f(\ldots)$ as a weighted sum of powers of $\zeta_i(\mathbf{X})$:

$$f\left(\zeta_1(\mathbf{X}), \zeta_2(\mathbf{X}), \ldots\right) = c_1\zeta_1(\mathbf{X})^{p_1} + c_2\zeta_2(\mathbf{X})^{p_2} + \ldots \tag{16}$$

where c_i are coefficients and p_i integer exponents, defined by the keywords `componentCoeff` and `componentExp`, respectively. Typical use cases of this feature range from sums of the same quantity computed over multiple individual molecules (e.g. the total dipole moment), to variables describing multiple states of a whole system.[63]

Custom mathematical functions

In all official releases of NAMD and LAMMPS, as well as patched releases of GROMACS, VMD and Tinker-HP, Colvars allows defining $f(\ldots)$ as a closed-form mathematical expression. This functionality is provided by the Lepton library,[61] originally developed within the OpenMM package;[5] the same library is also responsible for the automatic calculation of the derivatives of $f(\ldots)$ with respect to its arguments, $\partial f/\partial\zeta$. For example, the residual dipolar coupling (RDC) of a pair of bonded atoms may be defined, save for a multiplicative constant, as:

$$f\,(\texttt{dz},\texttt{d}) = \texttt{1.5 * (dz/d)\^{}2 - 0.5} \tag{17}$$

where the right-hand side of eq. 17 is the expression directly usable as the argument of the `customFunction` keyword and `dz` and `d` are, respectively, the labels of a `distanceZ` and a `distance` component defined on the pair of atoms.

Defining variables and biases by scripted code

In addition to controlling the flow of a simulation (see below), scripting languages may also be used to define additional code at runtime to implement existing collective variables and biases. Two distinct keywords are available:

- `scriptedFunction`, which defines the root name of two related functions that implement, respectively, $f(\zeta_1,\zeta_2,\ldots)$ and its derivatives $\partial_{\zeta_1} f(\zeta_1,\zeta_2,\ldots)$, $\partial_{\zeta_2} f(\zeta_1,\zeta_2,\ldots)$, …; during runtime, ζ_1, ζ_2, … are evaluated and their values passed as arguments to the scripted functions;

- `scriptedColvarForces`, which provides the name of a function that can compute and apply biasing forces on multiple CVs, thus allowing implementation of a new type of bias; at runtime, no arguments are provided to this function, which will obtain current values of the CVs and apply forces to them through the Colvars scripting API detailed below.

Currently, this feature supports only the Tcl scripting language, which is available in all installations of NAMD and VMD. Additionally, Colvars in Tinker-HP is also built with an embedded Tcl interpreter that supports callbacks, as used in.[69] A typical use case of scripted functions has been the prototyping of path CVs (see New coordinates above) and Euler-angle CVs,[18] which were later implemented in the library itself. Similarly, Adiabatic Bias MD (ABMD)[106] was initially implemented as a scripted bias and used in production,[109] but has since been reimplemented to be available in GROMACS and LAMMPS, which do not support Tcl scripting. Support for Python-scripted CVs and biases is under development: its implementation details will be tuned to comply with the packaging systems adopted by each MD engine as their support for Python improves. At the time of writing, among the MD engines supported by Colvars only LAMMPS offers a precompiled package that supports Python callbacks, albeit from a third-party package.

New or improved interfaces

GROMACS interface

The Colvars library is now part of the official GROMACS package,[110] starting with the GROMACS 2024 release. Previous releases (from 2020 to 2023) were also supported through patched versions of the GROMACS code base, or by applying patches manually.

Starting with the 2024 release, a copy of the Colvars library is included in the upstream version of GROMACS and is built by default during the compilation. The library is compatible with most of the GROMACS integrators, GPU offload, thread MPI and the Multiple Time Step feature; starting from version 2025, Colvars can also leverage the multi-simulation framework in GROMACS. A few Colvars features such as custom functions and protein secondary structure CVS are not yet included in the standard GROMACS releases, but are available as Colvars development branches.

To develop the Colvars-GROMACS interface, the C++ proxy class of Colvars was ex-

tended to match the "MDModules" framework of GROMACS, which simplifies adding external modules without modifying core GROMACS files. Other changes, of a more visible nature ti the user, were introduced to conform to the GROMACS workflow. For example, all input files (including the GROMACS MD parameters file, the Colvars configuration file and their dependencies) are embedded into the portable binary run input file of GROMACS (*.tpr*) during the '*pre-processing*' step. This ensures the validity of the input files and promotes reproducibility. Furthermore, all Colvars information is stored in the GROMACS checkpoint file (*.cpt*) during the simulation, allowing users to restart or continue a Colvars-GROMACS simulation transparently. Complete documentation is available in the GROMACS manual.[111]

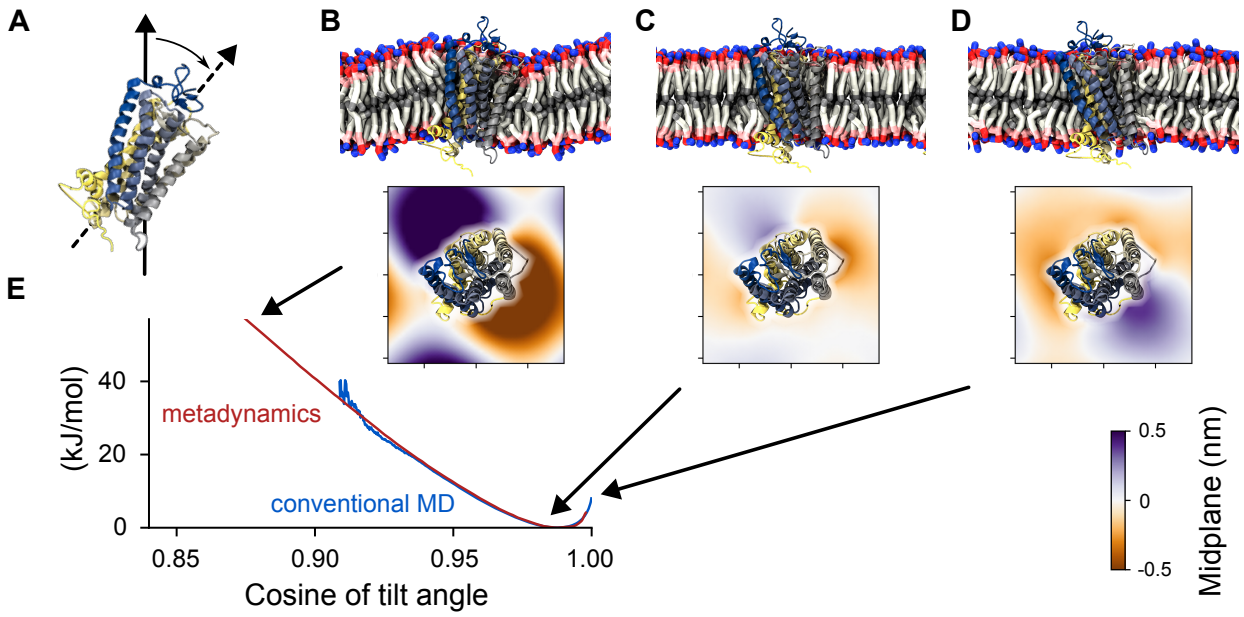

Figure 6: Simulation results, obtained using Colvars and GROMACS, of rhodopsin in a coarse-grained lipid bilayer as a function of the tilt angle θ (**A**) from the protein's initial orientation. Shown are simulation snapshots at high tilt (**B**), at the free-energy minimum (**C**), and at the initial orientation (**D**). Each heat map shows the deflection of the bilayer midplane around the protein (in nm) at the corresponding tilt angle. Panel **E** shows the free energy profile as a function of $\cos(\theta)$ computed using conventional MD (blue) or metadynamics (red).

Figure 6 illustrates the combined use of Colvars in a typical GROMACS simulation, i.e. a membrane-protein system simulated in coarse-grained representation with the MARTINI

force field.[112] See the Supporting Information for computational details. Lateral diffusion of the protein parallel to the bilayer and rotations around axes perpendicular to it were prevented by restraints applied onto `distanceXY` and `spinAngle` variables, respectively. The orientation of the protein during a simulation was quantified by a `tilt` variable, using as reference the initial coordinates. A 20 μs conventional MD simulation without any biasing forces on the tilt variable shows that the most favorable configuration has a $\approx 10^\circ$ angle ($\cos(\theta) \approx 0.985$) with respect to the initial orientation (Fig. 6E). Enhanced sampling with metadynamics, also for 20 μs, allows to explore much higher tilt angles ($\cos(\theta) < 0.9$). Comparison of the membrane's midplane deflection maps between the relevant states reveals that the minimum-energy orientation (Fig. 6C) minimizes the distortion of the membrane observed in other states (Fig. 6B,D).

Improved LAMMPS and NAMD interfaces

The Colvars-LAMMPS interface was recently improved by adding support for scripted workflows (see below) through the LAMMPS `fix_modify` command to send command-line instructions to Colvars. Syntactical differences notwithstanding, this functionality is similar to how Tcl scripting is used in NAMD and VMD. Typical use cases include the ability to customize a run's input based on the results of prior simulation steps, or controlling when and which output files are written. It is worth clarifying that the latter functionality is available alongside the standard LAMMPS checkpointing mechanism, without replacing it. In fact, using the LAMMPS restart file to checkpoint the Colvars has also become more efficient in recent versions, by supporting an unformatted (i.e. binary) representation of the Colvars state data, consistent with the LAMMPS state data such as atomic coordinates, velocities, etc.

The first MD engine that officially supported Colvars is NAMD, owing to the "*GlobalMaster*" infrastructure introduced to support biasing schemes such as steered molecular dynamics, which preceded the introduction of Colvars. The GlobalMaster-based interface

between Colvars and NAMD has been extended since its introduction through the following improvements:

- the computation of centers of mass of atom groups and their forces is distributed across parallel tasks;
- when multiple colvar components or biases are defined, their computation can be distributed over multiple threads of the first task;
- the values of volumetric maps[63] and of the alchemical λ variables[69] are communicated and processed by Colvars in the same way as atoms and groups centers of mass.

Lastly, although GlobalMaster has traditionally been limited to CPU-based or GPU-offload simulations, it has recently been extended by the NAMD developers to support CPU-based features like Colvars in a GPU-resident run,[3] an improvement that will be discussed in more detail in a different manuscript.

Tinker-HP

Colvars is now interfaced with the Tinker-HP package,[113] which features high-performance implementations of polarizable force fields and machine-learned potentials on CPUs and GPUs. The Colvars library is included in both the CPU and GPU versions of Tinker-HP 1.2.[69] To build this interface, the C++ proxy class of Colvars was extended with a C layer, which is linked to the Fortran core of Tinker-HP. This provides a template for C and Fortran-based interfaces, further increasing the portability of Colvars. A highlight of the Colvars/Tinker-HP interface is the lambda-dynamics feature (detailed in the section on new coordinates above), which is the core of the lambda-ABF method for alchemical free energy estimation.[69]

Contrary to VMD and NAMD, Tinker-HP does not natively contain a Tcl interpreter. Colvars has been extended with the possibility of building a dedicated, embedded Tcl interpreter. This is enabled in standard Tinker-HP builds, bringing the flexibility of scripted

variables and biases implemented as Tcl callbacks. Since Tinker-HP offers no built-in user interface for executing Tcl scripts, we have implemented the Colvars keyword `sourceTclFile`, which runs a specified script at startup in the Tcl interpreter that is either linked into the back-end or directly embedded in Colvars.

The VMD interface and the Colvars Dashboard

The Colvars library is linked into executable builds of VMD and can be controlled via a command-line scripting interface based on the Tcl language and the `cv` command which controls the scripting API detailed below. This interface offers the same functionality to access the molecular topology information available in MD engines. Although no MD integration schemes are available in VMD, the energy and atomic forces produced by Colvars can be analyzed for any configuration loaded in VMD. This feature may be used either to analyze the output of a previous simulation or to assess the feasibility of adding or modifying a biasing scheme to an existing simulation.

In the typical scenario where VMD is used interactively, the above command-line interface provides the basis for a graphical user interface: the Colvars Dashboard.[7] The Dashboard serves two roles: facilitate the analysis of datasets of molecular configurations to interpret them in terms of colvars, and provide helpers for setting up biased simulations. The configuration of variables and biases can be saved to a file that is directly readable by the supported MD simulation programs (GROMACS, LAMMPS, NAMD, and Tinker-HP).

The interactive display features of the Dashboard include a timeline display to explore the time dependency of the colvars, and pairwise scatterplots to analyze pairwise correlations. Displaying the atoms involved in a variable is useful for troubleshooting atom selections. Colvar gradients with respect to atomic coordinates can be shown as arrows, helping scientists understand the sensitivity of colvars to individual atom coordinates.

New features were added to the Dashboard since the reference publication, in addition to many small stability and usability improvements. Notably, interactive histograms can

now be computed and displayed. Clicking within the histogram changes the displayed frame to that with the closest value of the colvar. Arrow keys navigate the trajectory sorted by colvar values, which is particularly useful for exploring individual modes in multimodal distributions of colvars. The gradient display feature has been extended to all coordinates, including vector ones.

A pre-computed colvars trajectory can now be imported from a `.colvars.traj` file, typically output by a Colvars-enabled simulation. These values are cached and associated with loaded molecular configurations in VMD. This enables the visualization of parameters that cannot be computed in VMD, such as the dynamic alchemical parameter within a lambda-ABF trajectory.[69] It is also useful for troubleshooting any discrepancy between colvar values computed in VMD and in Colvars linked to MD engines. Lastly, the set of templates for collective variables and biasing algorithms has been extended and improved to cover a much larger set of the Colvars features.

Code design and functionality improvements

Scripting interface: workflow control

The range of scientific problems where molecular dynamics (MD) simulations are used and the amount of computational power available to each project are both increasing steadily. Therefore, many simulation packages also offer the ability to customize the flow of an MD simulation via scripting languages. The two languages most often used in MD simulations, Tcl and Python, support this goal in distinct ways, such as allowing easy integration with simulation software (Tcl) or leveraging a rich set of features and knowledge base (Python). Furthermore, certain MD simulation packages, such as LAMMPS, provide control instructions that provide, in essence, a specialized scripting language for that application. Colvars supports each of the three above languages: currently, the most mature and feature-rich interface is with the Tcl language (specifically, via the `cv` command), available in all deploy-

ments of NAMD and VMD, where it also forms the basis of the Colvars Dashboard graphical user interface described previously.

Through scripted code, individual collective variables and their biases may be added, removed, or modified midway through a simulation, which allows for fine-tuning performance or dynamically redefining the enhanced sampling scheme. Notable examples of this feature are the bias-exchange metadynamics method (see the section on metadynamics improvements above), or the implementation of path-learning methods such as milestoning using the ScMiles2 software,[114] both of which are implemented in Tcl.

Continuous integration (CI) testing

Integration and testing of all new code are performed via the public GitHub repository https://github.com/Colvars/colvars. To ensure the reliability of the library in any new release of each supported package, the Colvars library receives continuous updates and improvements and is deployed following a rolling-release model. To support this model, integration of any source code revisions is performed conditionally on the outcome of automated tests of the stand-alone library and the back-end engines. This measure guarantees that changes maintain the correctness of the computations across supported packages and platforms.

Conclusion

Colvars has been enhanced in many respects over the past decade, making it more broadly available and more useful to the community. Nonetheless, progress is not as immediately reflected in the numbering scheme of its versions: Colvars does not have major releases, because it is made available as part of the standard distribution of mainstream molecular simulation and analysis packages. Beyond that distribution method, advanced users may obtain the Colvars source code directly from its repository, accessing a continuous stream

of well-tested new features and improvements. Experimental features are also available in development branches, as they are discussed with the associated issues or pull requests, following modern software development practices. This manuscript surveyed improvements made since the Colvars reference paper was published,[2] with particular emphasis on novel features and increased portability. Enhancements in performance and scalability, including those currently under development, will be covered in separate publications.

In their early days, MD simulations were primarily used to illustrate models previously set forth by experimental measurements. However, dramatic improvement in the performance of MD simulations now allows for prediction of the outcome of experiments yet to be performed. Therefore, MD is especially effective at addressing problems that are otherwise challenging for methods such as experimental structure detection and machine-learning models trained on those structures. The applications shown here demonstrate, for example, that collective variable-based methods are effective at estimating the strength of binding between proteins and their ligands, or the mechanism of interaction between proteins and their surrounding lipid membrane. Importantly, by close integration of, and co-distribution with, Colvars and the packages it supports, we minimize the barriers to access such enhanced-sampling approaches, allowing widespread adoption of state-of-the-art methodologies to address critical challenges remaining in molecular science.

Supporting Information Available

Computational details for simulations and mathematical expression for biasing forces in a moving frame of reference.

Acknowledgments

We are grateful to the people who helped improve the library in various ways and interface it with back-ends, especially Olivier Adjoua, Alejandro Bernardin, Ronak Buch, Eric Bohm,

David Clark, Jeff Comer, Haohao Fu, David Hardy, Berk Hess, Michele Invernizzi, Louis Lagardère, Magnus Lundborg, Julio Maia, Mohammad Soroush Baraghi, Josh Vermaas, Andrew White and Wei Zhang.

GF and LRF acknowledge support from the Divisions of Intramural Research of the National Institute of Neurological Disorders and Stroke (NS003139) and the National Heart, Lung and Blood Institute; computing resources were provided in part by NIH, through the "Biowulf" cluster.

FM acknowledges support from Medical College of Wisconsin startup funds, as well as computational resources and technical assistance provided by the Research Computing Center at the Medical College of Wisconsin.

HC acknowledges support from the Agence Nationale de la Recherche (ProteaseInAction and LOR-AI) and the National Institute of Health (P41-GM104601 and R24-GM145965).

CC acknowledges support from the Agence Nationale de la Recherche (ProteaseInAction and LOR-AI) and the European Research Council (project 101097272 MilliInMicro).

JH acknowledges support from the French National Research Agency under grant LABEX DYNAMO (ANR-11-LABX-0011).

HC, CC, and JH acknowledge support from the Centre National de la Recherche Scientifique through its international research initiative (LIA CNRS-UIUC)

References

(1) Hénin, J.; Lelièvre, T.; Shirts, M. R.; Valsson, O.; Delemotte, L. Enhanced Sampling Methods for Molecular Dynamics Simulations [Article v1.0]. *Living J. Comp. Mol. Sci.* **2022**, *4*, 1583.

(2) Fiorin, G.; Klein, M. L.; Hénin, J. Using collective variables to drive molecular dynamics simulations. *Mol. Phys.* **2013**, *111*, 3345–3362.

(3) Phillips, J.; Hardy, D.; Maia, J.; Stone, J.; Ribeiro, J.; Bernardi, R.; Buch, R.;

Fiorin, G.; Hénin, J.; Jiang, W. et al. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *J. Chem. Phys.* **2020**, *153*, 044130.

(4) Páll, S.; Zhmurov, A.; Bauer, P.; Abraham, M.; Lundborg, M.; Gray, A.; Hess, B.; Lindahl, E. Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. *J. Chem. Phys.* **2020**, *153*, 134110.

(5) Eastman, P.; Galvelis, R.; Peláez, R. P.; Abreu, C. R. A.; Farr, S. E.; Gallicchio, E.; Gorenko, A.; Henry, M. M.; Hu, F.; Huang, J. et al. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *J. Phys. Chem. B* **2024**, *128*, 109–116, PMID: 38154096.

(6) Tribello, G. A.; Bonomi, M.; Branduardi, D.; Camilloni, C.; Bussi, G. PLUMED 2: New feathers for an old bird. *Comput. Phys. Commun.* **2014**, *185*, 604–613.

(7) Hénin, J.; Lopes, L. J. S.; Fiorin, G. Human Learning for Molecular Simulations: The Collective Variables Dashboard in VMD. *Journal of Chemical Theory and Computation* **2022**, *18*, 1945–1956.

(8) Humphrey, W.; Dalke, A.; Schulten, K. VMD: visual molecular dynamics. *J. Mol. Graph.* **1996**, *14*, 33–38.

(9) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in 't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D. et al. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comput. Phys. Commun.* **2022**, *271*, 108171.

(10) Sugita, Y.; Okamoto, Y. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.* **1999**, *314*, 141–151.

(11) Teo, I.; Mayne, C. G.; Schulten, K.; Lelièvre, T. Adaptive Multilevel Splitting Method

for Molecular Dynamics Calculation of Benzamidine-Trypsin Dissociation Time. *J. Chem. Theory Comput.* **2016**, *12*, 2983–2989.

(12) Lopes, L. J. S.; Lelièvre, T. Analysis of the adaptive multilevel splitting method on the isomerization of alanine dipeptide. *J. Comput. Chem.* **2019**, *40*, 1198–1208.

(13) Elber, R. Milestoning: An efficient approach for atomically detailed simulations of kinetics in biophysics. *Annual review of biophysics* **2020**, *49*, 69–85.

(14) Zuckerman, D. M.; Chong, L. T. Weighted ensemble simulation: review of methodology, applications, and software. *Annual review of biophysics* **2017**, *46*, 43–57.

(15) Comer, J.; Phillips, J. C.; Schulten, K.; Chipot, C. Multiple-replica strategies for free-energy calculations in NAMD: multiple-walker adaptive biasing force and walker selection rules. *J. Chem. Theory Comput.* **2014**, *10*, 5276–5285.

(16) Woo, H.-J.; Roux, B. Calculation of absolute protein–ligand binding free energy from computer simulations. *PNAS* **2005**, *102*, 6825–6830.

(17) Gumbart, J. C.; Roux, B.; Chipot, C. Standard Binding Free Energies from Computer Simulations: What Is the Best Strategy? *J. Chem. Theory Comput.* **2013**, *9*, 794–802.

(18) Fu, H.; Cai, W.; Hénin, J.; Roux, B.; Chipot, C. New Coarse Variables for the Accurate Determination of Standard Binding Free Energies. *J. Chem. Theory Comput.* **2017**, *13*, 5173–5178.

(19) Coutsias, E. A.; Seok, C.; Dill, K. A. Using quaternions to calculate RMSD. *J. Comput. Chem.* **2004**, *25*, 1849–1857.

(20) Fu, H.; Gumbart, J. C.; Chen, H.; Shao, X.; Cai, W.; Chipot, C. BFEE: A User-Friendly Graphical Interface Facilitating Absolute Binding Free-Energy Calculations. *J. Chem. Inf. Model.* **2018**, *58*, 556–560.

(21) Fu, H.; Chen, H.; Cai, W.; Shao, X.; Chipot, C. BFEE2: Automated, Streamlined, and Accurate Absolute Binding Free-Energy Calculations. *J. Chem. Inf. Model.* **2021**, *61*, 2116–2123.

(22) De Donder, T. *L'affinité*; Gauthier–Villars: Paris, 1927.

(23) Kirkwood, J. G. Statistical mechanics of fluid mixtures. *J. Chem. Phys.* **1935**, *3*, 300–313.

(24) E, W.; Ren, W.; Vanden-Eijnden, E. String method for the study of rare events. *Phys. Rev. B* **2002**, *66*, 052301.

(25) Pan, A. C.; Sezer, D.; Roux, B. Finding transition pathways using the string method with swarms of trajectories. *J. Phys. Chem. B* **2008**, *112*, 3432–3440.

(26) Chen, H.; Ogden, D.; Pant, S.; Roux, B.; Moradi, M.; Cai, W.; Tajkhorshid, E.; Chipot, C. A companion guide to the string method with swarms of trajectories. Characterization, performance, and pitfalls. *J. Chem. Theory Comput.* **2022**, *18*, 1406–1422.

(27) Jung, H.; Covino, R.; Arjun, A.; Leitold, C.; Dellago, C.; Bolhuis, P. G.; Hummer, G. Machine-guided path sampling to discover mechanisms of molecular self-organization. *Nature Comput. Sci.* **2023**, *3*, 334–345.

(28) Chen, H.; Roux, B.; Chipot, C. Discovering reaction pathways, slow variables, and committor probabilities with machine learning. *J. Chem. Theory Comput.* **2023**, *19*, 4414–4426.

(29) Kang, P.; Trizio, E.; Parrinello, M. Computing the committor with the committor to study the transition state ensemble. *Nature Comput. Sci.* **2024**, *4*, 451–460.

(30) Peters, B. Reaction coordinates and mechanistic hypothesis tests. *Ann. Rev. Phys. Chem.* **2016**, *67*, 669–690.

(31) Rogal, J. Reaction coordinates in complex systems—A perspective. *Euro. Phys. J. B* **2021**, *94*, 223.

(32) Chipot, C. Free energy methods for the description of molecular processes. *Ann. Rev. Biophys.* **2023**, *52*, 113–138.

(33) Branduardi, D.; Gervasio, F. L.; Parrinello, M. From A to B in free energy space. *J. Chem. Phys.* **2007**, *126*, 054103.

(34) Díaz Leines, G.; Ensing, B. Path finding on high-dimensional free energy landscapes. *Phys. Rev. Lett.* **2012**, *109*, 020601.

(35) Fu, H.; Chen, H.; Blazhynska, M.; Goulard Coderc de Lacam, E.; Szczepaniak, F.; Pavlova, A.; Shao, X.; Gumbart, J. C.; Dehez, F.; Roux, B. et al. Accurate determination of protein:ligand standard binding free energies from molecular dynamics simulations. *Nat. Protoc* **2022**, *17*, 1114–1141.

(36) Salari, R.; Joseph, T.; Lohia, R.; Hénin, J.; Brannigan, G. A Streamlined, General Approach for Computing Ligand Binding Free Energies and Its Application to GPCR-Bound Cholesterol. *J. Chem. Theory Comput.* **2018**, *14*, 6560–6573.

(37) Santiago-McRae, E.; Ebrahimi, M.; Sandberg, J. W.; Brannigan, G.; Hénin, J. Computing absolute binding affinities by Streamlined Alchemical Free Energy Perturbation [Article v1.0]. *Living J. Comp. Mol. Sci.* **2023**, *5*, 2067.

(38) Ebrahimi, M.; Hénin, J. Symmetry-Adapted Restraints for Binding Free Energy Calculations. *J. Chem. Theory Comput.* **2022**, *18*, 2494–2502.

(39) Islam, S. M.; Stein, R. A.; Mchaourab, H. S.; Roux, B. Structural Refinement from Restrained-Ensemble Simulations Based on EPR/DEER Data: Application to T4 Lysozyme. *J. Phys. Chem. B* **2013**, *117*, 4740–4754.

(40) Marinelli, F.; Faraldo-Gómez, J. Ensemble-Biased Metadynamics: A Molecular Simulation Method to Sample Experimental Distributions. *Biophysical Journal* **2015**, *108*, 2779–2782.

(41) Hustedt, E. J.; Marinelli, F.; Stein, R. A.; Faraldo-Gómez, J. D.; Mchaourab, H. S. Confidence Analysis of DEER Data and Its Structural Interpretation with Ensemble-Biased Metadynamics. *Biophysical Journal* **2018**, *115*, 1200–1216.

(42) Marinelli, F.; Fiorin, G. Structural Characterization of Biomolecules through Atomistic Simulations Guided by DEER Measurements. *Structure* **2019**, *27*, 359–370.e12.

(43) Schmidt, T.; Wälti, M. A.; Baber, J. L.; Hustedt, E. J.; Clore, G. M. Long Distance Measurements up to 160 Å in the GroEL Tetradecamer Using Q-Band DEER EPR Spectroscopy. *Angew. Chem. Int. Ed.* **2016**, *55*, 15905–15909.

(44) Milov, A.; Ponomarev, A.; Tsvetkov, Y. Electron-electron double resonance in electron spin echo: Model biradical systems and the sensitized photolysis of decalin. *Chem. Phys. Lett.* **1984**, *110*, 67–72.

(45) Milov, A. D.; Tsvetkov, Y. D. Double electron-electron resonance in electron spin echo: Conformations of spin-labeled poly-4-vinilpyridine in glassy solutions. *Appl. Magn. Reson.* **1997**, *12*, 495–504.

(46) Jeschke, G.; Chechik, V.; Ionita, P.; Godt, A.; Zimmermann, H.; Banham, J.; Timmel, C. R.; Hilger, D.; Jung, H. DeerAnalysis2006—a comprehensive software package for analyzing pulsed ELDOR data. *Appl. Magn. Reson.* **2006**, *30*, 473–498.

(47) Brandon, S.; Beth, A. H.; Hustedt, E. J. The global analysis of DEER data. *J. Magn. Reson.* **2012**, *218*, 93–104.

(48) Edwards, T. H.; Stoll, S. A Bayesian approach to quantifying uncertainty from experimental noise in DEER spectroscopy. *J. Magn. Reson.* **2016**, *270*, 87–97.

(49) Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks* **1991**, *4*, 251–257.

(50) Mardt, A.; Pasquali, L.; Wu, H.; Noé, F. VAMPnets for deep learning of molecular kinetics. *Nat. Commun* **2018**, *9*, 5.

(51) Wang, Y.; Ribeiro, J. M. L.; Tiwary, P. Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics. *Nat. Commun* **2019**, *10*, 1–8.

(52) Chen, W.; Sidky, H.; Ferguson, A. L. Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets. *J. Chem. Phys.* **2019**, *150*, 214114.

(53) Sidky, H.; Chen, W.; Ferguson, A. L. Machine learning for collective variable discovery and enhanced sampling in biomolecular simulation. *Mol. Phys.* **2020**, *118*, e1737742.

(54) Zhang, J.; Lei, Y.-K.; Zhang, Z.; Chang, J.; Li, M.; Han, X.; Yang, L.; Yang, Y. I.; Gao, Y. Q. A Perspective on Deep Learning for Molecular Modeling and Simulations. *J. Phys. Chem. A* **2020**, *124*, 6745–6763.

(55) Bonati, L.; Trizio, E.; Rizzi, A.; Parrinello, M. A unified framework for machine learning collective variables for enhanced sampling simulations: mlcolvar. *J. Chem. Phys* **2023**, *159*, 014801.

(56) Ketkaew, R.; Luber, S. DeepCV: A Deep Learning Framework for Blind Search of Collective Variables in Expanded Configurational Space. *J. Chem. Inf. Model.* **2022**, *62*, 6352–6364.

(57) Ramil, M.; Boudier, C.; Goryaeva, A. M.; Marinica, M.-C.; Maillet, J.-B. On Sampling Minimum Energy Path. *J. Chem. Theory Comput.* **2022**, *18*, 5864–5875.

(58) Chen, H.; Liu, H.; Feng, H.; Fu, H.; Cai, W.; Shao, X.; Chipot, C. MLCV: Bridging

Machine-Learning-Based Dimensionality Reduction and Free-Energy Calculation. *J. Chem. Inf. Model.* **2022**, *62*, 1–8.

(59) Kleiman, D. E.; Nadeem, H.; Shukla, D. Adaptive Sampling Methods for Molecular Dynamics in the Era of Machine Learning. *J. Phys. Chem. B* **2023**, *127*, 10669–10681.

(60) Fu, H.; Bian, H.; Shao, X.; Cai, W. Collective Variable-Based Enhanced Sampling: From Human Learning to Machine Learning. *J. Phys. Chem. Lett.* **2024**, *15*, 1774–1783.

(61) Eastman, P.; Pande, V. In *GPU Computing Gems Jade Edition*; Hwu, W.-m. W., Ed.; Applications of GPU Computing Series; Morgan Kaufmann: Boston, 2012; pp 399–407.

(62) Fu, H.; Liu, H.; Xing, J.; Zhao, T.; Shao, X.; Cai, W. Deep-Learning-Assisted Enhanced Sampling for Exploring Molecular Conformational Changes. *J. Phys. Chem. B* **2023**, *127*, 9926–9935.

(63) Fiorin, G.; Marinelli, F.; Faraldo-Gómez, J. D. Direct Derivation of Free Energies of Membrane Deformation and Other Solvent Density Variations From Enhanced Sampling Molecular Dynamics. *J. Comput. Chem.* **2020**, *41*, 449–459.

(64) Coronel, L.; Muccio, G. D.; Rothberg, B.; Giacomello, A.; Carnevale, V. Lipid-mediated hydrophobic gating in the BK potassium channel. 2024; https://arxiv.org/abs/2405.04644.

(65) Zhou, W.; Fiorin, G.; Anselmi, C.; Karimi-Varzaneh, H. A.; Poblete, H.; Forrest, L. R.; Faraldo-Gómez, J. D. Large-scale state-dependent membrane remodeling by a transporter protein. *eLife* **2019**, *8*, e50576.

(66) Vant, J. W.; Sarkar, D.; Streitwieser, E.; Fiorin, G.; Skeel, R.; Vermaas, J. V.;

Singharoy, A. Data-guided Multi-Map variables for ensemble refinement of molecular movies. *J. Chem. Phys.* **2020**, *153*.

(67) Fiorin, G.; Forrest, L. R.; Faraldo-Gómez, J. D. Membrane free-energy landscapes derived from atomistic dynamics explain nonuniversal cholesterol-induced stiffening. *PNAS Nexus* **2023**, *2*, pgad269.

(68) Wells, D. B.; Abramkina, V.; Aksimentiev, A. Exploring transmembrane transport through α-hemolysin with grid-steered molecular dynamics. *J. Chem. Phys.* **2007**, *127*, 125101.

(69) Lagardère, L.; Maurin, L.; Adjoua, O.; El Hage, K.; Monmarché, P.; Piquemal, J.-P.; Hénin, J. Lambda-ABF: Simplified, Portable, Accurate, and Cost-Effective Alchemical Free-Energy Computation. *J. Chem. Theory Comput.* **2024**, *20*, 4481–4498.

(70) Tidor, B. Simulated annealing on free energy surfaces by a combined molecular dynamics and Monte Carlo approach. *J. Phys. Chem.* **1993**, *97*, 1069–1073.

(71) Kong, X.; Brooks III, C. L. λ-dynamics: A new approach to free energy calculations. *J. Chem. Phys.* **1996**, *105*, 2414–2423.

(72) Bou-Rabee, N.; Owhadi, H. Long-Run Accuracy of Variational Integrators in the Stochastic Context. *SIAM Journal on Numerical Analysis* **2010**, *48*, 278–297.

(73) Chen, H.; Maia, J. D. C.; Radak, B. K.; Hardy, D. J.; Cai, W.; Chipot, C.; Tajkhorshid, E. Boosting Free-Energy Perturbation Calculations with GPU-Accelerated NAMD. *J. Chem. Inf. Model.* **2020**, *60*, 5301–5307.

(74) Pitera, J. W.; Chodera, J. D. On the Use of Experimental Observations to Bias Simulated Ensembles. *J. Chem. Theory Comput.* **2012**, *8*, 3445–3451.

(75) White, A. D.; Voth, G. A. Efficient and Minimal Method to Bias Molecular Simulations with Experimental Data. *J. Chem. Theory Comput.* **2014**, *10*, 3023–3030.

(76) Roux, B.; Islam, S. M. Restrained-Ensemble Molecular Dynamics Simulations Based on Distance Histograms from Double Electron–Electron Resonance Spectroscopy. *J. Phys. Chem. B* **2013**, *117*, 4733–4739.

(77) Roux, B.; Weare, J. On the statistical equivalence of restrained-ensemble simulations with the maximum entropy method. *J Chem Phys* **2013**, *138*, 084107.

(78) White, A. D.; Dama, J. F.; Voth, G. A. Designing Free Energy Surfaces That Match Experimental Data with Metadynamics. *J. Chem. Theory Comput.* **2015**, *11*, 2451–2460.

(79) Laio, A.; Parrinello, M. Escaping free energy minima. *Proc. Natl. Acad. Sci. U.S.A.* **2002**, *99*, 12562–12565.

(80) Raiteri, P.; Laio, A.; Gervasio, F. L.; Micheletti, C.; Parrinello, M. Efficient Reconstruction of Complex Free Energy Landscapes by Multiple Walkers Metadynamics. *J. Phys. Chem. B* **2006**, *110*, 3533–3539.

(81) Darve, E.; Pohorille, A. Calculating free energies using average force. *J. Chem. Phys.* **2001**, *115*, 9169–9183.

(82) Hénin, J.; Chipot, C. Overcoming free energy barriers using unconstrained molecular dynamics simulations. *J. Chem. Phys.* **2004**, *121*, 2904–2914.

(83) Darve, E.; Rodríguez-Gómez, D.; Pohorille, A. Adaptive biasing force method for scalar and vector free energy calculations. *J. Chem. Phys.* **2008**, *128*, 144120.

(84) Hénin, J.; Fiorin, G.; Chipot, C.; Klein, M. L. Exploring multidimensional free energy landscapes using time-dependent biases on collective variables. *J. Chem. Theory Comput.* **2010**, *6*, 35–47.

(85) Lesage, A.; Lelièvre, T.; Stoltz, G.; Hénin, J. Smoothed Biasing Forces Yield Unbiased

Free Energies with the Extended-System Adaptive Biasing Force Method. *J. Phys. Chem. B* **2017**, *121*, 3676–3685.

(86) Fu, H.; Shao, X.; Chipot, C.; Cai, W. Extended Adaptive Biasing Force algorithm. An on-the-fly implementation for accurate free-energy calculations. *J. Chem. Theory Comput.* **2016**, *12*, 3506–3513.

(87) Fu, H.; Zhang, H.; Chen, H.; Shao, X.; Chipot, C.; Cai, W. Zooming across the free-energy landscape: shaving barriers, and flooding valleys. *J. Phys. Chem. Lett* **2018**, *9*, 4738–4745.

(88) Fu, H.; Shao, X.; Cai, W.; Chipot, C. Taming Rugged Free Energy Landscapes Using an Average Force. *Accounts of Chemical Research* **2019**, *52*, 3254–3264.

(89) Chen, H.; Fu, H.; Chipot, C.; Shao, X.; Cai, W. Overcoming Free-Energy Barriers with a Seamless Combination of a Biasing Force and a Collective Variable-Independent Boost Potential. *J. Chem. Theory Comput.* **2021**, *17*, 3886–3894.

(90) Hénin, J. Fast and Accurate Multidimensional Free Energy Integration. *J. Chem. Theory Comput.* **2021**, *17*, 6789–6798, PMID: 34665624.

(91) Minoukadeh, K.; Chipot, C.; Lelièvre, T. Potential of Mean Force Calculations: A Multiple-Walker Adaptive Biasing Force Approach. *J. Chem. Theory Comput.* **2010**, *6*, 1008–1017.

(92) Comer, J.; Gumbart, J. C.; Hénin, J.; Lelièvre, T.; Pohorille, A.; Chipot, C. The adaptive biasing force method: everything you always wanted to know but were afraid to ask. *J. Phys. Chem. B* **2015**, *119*, 1129–1151.

(93) Barducci, A.; Bussi, G.; Parrinello, M. Well-tempered metadynamics: a smoothly converging and tunable free-energy method. *Phys. Rev. Lett.* **2008**, *100*, 020603.

(94) Crespo, Y.; Marinelli, F.; Pietrucci, F.; Laio, A. Metadynamics convergence law in a multidimensional system. *Phys. Rev. E* **2010**, *81*, 055701.

(95) McGovern, M.; de Pablo, J. A boundary correction algorithm for metadynamics in multiple dimensions. *J. Chem. Phys.* **2013**, *139*, 084102.

(96) Baftizadeh, F.; Cossio, P.; Pietrucci, F.; Laio, A. Protein Folding and Ligand-Enzyme Binding from Bias-Exchange Metadynamics Simulations. *Current Physical Chemistry* **2012**, *2*, 79–91.

(97) Bussi, G.; Laio, A.; Parrinello, M. Equilibrium Free Energies from Nonequilibrium Metadynamics. *Phys. Rev. Lett.* **2006**, *96*, 090601.

(98) Marinelli, F.; Pietrucci, F.; Laio, A.; Piana, S. A Kinetic Model of Trp-Cage Folding from Multiple Biased Molecular Dynamics Simulations. *PLOS Computational Biology* **2009**, *5*, 1–18.

(99) Biarnés, X.; Pietrucci, F.; Marinelli, F.; Laio, A. METAGUI. A VMD interface for analyzing metadynamics and molecular dynamics simulations. *Comput. Phys. Commun.* **2012**, *183*, 203–211.

(100) Giorgino, T.; Laio, A.; Rodriguez, A. METAGUI 3: A graphical user interface for choosing the collective variables in molecular dynamics simulations. *Comput. Phys. Commun.* **2017**, *217*, 204–209.

(101) Marinelli, F.; Faraldo-Gómez, J. D. Force-Correction Analysis Method for Derivation of Multidimensional Free-Energy Landscapes from Adaptively Biased Replica Simulations. *J. Chem. Theory Comput.* **2021**, *17*, 6775–6788.

(102) Invernizzi, M.; Parrinello, M. Rethinking Metadynamics: From Bias Potentials to Probability Distributions. *The Journal of Physical Chemistry Letters* **2020**, *11*, 2731–2736.

(103) Fukunishi, H.; Watanabe, O.; Takada, S. On the Hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction. *J. Chem. Phys.* **2002**, *116*, 9058–9067.

(104) Piana, S.; Laio, A. A bias-exchange approach to protein folding. *J. Phys. Chem. B* **2007**, *111*, 4553–4559.

(105) Jiang, W.; Phillips, J. C.; Huang, L.; Fajer, M.; Meng, Y.; Gumbart, J. C.; Luo, Y.; Schulten, K.; Roux, B. Generalized scalable multiple copy algorithms for molecular dynamics simulations in NAMD. *Comput. Phys. Commun.* **2014**, *185*, 908–916.

(106) Marchi, M.; Ballone, P. Adiabatic bias molecular dynamics: A method to navigate the conformational space of complex molecular systems. *J. Chem. Phys.* **1999**, *110*, 3697–3702.

(107) Paci, E.; Karplus, M. Forced unfolding of fibronectin type 3 modules: an analysis by biased molecular dynamics simulations. *J. Mol. Biol.* **1999**, *288*, 441–459.

(108) A Beccara, S.; Škrbić, T.; Covino, R.; Faccioli, P. Dominant folding pathways of a WW domain. *Proceedings of the National Academy of Sciences* **2012**, *109*, 2330–2335.

(109) Gagelin, A.; Largeau, C.; Masscheleyn, S.; Piel, M. S.; Calderón-Mora, D.; Bouillaud, F.; Hénin, J.; Miroux, B. Molecular determinants of inhibition of UCP1-mediated respiratory uncoupling. *Nature Communications* **2023**, *14*, 2594.

(110) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1-2*, 19–25.

(111) Abraham, M.; Alekseenko, A.; Basov, V.; Bergh, C.; Briand, E.; Brown, A.; Doijade, M.; Fiorin, G.; Fleischmann, S.; Gorelov, S. et al. GROMACS 2024.1 Manual. 2024; https://doi.org/10.5281/zenodo.10090355, Accessed on 04-10-2024.

(112) Monticelli, L.; Kandasamy, S. K.; Periole, X.; Larson, R. G.; Tieleman, D. P.; Marrink, S.-J. The MARTINI coarse-grained force field: extension to proteins. *J. Chem. Theory Comput.* **2008**, *4*, 819–834.

(113) Adjoua, O.; Lagardère, L.; Jolly, L.-H.; Durocher, A.; Very, T.; Dupays, I.; Wang, Z.; Inizan, T. J.; Célerse, F.; Ren, P. et al. Tinker-HP: Accelerating molecular dynamics simulations of large complex systems with advanced point dipole polarizable force fields using GPUs and multi-GPU systems. *J. Chem. Theory Comput.* **2021**, *17*, 2034–2053.

(114) Cardenas, A. E.; Hunter, A.; Wang, H.; Elber, R. ScMiles2: A script to conduct and analyze Milestoning trajectories for long time dynamics. *J. Chem. Theory Comput.* **2022**, *18*, 6952–6965.