

Molecular Physics

An International Journal at the Interface Between Chemistry and Physics

ISSN: 0026-8976 (Print) 1362-3028 (Online) Journal homepage: <https://www.tandfonline.com/loi/tmph20>

Using collective variables to drive molecular dynamics simulations

Giacomo Fiorin, Michael L. Klein & Jérôme Hénin

To cite this article: Giacomo Fiorin, Michael L. Klein & Jérôme Hénin (2013) Using collective variables to drive molecular dynamics simulations, Molecular Physics, 111:22-23, 3345-3362, DOI: [10.1080/00268976.2013.813594](https://doi.org/10.1080/00268976.2013.813594)

To link to this article: <https://doi.org/10.1080/00268976.2013.813594>



© 2013 The Author(s). Published by Taylor & Francis.



Accepted author version posted online: 14 Jun 2013.
Published online: 16 Jul 2013.



Submit your article to this journal [↗](#)



Article views: 15336



View related articles [↗](#)



Citing articles: 241 View citing articles [↗](#)

INVITED ARTICLE

Using collective variables to drive molecular dynamics simulations

Giacomo Fiorin^{a,*}, Michael L. Klein^a and Jérôme Hénin^b^aDepartment of Chemistry and Institute for Computational Molecular Science, Temple University, Philadelphia, PA, USA; ^bLaboratoire de Biochimie Théorique, Institut de Biologie Physico-Chimique, CNRS, Paris, France

(Received 24 April 2013; final version received 4 June 2013)

A software framework is introduced that facilitates the application of biasing algorithms to collective variables of the type commonly employed to drive massively parallel molecular dynamics (MD) simulations. The modular framework that is presented enables one to combine existing collective variables into new ones, and combine any chosen collective variable with available biasing methods. The latter include the classic time-dependent biases referred to as steered MD and targeted MD, the temperature-accelerated MD algorithm, as well as the adaptive free-energy biases called metadynamics and adaptive biasing force. The present modular software is extensible, and portable between commonly used MD simulation engines.

Keywords: molecular dynamics simulation; collective variable; free-energy calculation; adaptive bias; sampling

Introduction

The potential energy functions used in molecular dynamics (MD) simulations are often augmented with additional biasing terms, either to overcome limitations in the physical model itself or to obtain higher statistical sampling. Approaches to this goal are often founded on the selection of certain collective variables (in short, ‘colvars’) [1–9] to describe macroscopic phenomena. Few colvars can be sampled extensively to calculate statistical quantities accurately, unlike the far more numerous atomic positions. Furthermore, reduced representations based on colvars easily lend themselves to comparison with experiments, or with empirical models constructed at lower levels of detail. Depending on a particular problem, the simplicity of this comparison can be of great advantage over many other efficient approaches that bias ‘microscopically’ the exploration of the phase space [10–17].

However, implementation of even the simpler methods based on colvars is often slowed down by the need to re-implement all of the most commonly used variable functions as well. While removing this bottleneck is mostly a software design challenge, new methods are more easily developed when previous software can be reused. Computational scientists also benefit from simultaneously gaining access to a wide set of methods, all sharing a consistent formalism. To this end, we wrote a software addition for MD simulation programs, called the ‘collective variables module’ (hereafter colvars module), wherein variables are defined as an expansion over a set of commonly used functional forms. First released as an integral component of the NAMD program [18,19], the colvars module has been used

in a variety of applications and recently been interfaced with the LAMMPS program [20]. One separate, notable effort to implement several types of colvar biases is the PLUMED package [21]. Our approach differs from that package by including thermodynamic force measurements and the adaptive biasing force (ABF) family of algorithms, as well as removing the need for artificial restraints in many occasions by using a transparent formalism for treating moving frames of reference. Conversely, our implementation does not include exchange-based variants of the metadynamics method, or other methods which require *a posteriori* calculations to remove the effects of biasing potentials before analysing statistical distributions.

The colvars module includes several commonly used sampling algorithms, in combination with a wide set of functions to be used as variables. A unique advantage is the concurrent availability of methods based on the probability density [1,3,4,7] alongside methods based on average forces [5,22–24]. In a previous manuscript [25], we used this versatility to directly compare a novel multidimensional implementation of the ABF method [5] with the metadynamics method [7]. New implementations of other algorithms have also been added by other researchers: the well-tempered correction of metadynamics [26], the replica-exchange umbrella sampling [10,11] and the ‘swarms of trajectories’ methods [8,27]. The computational overhead of this implementation is negligible in most typical applications, thanks to a design that matches the requirements of modern computing resources. Algorithms that use more than one replica of the system but do not require strict synchronisation can also take advantage of a specialised communication

*Corresponding author. Email: giacomo.fiorin@temple.edu

algorithm that preserves near-ideal efficiency in massively parallel runs.

One of the innovations of our formalism is a generic and robust treatment of rotational degrees of freedom. Typical solutions to define variables on rotating frames such as flexible macromolecules require applying artificial restraints that prevent rotation, but also perturb the internal dynamics of the system. Using an approach based on quaternions [28], we generalise the least-squares superposition approach used for root-mean-square deviations (RMSDs) to any type of colvar that is not intrinsically invariant under rotations. This choice allows for a straightforward comparison with simulation studies that rely on a rigid-body assumption [29–34]. We further extend the quaternion-based approach to define several novel colvars, to model the orientation of macromolecules and flexible structures.

Our transferable software, founded on a relatively simple formalism, can be used to leverage the most advanced computational resources and to attack increasingly complex problems. We here demonstrate several applications through usage cases, which enable the reader with the ability to tackle advanced applications or to develop new methods as well.

1. Defining collective variables and their derived physical quantities

The most general definition of a colvar ξ is as a differentiable function of the vector of $3N$ atomic Cartesian coordinates, \mathbf{X} :

$$\xi(\mathbf{X}) = \xi(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N). \quad (1)$$

Depending on the structure of the system, $\xi(\mathbf{X})$ is often a function of much fewer arguments than $3N$, or can be expressed as a combination of such functions:

$$\xi(\mathbf{X}) = \xi(z^{(1)}(\mathbf{X}), z^{(2)}(\mathbf{X}), \dots, z^{(\alpha)}(\mathbf{X}) \dots) \quad (2)$$

with the number of basis functions $z^{(\alpha)}$ much smaller than the number of atoms. We refer to $z^{(\alpha)}(\mathbf{X})$ as a *colvar component*: in the simpler and most common scenario, a single component z is identified with the colvar ξ . Historically, these elementary functions have been distances, angles or dihedrals, effectively mimicking the interaction terms of many empirical Hamiltonians [35–41]. More recently MD simulations are applied to complex macromolecules; therefore, components are used that mimic many-body interactions or track collective motions, such as moments of inertia, principal components from covariance analysis [42], etc. By implementing the calculation of derivatives for the most commonly used components $z^{(\alpha)}(\mathbf{X})$, as well as combination rules following Equation (2) (in our current software implementation, a polynomial expansion), algorithms

are automatically generalised to a virtually unlimited set of variable functions. Such combinations may be used, for example, after an initial mapping of the colvar space has been performed by any method: for an efficient calculation of a potential of mean force (PMF), colvars known to be relevant may be combined to obtain a single ‘reaction coordinate’.

The Cartesian gradient of the colvar, $\nabla_{\mathbf{X}}\xi(\mathbf{X}) = (\nabla_{\mathbf{x}_1}\xi(\mathbf{X}), \nabla_{\mathbf{x}_2}\xi(\mathbf{X}), \dots)$, is straightforwardly written as a function of those of the components using the chain rule. In most applications the gradient is needed to apply biasing forces to the colvars, and, when applicable, to compute a colvar’s velocity $d\xi/dt$.

Another relevant physical quantity is the so-called ‘inverse gradient’:

$$\frac{\partial \mathbf{X}}{\partial \xi} = \left(\frac{\partial \mathbf{x}_1}{\partial \xi}, \frac{\partial \mathbf{x}_2}{\partial \xi}, \dots, \frac{\partial \mathbf{x}_N}{\partial \xi} \right) \quad (3)$$

which is needed in thermodynamic integration (TI) [22] to evaluate the thermodynamic force exerted on the colvars by the entire system [2]. Unlike the gradient, multiple definitions of the inverse gradient are valid, depending on the choice of a differentiable map between the $3N$ Cartesian coordinates and a set of $3N$ variables that includes the chosen colvars ξ_i . In practice, defining explicitly this map is not necessary, and the inverse gradients may be defined as any set of vector functions that satisfy the following orthonormality condition for any two colvars ξ_i and ξ_j [23,24]:

$$\frac{\partial \mathbf{X}}{\partial \xi_i} \cdot (\nabla_{\mathbf{X}} \xi_j) = \delta_{ij}. \quad (4)$$

When the inverse gradients are integrated over an ensemble of states, a Jacobian derivative should be introduced to preserve the density of phase space as a function of ξ :

$$\frac{\partial \ln |J(\mathbf{X})|}{\partial \xi} = \sum_i \nabla_{\mathbf{x}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial \xi}. \quad (5)$$

For a ‘statistical’ definition of the PMF $F(\xi)$ under the TI formalism [2], the Jacobian term must be explicitly included in the estimate of the thermodynamic gradient $\partial F(\xi)/\partial \xi$. However, a more traditional definition of the PMF describes only actual interaction terms, excluding this geometric entropy contribution. That definition originates in the statistical theory of simple liquids, and is based on the normalised radial distribution function $g(r)$:

$$F(r) = -k_B T \ln(g(r)), \quad (6)$$

where $g(r)$ is obtained dividing by $4\pi r^2$ the probability density of r . This ensures that the interparticle PMF, like $g(r)$, decays to a plateau at large separations: the value of the plateau is usually set at zero. This convention is

intuitive in the case of interparticle distances, but loses its simple interpretation when applied to geometric quantities that are incommensurable with the distance between two molecular species. In the colvars module, we consider any PMF as following the ‘statistical’ convention, i.e. including the Jacobian term [25]: for distances, this term is equal to $-2k_B T \ln(r)$. This provides a more consistent treatment among colvars of different types.

Each colvar may be harmonically coupled to an extended degree of freedom, which is set to undergo Hamiltonian or Langevin dynamics. The extended coordinate acts as a numerically convenient proxy for the actual variable. For appropriate values of the extended system temperature and friction coefficient, this implements the temperature-accelerated MD (TAMD) paradigm [8], which has been proposed as a way to sample conformational changes in proteins [43]. While TAMD does not yield free-energy estimates, it treats all biased degrees of freedom independently, making it applicable to high-dimensional problems – which it effectively treats as separate, one-dimensional problems.

In the remainder of this section, we discuss the properties of different types of colvars. Because the most frequent scenario involves colvars $\xi(\mathbf{X})$ with only one component $z(\mathbf{X})$ each, we will identify z with ξ for the remainder of this section, unless otherwise noted. The remainder of our discussion also applies to the most general scenario of colvars constructed from combining many different components (Equation (2)). Physically meaningful colvars should be invariant under global translations and rotations of the model system; however, certain functions are only invariant when combined with a change of coordinates to a moving frame of reference. We group the colvar components into six classes (Classes I–VI) based on their rotational and translational invariance, and on other more specific criteria. With the only exception of Class IV, all components feature a computational cost $\mathcal{O}(N)$.

Class I components describe the relative arrangement of few geometric centres (individual atoms or groups of atoms): these include the traditionally used distances, angles and dihedrals. Class II components describe the distribution of atomic positions around their centre-of-geometry or centre-of-mass, via physically measurable quantities such as the radius of gyration or the moment of inertia. Class III components are defined based on the connectivity within a macromolecule (for example, a protein or nucleic acid): members of this class include a score function for the secondary structure in protein segments, and projections of a set of dihedral angles on to *dihedral principal components* [44]. Class IV components are based on distances between all pairs of atoms within two groups; therefore, variables constructed upon components of this class are the only ones with $\mathcal{O}(N^2)$ computational cost. Class V components describe the quasi-rigid orientation of a group of atoms, computed by least-squares fitting against a set of reference coordinates: the full orientation is

described by a unit quaternion, from which other variables are extracted to isolate individual subrotations; several of such variables are a novel feature of our software. Class VI components use the same approach as Class V components to express the positions of a set of atoms in a rotated frame: within that frame, the RMSD from the reference positions (*rmsd* component) and the projection on a $3N$ -dimensional displacement vector (*eigenvector*) can be calculated. Of the six classes, only the last two rely on a least-squares superposition to remove the effect of roto-translations. However, our software makes the superposition procedure available transparently to all classes, to allow for its reuse in the development of new types of variables.

Class I colvar components: functions of few centres-of-geometry

Many colvars are defined as simple quantities computed between few positions. These are, for example, the distance between two positions, the angle between three positions and the dihedral (torsional angle) between four positions. The components that define these and other variables of similar type are listed in the appendix (Table A1). Each position may be defined as the position of a single atom, or the centre-of-mass of a group of atoms: under periodic boundary conditions, the centre-of-mass is calculated assuming that all atomic positions are at their closest periodic images from the centre-of-mass itself. However, all distances between centres-of-mass are computed by the minimal image convention. Optionally, the latter behaviour can be disabled, for example, if the atoms are distributed over a distance larger than half of the unit cell in any direction.

The component *dihedral* and, under periodic boundary conditions, the component *distanceZ* have values within a periodic interval. When a colvar ξ is defined based on a periodic component, the software uses the minimal image convention transparently when performing sums or differences between values of ξ .

Class II colvar components: internal structure of a group of atoms

Class II colvar components, listed in the appendix (Table A2), represent simple physical quantities associated with a set of atoms taken as a whole. The radius of gyration (*gyration* component) can be computed in two alternative ways: one assumes equal weights for all the atoms involved or one uses the atomic masses to weigh the sum. The *inertia* component measures the moment of inertia of the group of atoms, or the trace of the inertia tensor. The *inertiaZ* component instead measures the diagonal element of the inertia tensor along a user-defined axis.

Class III colvar components: functions on many atomic positions

The two components in this class, listed in the appendix (Table A3), are designed to model the structure of a long polymer, for example, a protein. The component *alpha* measures the content of the α -helix structure in a certain structure, using a scoring function for the angles between C_α atoms and another scoring function for α -helical hydrogen bonds. The angle term is computed via a scoring function similar in concept to the potential used by Alemani *et al.* [45]: the angle ω between three consecutive C_α atoms is used to evaluate a scoring function with respect to a reference value ω_0 , which is by default 88° . The hydrogen-bond term measures the number of α -helical hydrogen bonds formed between backbone groups four residues apart in the sequence. The smooth scoring function is similar to that previously used to measure atomic coordination numbers [46]. The value of the *alpha* component is then

$$\begin{aligned} z(\mathbf{x}_{C_\alpha}^{N_0}, \mathbf{x}_O^{N_0}, \mathbf{x}_{C_\alpha}^{N_0+1}, \mathbf{x}_O^{N_0+1}, \dots, \mathbf{x}_N^{N_0+5}, \mathbf{x}_{C_\alpha}^{N_0+5}, \dots) \\ = \frac{C_{\text{ang}}}{(N-2)} \sum_{n=N_0}^{N_0+N-2} \frac{1 - (\omega_\alpha^{n \rightarrow n+2} - \omega^{\text{ref}})^2 / (\Delta\omega^{\text{tol}})^2}{1 - (\omega_\alpha^{n \rightarrow n+2} - \omega^{\text{ref}})^4 / (\Delta\omega^{\text{tol}})^4} \\ + \frac{C_{\text{HB}}}{(N-4)} \sum_{n=N_0}^{N_0+N-4} \frac{1 - (d_{O-N}^{n \rightarrow n+4} / d^{\text{ref}})^6}{1 - (d_{O-N}^{n \rightarrow n+4} / d^{\text{ref}})^8}, \end{aligned} \quad (7)$$

where \mathbf{x}_A^n is the position of atom A within the amino acid with sequence number N , $\omega_\alpha^{n \rightarrow n+2}$ is the angle formed by the position of the three consecutive C_α atoms of the amino acids n , $n+1$ and $n+2$, $d_{O-N}^{n \rightarrow n+4}$ is the distance between the backbone oxygen of the amino acid n and the backbone nitrogen of the amino acid $n+4$, and C_{ang} and C_{HB} are two parameters whose sum is 1 (by default, both 0.5).

The component *dihedralPC* can be used to model arbitrary secondary structures of proteins, or small tertiary structures. Its value represents the projection of backbone dihedral angles within a protein segment on to a ‘dihedral principal component’, which is the projection of the protein’s backbone structure on to a predefined dihedral ‘eigenvector’. Such an eigenvector may be obtained following the formalism of dihedral principal component analysis (dPCA) proposed by Mu *et al.* [47] and documented in detail by Altis *et al.* [48]. Given a peptide or protein segment of N residues, each with Ramachandran angles ϕ_i and ψ_i , dPCA consists of a variance/covariance analysis of the $4(N-1)$ variables $\cos(\psi_1)$, $\sin(\psi_1)$, $\cos(\phi_2)$, $\sin(\phi_2)$, ..., $\cos(\phi_N)$, $\sin(\phi_N)$. For a dihedral eigenvector of coefficients $(k_i)_{1 \leq i \leq 4(N-1)}$, the projection of the current backbone conformation on to such an eigenv-

ector is

$$z(\mathbf{X}) = \sum_{n=1}^{N-1} (k_{4n-3} \cos(\psi_n) + k_{4n-2} \sin(\psi_n) + k_{4n-1} \cos(\phi_{n+1}) + k_{4n} \sin(\phi_{n+1})). \quad (8)$$

Class IV colvar components: based on multiple pairwise atomic distances

Class IV colvar components, listed in the appendix (Table A4), are defined as non-linear combinations of many distances between pairs of atoms, and allow to model more complex structural changes within a group of atoms, without assuming that they may be linked by covalent bonds. The first of such components is *distanceInv*, which represents a generalised ‘mean distance’ between two groups of atoms:

$$d_{A,B}^{[n]} = \left(\frac{1}{N_A N_B} \sum_{\text{atom } i \text{ group A}} \sum_{\text{atom } j \text{ group B}} \left(\frac{1}{\|\mathbf{d}^{ij}\|} \right)^n \right)^{-1/n}, \quad (9)$$

where $\|\mathbf{d}^{ij}\|$ is the distance between atoms i and j , and n is an even integer. This definition yields an average distance between two groups that behaves as $1/d^n$.

Usage Case 1: nuclear Overhauser effect (NOE) restraints. A major application of *distanceInv* is to enforce experimental distance restraints based on NOEs measured in NMR experiments. The appropriate functional form is obtained by setting the exponent n to 6 (default value). This has been used to optimise the bound structure of a novel inhibitor with a viral proton channel [49].

Another important variable to describe a set of interacting atoms is the total number of contacts, or the coordination number (*coordNum*) between two groups within a chosen cutoff distance d^0 . In order to be used as a variable, the coordination number must be approximated by a differentiable function, such as [46]:

$$CN_{A,B} = \sum_{\text{atom } i \text{ group A}} \sum_{\text{atom } j \text{ group B}} \frac{1 - (\|\mathbf{d}^{ij}\|/d^0)^n}{1 - (\|\mathbf{d}^{ij}\|/d^0)^m}, \quad (10)$$

where A and B are the two groups, which share no atoms, $\|\mathbf{d}^{ij}\|$ is the distance between atoms i and j , and n and m are even integers. A variant of this component, *selfCoordNum*, is used in the case where the two groups are identical.

Usage Case 2: aggregation of solute molecules. Restraints on a *selfCoordNum* component may be used to prevent, or to enforce, the aggregation of a set of solute molecules or ions. One application is the computational insertion of many hydrophobic molecules in an aqueous solution, from where they are expected to partition into different environments such as protein pockets or a lipid bilayer, as was done to study binding of a general anaesthetic to a trans-membrane receptor channel [50]. A bias to prevent aggregation is useful in the transient regime where the local concentration in water is very high, to prevent formation of long-lived clusters that could slow down partitioning. This restraint may be lifted once partitioning has taken place and the tendency of the solute to self-aggregate in water is reduced.

Class V colvar components: pseudo-rigid orientations of a group of atoms

The components in this class, listed in the appendix (Table A5), are all derived from the optimal rotation between a set of reference coordinates \mathbf{X}^0 and the set of coordinates \mathbf{X} . The algorithms to calculate such optimal rotation are described in Section 3.1. The value of the component *orientation* is the unit quaternion \mathbf{q} (Equation (18)). Its four real components (q_0, q_1, q_2, q_3) are linked by the constraint that their sum of squares is equal to 1: this constraint is automatically enforced by all the implemented biasing methods. Additionally, because a quaternion \mathbf{q} and its opposite $-\mathbf{q}$ describe identical orientations, distances between two quaternion values are computed by the minimal image convention. Likewise, densities of states are measured accounting for the same symmetry.

Usage Case 3: orientational restraint for a flexible molecule. When one wishes to restrain the rotation of a flexible molecule, defining its orientation is not trivial. In such a case, defining and restraining an *orientation* variable has the following two advantages: (i) only the global rotation is restrained, without applying any force to the molecule's internal degrees of freedom; and (ii) this component is able to measure the overall molecular orientation with a high tolerance for conformational changes, which may cause the failure of other approaches. For example, it is difficult to define the principal axes of the inertia tensor of a nearly-spherical molecule, but the *orientation* component yields a robust value also in that case.

In many applications, the three effective degrees of freedom of one *orientation* component can be sampled efficiently. However, in cases where a strong anisotropy is present in the rotational motions, separating one of the

degrees of freedom from the other two may be an efficient strategy. The simplest choice for such one-dimensional variable is the amplitude of the rotation measured by the angle θ (*orientationAngle* component). Because of divergent gradients $\nabla_{\mathbf{x}}\theta$ near $\theta = 0$, the component *orientationAngle* is best used only during analysis, while the *orientationProj* component defined as $\cos(\theta)$ may be used instead to apply forces.

When a particular axis of interest exists, indicated by the unit vector \mathbf{e} , it can be used to decompose one complete rotation in two subrotations [52,53]: these are the *spin* rotation around \mathbf{e} and the *tilt* rotation, i.e. rotation 'away from the direction \mathbf{e} '. Of the three degrees of freedom describing the full rotation, only two are represented by the tilt and spin angles: the third is the orientation within the plane orthogonal to \mathbf{e} of the axis generating the tilt subrotation (Figure 1). This decomposition has the advantage that the two angles of subrotation ϕ (spin) and ω (tilt) have the same values, regardless of which subrotation is applied first. Because the axis of the 'tilt' subrotation is not resolved with its sign, only the absolute value of the tilt angle, $|\omega|$, is determined; therefore, its atomic gradients $\nabla_{\mathbf{x}}|\omega|$ are finite but discontinuous at $|\omega| = 0^\circ$ and $|\omega| = 180^\circ$, where ω changes sign. For this reason, we defined the *tilt* component as $t = \cos(\omega)$ to be used as a variable whose derivatives are continuous everywhere.

The component *spinAngle* calculates the value of the spin angle ϕ within the interval $[-180^\circ; 180^\circ]$. However, a 'gimbal lock' condition may arise if $\omega = 180^\circ$ and if the axis of the complete rotation (composition of tilt and spin) is orthogonal to \mathbf{e} [see the expression for $\nabla_{\mathbf{x}}\phi$ in Table A5 when $q_0^2 = 0$ and $(\mathbf{q} \cdot \mathbf{e})^2 = 0$]. In applications where it is possible that both conditions $q_0^2 = 0$ and $(\mathbf{q} \cdot \mathbf{e})^2 = 0$ occur simultaneously, rather than *spinAngle* we recommend to use the complete quaternion provided by the *orientation* component, and to only extract subrotation angles during subsequent analysis of the results.

Class VI colvar components: deviations from a reference set of coordinates

Class VI components, listed in the appendix (Table A6), measure changes in the internal structure of a group of atoms, with respect to a set of reference coordinates. As for Class V components, an optimal roto-translation is calculated between the reference positions and the positions during the simulation; however, this roto-translation is only used to remove the effects of rigid-body motions such as molecular tumbling.

The first component within Class VI is the widely used minimum RMSD between the reference and the current positions. The second component (*eigenvector*) represents the projection of the atomic displacements from a set of reference positions on to a $3N$ -dimensional vector. The $3N$ -dimensional vector is typically an eigenvector of

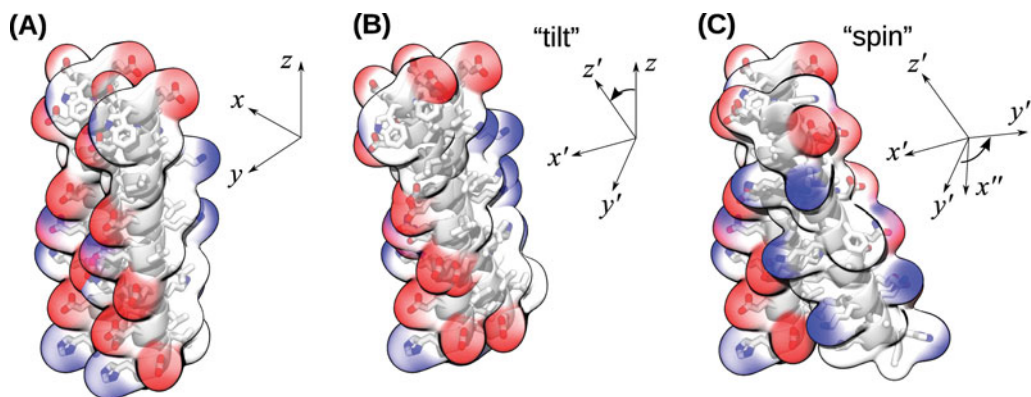


Figure 1. Schematic representation of the *tilt* and *spinAngle* variables. Two identical peptides known to form a ‘leucine zipper’ structure [51] are shown as cartoons in three successive configurations. (A) The two peptides are fully parallel. (B) The second peptide is rotated by 20° around an axis orthogonal to z (‘tilt’ rotation): the Cartesian axes in the rotated frame attached to the second peptide are indicated by x' , y' and z' . This operation is similar to the nutation around the line of nodes (angle θ) in the Euler angles formalism, but in this case no prior precession rotation is applied (angle ϕ). (C) A second rotation by 180° around z' (‘spin’ rotation) brings x' and y' into x'' and y'' , respectively. This second rotation effectively combines the effect of the precession (ϕ) and intrinsic (ψ) rotations in the Euler angles formalism. See Usage Case 7 for details on how to use the *spinAngle* variable to model interactions between membrane proteins.

the Hessian matrix from a previous normal mode analysis calculation, or of the covariance matrix from an essential dynamics calculation [42]. This component may also be used as a first-order approximation of any coordinate, if the $3N$ -dimensional vector is computed between two sets of reference positions.

2. Applying collective variable based algorithms

Once colvars have been defined, several algorithms can be used to analyse their properties, bias their dynamics towards an empirical ensemble, or enhance the rate of sampling of the phase space along them. In our software implementation, all algorithms are multidimensional, in the sense that each can be applied on a set of one or more colvars. Conversely, each colvar can be accessed or biased by multiple algorithms at the same time.

Different algorithms make use of different quantities: analysis methods only use the value of a colvar ξ or of its velocity $d\xi/dt$. Biasing algorithms (harmonic restraints that implement steered MD and umbrella sampling, metadynamics) also require the calculation of the set of atomic gradients $\partial\xi/\partial\mathbf{X}$. The free-energy calculation methods based on the TI formalism, like ABF, require also the calculation of the inverse gradient field $\partial\mathbf{X}/\partial\xi$ and of the Jacobian term $\partial\ln|J|/\partial z$.

2.1. Runtime analysis

MD simulations now are reaching the point where analysing a long trajectory requires an amount of computer power and storage exceeding the capacity of desktop computers. The tremendous improvements in computing hardware

and software has made this possible. For colvars-related properties, both issues can be circumvented through runtime calculation. The value of a colvar ξ , its velocity $d\xi/dt$, the external biasing force applied to it, and the total force $-\partial V/\partial\xi$ acting on the colvar from the system are all recorded during the simulation, lifting the requirement to save the complete set of positions of the system. Running averages, standard deviations (SDs) and multidimensional histograms of these quantities can also be computed at runtime, and output as part of the colvars trajectory.

Time-correlation functions $C^{(k,l)}(\tau)$ between a chosen pair of variables ξ_k and ξ_l , or their velocities, can also be calculated. Given the total simulation time t_{sim} and a predefined highest correlation time t_{max} , $C^{(k,l)}(\tau)$ is defined as

$$C^{(k,l)}(\tau) = \langle \Pi(\xi_k(t), \xi_l(t + \tau)) \rangle_{0 \leq t \leq (t_{\text{sim}} - t_{\text{max}})}, \quad (11)$$

where $\langle \cdot \rangle_{0 \leq t \leq (t_{\text{sim}} - t_{\text{max}})}$ indicates an average taken on all time frames t ranging from 0 to $t_{\text{sim}} - t_{\text{max}}$, and $\Pi(\xi_k, \xi_l)$ is a scalar product between the variables ξ_k and ξ_l . The product is defined differently depending on their type: for scalars and three-dimensional vectors, the real product and dot product are used. For two unit quaternions \mathbf{q}_k and \mathbf{q}_l , the product $\Pi(\mathbf{q}_k, \mathbf{q}_l)$ is best defined as the cosine of the angle θ_{kl} between the two. This quantity can be shown to be closely related to the standard four-dimensional dot product of the two unit quaternions:

$$\Pi(\mathbf{q}_k, \mathbf{q}_l) = 2(\mathbf{q}_k \cdot \mathbf{q}_l)^2 - 1. \quad (12)$$

For three-dimensional vectors and quaternions, the product $\Pi(\xi_k, \xi_l)$ is the cosine $\cos(\theta_{kl})$, which can also be used to evaluate the second-order Legendre polynomial,

$(3\cos^2(\theta_{kl}) - 1)/2$:

$$C_2^{(k,l)}(\tau) = \frac{1}{2} \langle 3\cos^2(\theta_{kl}) - 1 \rangle_{0 \leq t \leq (t_{\text{sim}} - t_{\text{max}})} . \quad (13)$$

The angular autocorrelation function $C_2^{(k,k)}(\tau)$ can then be used to calculate experimental properties of macromolecules such as NMR relaxation rates [54], or other measures of the magnitude of dipolar interactions.

2.2. Fixed and moving harmonic restraints

Harmonic restraints may be introduced on any vector of colvars, using a single force constant K . To handle inhomogeneities in the values of different colvars, K is rescaled according to a predefined width parameter $\Delta\xi_i$ of each colvar ξ_i , according to

$$V(\xi_1 \dots \xi_n) = \frac{1}{2} K \sum_{i=1}^n \left(\frac{\xi_i - \xi_i^0}{\Delta\xi_i} \right)^2 . \quad (14)$$

This is equivalent to expressing the force constant in units of kcal/mol/ $\Delta\xi_i^2$. Conversely, if $\Delta\xi_i$ is taken to represent the unit value of the colvar, this reverts to the classic expression for the harmonic potential.

Usage Case 4: geometric restraints for alchemical free-energy calculations. In alchemical free-energy calculations of ligand–receptor interactions, excessive motion of the quasi-decoupled ligand leads to large numerical errors, which can be alleviated by positional and orientational restraints. In the case of a flexible ligand, conformational restraints may bring further convergence improvements [55]. A combination of such restraints in the colvars module has been successfully applied to the calculation of the peptide–protein binding free energy [56].

The time evolution of the restraints can be defined in two forms: in the first form, the restraint centres ξ_i^0 evolve linearly over time towards target values; and, in the second form of time-dependent restraints, the force constant K_t is scaled as a function of simulation time according to

$$K_t = K_0 + \lambda_t^\alpha (K_1 - K_0), \quad (15)$$

where K_0 , K_t and K_1 are the initial, current and final values of the force constant, respectively, and λ_t changes linearly from 0 to 1 over a preset amount of simulation time. A value of the exponent α larger than 1 results in a smoother initial behaviour; for a more general definition of the λ schedule, a list of values of λ_t can be supplied as well.

The time evolution of the restraint centres or force constant can be set to occur either continuously (slow growth) or in discrete stages (perturbation). Continuous evolution of the restraint centres implements a generalised, multidimensional form of the steered MD method [57]. Discrete perturbation of the restraint centres may be used in preparing stratified umbrella sampling calculations [1]. Discrete perturbation of the force constant allows for computing free energies of restraints via the free-energy perturbation [58] or TI [22] estimators.

Usage Case 5: targeted MD. Applying a harmonic restraint on an RMSD variable with a moving restraint centre implements the targeted MD method [59]. A flexible molecule may thus be driven at a chosen rate towards any known conformation. The same colvar definition may be applied towards a free-energy-profile calculation, in combination with fixed harmonic restraints (for umbrella sampling), ABF or metadynamics.

2.3. Adaptive biasing force

The version of the ABF method [5,60] implemented in the colvars module is an extension to multidimensional variables of the original NAMD implementation [61]. This implementation was described in detail in a previous report [25]. The limitations of our software correspond to the requirements of unconstrained multidimensional TI [24,25]. These requirements are twofold: analytical second derivatives of each variable in order to calculate a Jacobian term and a form of mutual orthogonality condition that constrains the way multidimensional calculations can be performed.

When an ABF calculation is performed in the presence of other, user-defined potentials and restraints that act *directly* on the variable used for ABF, ensuring that one obtains the intended result requires special precautions. Our software implementation has the advantage that such external restraints are not included in the measured thermodynamic forces, nor in the resulting PMF. For the purpose of comparison with PMFs calculated by histogram-based methods [1,3,4,7], the contribution of the external restraints should be included *a posteriori*.

A more recent variant implemented in our software is the extended-system ABF method (eABF) of Lelièvre *et al.* [62,63], where the adaptive bias is not applied directly to the colvar ξ , but rather to an extended degree of freedom ξ^* , which obeys Hamiltonian dynamics and is harmonically coupled to the actual collective coordinate of interest. The method has also been independently developed by Yang and coworkers [64] within a variant of their orthogonal space random walk approach [65]. While akin to the extended-system variant of metadynamics [46], eABF yields very similar results to standard ABF, while lifting its requirements [24,25], thus being applicable to all types of colvars.

The eABF algorithm also includes external biasing forces in the computed PMFs, making it directly comparable to histogram-based methods even when such external forces are applied. Further characterisation of the eABF method for conformational sampling is in progress, and beyond the scope of this report.

2.4. Metadynamics

The metadynamics method [7] features a time-dependent biasing potential, constructed along the trajectory of a set of user-defined colvars, $\xi = (\xi_1, \dots, \xi_k, \dots, \xi_{N_{cv}})$. The biasing potential is defined as a sum of repulsive Gaussian functions, each created at regular time intervals of length δt :

$$V_{\text{meta}}(\xi(t)) = \sum_{t_0=0, \delta t, 2\delta t, \dots}^{(t-\delta t)} W \prod_{k=1}^{N_{cv}} \exp\left(-\frac{(\xi_k(t) - \xi_k(t_0))^2}{2\delta\xi_k^2}\right), \quad (16)$$

where W is a constant with the dimension of an energy, $\xi_k(t)$ is the value of the i th colvar at the time t , $\xi_k(t_0)$ is its value at the previous time t_0 and $\delta\xi_k$ is the user-defined half-width of the Gaussian along the direction of ξ_k . The method generalises to arbitrary colvars the conformational flooding [4] and local elevation [3] methods, which were formulated specifically to use as colvars the principal components of a covariance matrix or a set of dihedral angles, respectively.

In our implementation, the biasing potential $V_{\text{meta}}(\xi)$ can either be computed analytically at each step, or mapped and accumulated on to a grid where the colvars ξ_k are discretised. A similar grid, holding vector values instead of scalar numbers, is used to represent the potential's gradients $\partial V_{\text{meta}}(\xi)/\partial\xi$. The most convenient choice of the ratio between the width of a Gaussian and the spacing of the grid, $\delta\xi_k/\Delta\xi_k$, is $\sqrt{2\pi}/2 \simeq 1.25$: by this choice, the integral of one Gaussian function ($\prod_k \exp(-(\xi_k - \xi_k(t_0))^2/2\delta\xi_k^2)$) equals the volume of one grid element, ($\prod_k \Delta\xi_k$). At each simulation step, the forces that are applied to the colvars ξ_k are taken from the nearest bin in the grid.

We tested the magnitude of the errors from discretisation using an ensemble of simulations of a tiny system, a K^+ and a Cl^- ion in an implicit solvent, modelled by the generalised Born approach [66]. We defined the colvar ξ as the distance between the two, and ran multiple PMF calculations: for better comparison with realistic applications, we ran all simulations for 10 ns, using Gaussian parameters capable to reconstruct a PMF within that time. A confining harmonic potential $1/2 K_{\text{wall}}(\xi - \xi_{\text{max}})^2$ was applied for $\xi > \xi_{\text{max}} = 8 \text{ \AA}$. In Figure 2, we evaluate the systematic errors of PMFs reconstructed by metadynamics with analytical Gaussians, and by metadynamics with interpolating grids of different spacings. For all the grid spacings tested, discreti-

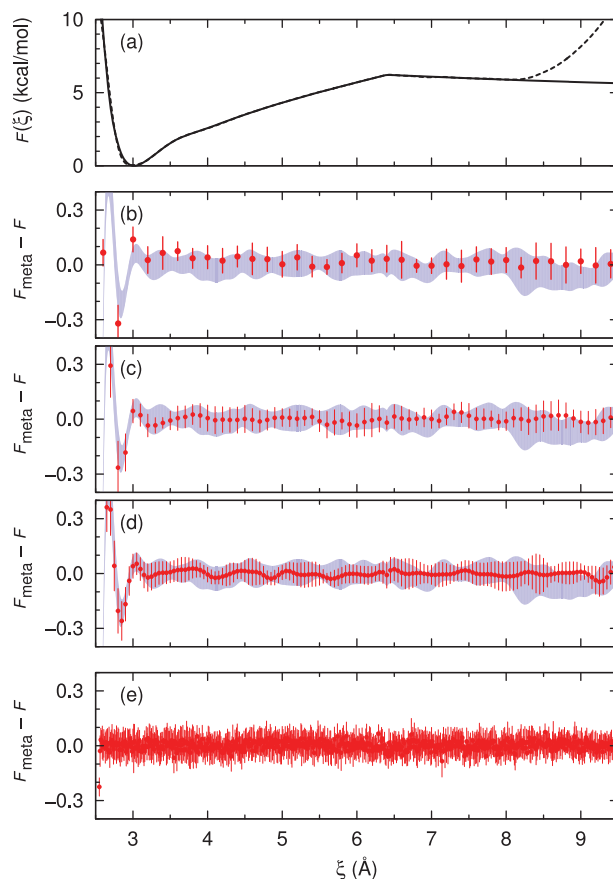


Figure 2. Influence of the grid spacing $\Delta\xi$ on PMF calculations by metadynamics. (a) Shown are the exact PMF $F(\xi)$ (solid line) and the PMF reconstructed by metadynamics $F_{\text{meta}}(\xi)$ (dashed line). The harmonic confining potential for $\xi > 8 \text{ \AA}$ is visible in $F_{\text{meta}}(\xi)$, but was removed in panels (b–e) to calculate differences with $F(\xi)$. (b, c and d) Differences between $F(\xi)$ and $F_{\text{meta}}^{(\text{grid})}(\xi)$, the PMF calculated using Gaussians interpolated on a grid. The red points indicate the mean of $(F_{\text{meta}}^{(\text{grid})}(\xi) - F(\xi))$ and its SD over a set of 10 independent runs, using grids of spacing $\Delta\xi = 0.2 \text{ \AA}$ (b), 0.1 \AA (c) and 0.05 \AA (d), respectively. The light blue filled curves indicate the mean ± 1 SD of the PMF calculated by analytical Gaussians ($F_{\text{meta}}(\xi) - F(\xi)$) over a set of 10 independent runs. The width of the Gaussian functions was $2\delta\xi = 0.25 \text{ \AA}$ both for analytical and for interpolated Gaussians. (e) Mean and SD of $(F_{\text{meta}}^{(\text{grid})}(\xi) - F(\xi))$ in a set of 10 runs with Gaussian width $2\delta\xi = 0.01 \text{ \AA}$ and grid spacing $\Delta\xi = 0.0125 \text{ \AA}$.

sation errors are equal to or smaller than the sampling error arising from the finite height of the Gaussian function itself. Furthermore, the ‘low-pass’ systematic errors due to the finite Gaussian width $2\delta\xi$ in regions of high curvature occur both when the Gaussian functions are interpolated by a grid and when they are analytically differentiated (Figure 2c–2d, at $\xi < 3 \text{ \AA}$). To eliminate such errors, the only possibility is to decrease the Gaussian width $2\delta\xi$ (Figure 2e) and the grid spacing $\Delta\xi$ in proportion. To handle applications where higher curvature than expected appears in a PMF calculation, the code offers the possibility to adjust the

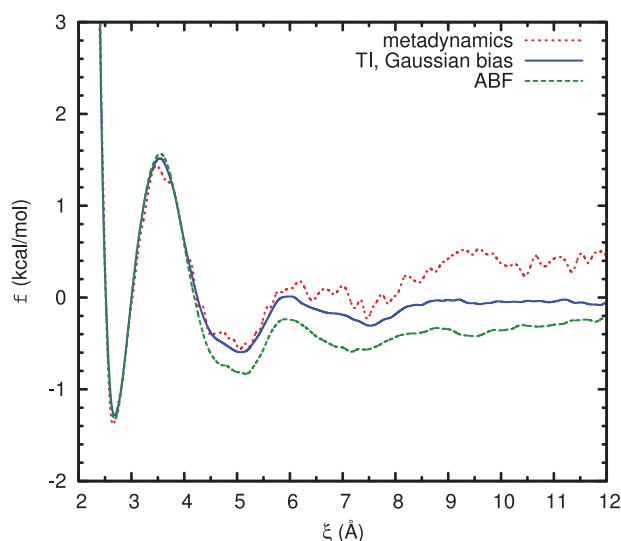


Figure 3. Calculation of a PMF for an $\text{Na}^+ - \text{Cl}^-$ pair in an explicit solvent [35] with a 10 ns MD simulation with a metadynamics biasing force. The PMF reconstructed by metadynamics itself (red) is compared to the one accumulated by a concurrent ABF calculation with its biasing force disabled (blue). Both are compared to the one calculated by a 10 ns ABF calculation (green). All PMFs are plotted following the ‘statistical’ definition, with the Jacobian term $-2k_B T \ln(r)$ subtracted.

simulation parameters while preserving the accumulated information. The same flexibility applies to the boundaries of the grid, ξ_{\min} and ξ_{\max} , along the direction of each colvar ξ : these can be automatically changed by the software to accommodate a wider grid as the simulation progresses, to maintain a minimal distance of ξ from its boundaries of $6\delta_\xi$.

Usage Case 6: calculation of a PMF with a mixed approach.

While using the biasing potential of one algorithm, it is possible to calculate PMFs using the free-energy estimator from another algorithm. Figure 3 shows the PMF calculated using an adaptive Gaussian bias [3,4,7] with a free-energy estimator based on TI [5,22–24,61]. This is implemented by defining a metadynamics potential concurrently with an ABF calculation whose biasing force is disabled. Some advantages of both methods are thus combined, such as accelerated diffusion over flat regions of the PMF, together with an alternative, asymptotically unbiased free-energy estimator. This demonstrates how the present framework may be used to develop and test new methods as combinations of existing algorithms.

3. Design and implementation

Our software implementation can function within any MD or molecular modelling program, augmenting it with inter-

acting high-level objects. These objects fall in two main categories: colvars and algorithms acting on them (also referred to as ‘biases’) that enhance sampling, implement restraints or user-defined potentials, and perform runtime analysis and PMF calculations. Biases have access to the values of colvars, and may apply external biasing forces on them; such forces are then transparently distributed on to the corresponding atoms.

Retrieval of the atomic coordinates and communication of any biasing forces are performed by a ‘proxy’ object. One of the functions of the proxy object is to combine all operations that are performed on the same atoms. This is a major advantage in the situation where multiple variables are computed on the same set of atoms. The proxy object is also the only section of the code that must be rewritten when interfacing to a different program, while the remainder of the source code remains unchanged. A proxy object for NAMD up to version 2.9 [18,19] was part of our initial implementation, used to adapt the ABF method [25]. Recently, a version of this object to interface LAMMPS [20] was released for public usage, and the development of a version for GROMACS [67] is underway.

Within the section of the code that does not depend on the particular MD program, interaction between the proxy object and the colvars is implemented by objects representing *atom groups*. As input, these objects collect the Cartesian coordinates of the atoms and the forces originating from the system. Optionally, atom group objects can also calculate and apply to their atomic positions rotational transformations from the laboratory frame towards a non-inertial frame where the analytical expression of certain colvars is simpler. This a mandatory step for colvars based on Class V and VI components, but is also available to all types of colvars. The transformation between the laboratory frame and the rotated frame is described in more detail in the following (Section 3.1). During the calculation of each colvar, atom groups are responsible for storing the various derivatives of the colvar (gradients, inverse gradients and Jacobian term). Such derivatives can then be used implicitly, to propagate a generalised force acting on the colvar, or explicitly accessed by biasing and analysis algorithms.

There are no restrictions on which atoms can be included in the groups, other than limitations due to computational efficiency. Two factors play a role in creating these limitations: the first is the raw cost to calculate the functional forms of the components involved. In the current design of our software, we followed an approach that is most efficient in a typical case, where the coordinates of the atoms involved and their forces are collected and cached in contiguous memory locations, to benefit from the processor’s cache and compile-time optimisations. However, certain functional forms such as those with an $\mathcal{O}(N^2)$ overhead may have a non-negligible impact on performance. The second overhead is represented by the

communication of the coordinates of the atoms involved, as messages between different sections of the program in serial runs, and between different computational nodes in parallel runs. Although a parallelised design would be beneficial for very expensive colvars by reducing the size of the messages, most colvars are not additive (of those here described, the only exception is *distanceZ*). To evaluate a non-additive colvar and the corresponding biasing force, two iterations of broadcast and reduction are required between the master node and the other computational nodes, thereby multiplying the total latency times. Because the parallel performance of MD software is typically more sensitive to latency times than to the transfer bandwidth, we did not pursue so far a fully parallelised design, reserving it to future applications with extreme numbers of atoms involved. The impact on performance of both the raw cost and the communication overhead is analysed in more detail in Section 3.3.

3.1. Moving frames of reference

Atom group objects feature the ability to compute an optimal roto-translation that best superimposes its atomic coordinates \mathbf{x}_i to a set of reference coordinates $\mathbf{x}_i^{\text{ref}}$. This transformation is transparently applied at each step of the simulation, before the atomic coordinates are used to calculate the colvar components. Hence, the colvars are calculated on the following set of roto-translated coordinates:

$$\mathbf{x}_i' = R(\mathbf{x}_i - \mathbf{x}^{\text{C}}) + \mathbf{x}^{\text{ref}}, \quad (17)$$

where \mathbf{x}^{C} and \mathbf{x}^{ref} are the centres-of-geometry of the current and reference coordinates, and R is the rotation matrix that superimposes the two sets. Optionally, the atoms from which the roto-translation is calculated can be different from those from which the colvars are calculated. In that case, the gradients of the colvar, $\partial\xi/\partial\mathbf{x}_i$, are also propagated to all the atoms defining the rotation, even though their positions do not appear explicitly in the colvar's mathematical expression.

Usage Case 7: permeation through a mobile channel. Thanks to custom reference frames, permeation of a small molecule through a pore or channel may be studied using a *distanceZ* component based on a specified axis, even if the channel's position and orientation are not constant in time. Cancelling the rotation of the channel ensures that the defined axis always describes the direction of permeation, while compensating for overall translation keeps the reference position fixed with respect to the channel. In such a setup, the rotation contribution to the colvar gradient has non-zero terms on the channel atoms. Thus, biasing forces on the solute will automatically induce a counter-force on the

channel, even though only solute atoms are explicitly part of the colvar definition.

Class IV components such as *rmsd* and *eigenvector* are defined based on the distance between the current coordinates and the reference coordinates: in most applications, a moving frame of reference is useful to remove tumbling motions that are physically irrelevant. For situations where a physically relevant rotation is to be measured, Class V components such as *orientation* and its derivatives are an appropriate choice of colvar. Such rotations may in turn be expressed in a moving frame of reference.

Usage Case 8: relative rotation of interacting membrane proteins. The *spinAngle* coordinate describes the rotation of an object around an axis. When applied to a trans-membrane protein or peptide, it is most useful when expressed in a rotating frame of reference linked to another membrane-embedded protein: it then describes relative rotation of one protein with respect to the other. Combined with an in-plane distance (*distanceXY*), it provides a more detailed description of the mode of lateral interaction of the two proteins. Internally, measuring such a relative rotation implies two nested coordinate fits: first, fitting the dimer to its reference positions to cancel global rotation and, second, fitting one of the proteins on to its own reference structure to measure its rotation within the dimer.

The optimal translation is defined as the difference between the centres-of-geometry of the group of atoms and of the reference coordinates. The optimal rotation, instead, is computed via a quaternion-based procedure [28] that uses the signalisation of a 4×4 matrix constructed using both the current and the reference coordinates. Given the eigenvalues $\lambda^{(1)} > \lambda^{(2)} > \lambda^{(3)} > \lambda^{(4)}$ and the corresponding four-dimensional eigenvectors ($\mathbf{q}^{(1)}$, $\mathbf{q}^{(2)}$, $\mathbf{q}^{(3)}$, $\mathbf{q}^{(4)}$) of this matrix, the leading eigenvector $\mathbf{q}^{(1)}$ is a unit quaternion that describes a rotation that minimises the RMSD between the current and the reference coordinates [28]. While giving identical results to the widely used Kabsch approach [68], the quaternion-based approach provides a simpler mathematical object, a unit quaternion, which is more suitable to be used as a colvar than a 3×3 unitary matrix. It also involves a linear loop over the set of coordinates, which is easily optimised by taking advantage of the cache memory of modern processors. The unit quaternion $\mathbf{q}^{(1)}$ computed via the procedure of Dill and coworkers [28] is related to the unitary matrix of rotation R by the

expression

$$R(\mathbf{q}^{(1)}) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_0q_3 + q_1q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}, \quad (18)$$

where $\mathbf{q}^{(1)} = (q_0, q_1, q_2, q_3)$.

The derivatives of $\lambda^{(1)}$ and $\mathbf{q}^{(1)}$ can be expressed on the orthonormal basis in \mathbb{R}^4 formed by $(\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \mathbf{q}^{(3)}, \mathbf{q}^{(4)})$:

$$\frac{\partial \mathbf{q}^{(1)}}{\partial x_i} = \sum_{\alpha=1}^4 w_i^{(1,\alpha)} \mathbf{q}^{(\alpha)}, \quad (19)$$

where $\partial/\partial x_i$ indicates the derivative with respect to the i th Cartesian coordinate (i going from 1 to $3N$). Differentiating both sides of the eigenvalue equation $\mathbf{S}\mathbf{q}^{(1)} = \lambda^{(1)}\mathbf{q}^{(1)}$ and solving for the coefficients $w_i^{(1,\alpha)}$, the following expressions are obtained:

$$\frac{\partial \lambda^{(1)}}{\partial x_i} = \mathbf{q}^{(1)} \cdot \frac{\partial \mathbf{S}}{\partial x_i} \mathbf{q}^{(1)}, \quad (20)$$

$$\frac{\partial \mathbf{q}^{(1)}}{\partial x_i} = \sum_{\alpha=2}^4 \frac{1}{\lambda^{(\alpha)} - \lambda^{(1)}} \left(\mathbf{q}^{(\alpha)} \cdot \frac{\partial \mathbf{S}}{\partial x_i} \mathbf{q}^{(1)} \right) \mathbf{q}^{(\alpha)}, \quad (21)$$

where the matrix $\partial \mathbf{S} / \partial x_i$ and all of the scalar products $(\mathbf{q}^{(\alpha)} \cdot (\partial \mathbf{S} / \partial x_i \mathbf{q}^{(\beta)}))$ are known. If the two sets of atomic coordinates $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ and $\mathbf{X}' = (\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_N)$ are different enough that the optimal transform is a roto-inversion rather than a direct rotation, crossing can occur between the eigenvalues $\lambda^{(1)}$ and $\lambda^{(2)}$, leading to discontinuous atomic derivatives [28]. The same phenomenon is also well known to occur with the Kabsch method [68]. No general algorithm exists to rigorously correct for the arising discontinuities: in our software implementation, a message is printed to warn if the inner product between eigenvectors computed at two consecutive time steps, $\mathbf{q}^t \cdot \mathbf{q}^{t+\Delta t}$, differs from 1 by more than a predefined threshold (0.01 by default).

3.2. Extending the set of variable components

The colvars module offers the possibility to create new variables finely tuned to a specific system, by combining existing components in a polynomial sum:

$$\xi(\mathbf{X}) = \sum_{\alpha} C_{\alpha} (z^{(\alpha)}(\mathbf{X}))^{p_{\alpha}} \quad (22)$$

where C_{α} are real constants and p_{α} non-negative integers. Combinations at the source code level are also straightforward: for example, the *alpha* component is entirely implemented as a combination of *angle* and *coordNum*

with virtually no replicated code. Similarly, the *dihedralPC* component is built internally as an array of *dihedral* components with a small amount of added functionality.

Usage Case 9: mean and fluctuation of the dipole moment of a water cluster. The projection on to a chosen axis \mathbf{e} of the dipole moment \mathbf{d} of a water molecule can be calculated as a *distanceZ* function between the centre-of-mass of the two hydrogen atoms and the oxygen atom. If we define $d_x = \mathbf{d} \cdot \mathbf{e}$, it is possible to calculate the average dipole moment $\langle d_x \rangle = N^{-1} \sum_{\alpha} d_{x,\alpha}$ as a combination using $C_{\alpha} = N^{-1}$ and $p_{\alpha} = 1$. The average square dipole moment $\langle d_x^2 \rangle = N^{-1} \sum_{\alpha} d_{x,\alpha}^2$ can then be defined as a combination using $C_{\alpha} = N^{-1}$ and $p_{\alpha} = 2$; from that quantity, a single *distanceZ* function defined between all hydrogens and all oxygens using $C_{\alpha} = N^{-2}$ and $p_{\alpha} = 2$ can be subtracted to obtain the variance, $\langle d_x^2 \rangle - \langle d_x \rangle^2$.

3.3. Computational cost of colvars and algorithms

We performed three groups of benchmark calculations with our software, as implemented in NAMD version 2.9 [18,19]. The first group of tests estimates the raw cost of computing the basic quantities on the smallest test system (two atoms). The second group of tests examines the computational cost as a function of the number of atoms (up to several thousands). The third group of tests shows the performance of the colvar calculations in parallel clusters or supercomputers.

To evaluate the raw cost to compute one colvar, we first tested a $\text{Na}^+ - \text{Cl}^-$ pair in an implicit solvent, with the same configuration as used for Figure 2. The average time per MD step with no colvar calculation was 21.3 μs . If a *distance* colvar was defined, the average time per MD step increased to 37.9 μs . A PMF calculation with metadynamics, interpolated by a grid with the same parameters as Figure 2(b) yielded an average time per MD step of 40.6 μs . A very similar timing (40.1 μs) was obtained with ABF [25] defined on an identical grid.

We then proceeded to examine the computational cost as a function of the number of atoms (Figure 4). We simulated three NaCl cubic lattices, with their unit cells composed of 128, 1024 or 4096 atoms. Electrostatic interactions were treated by the particle mesh Ewald method [69] with a grid spacing of 1 Å, and van der Waals interactions were cut off at 12 Å. The colvars examined were: the distance

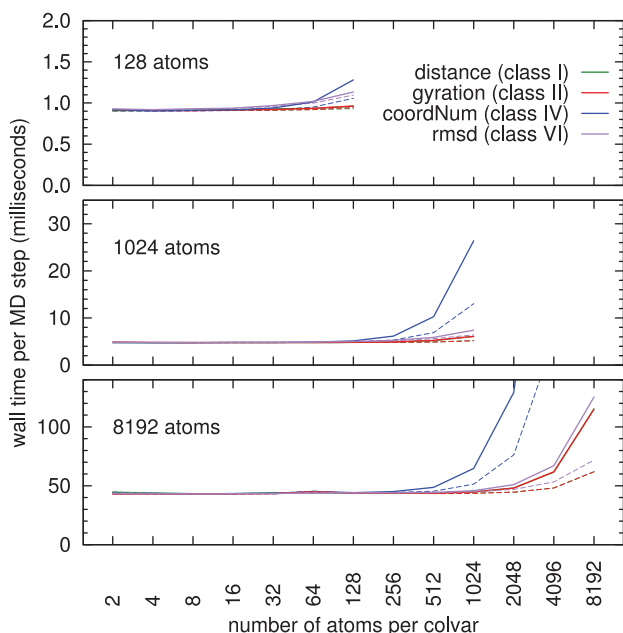


Figure 4. Time per MD step to calculate colvars based on different components (*distance*, *gyration*, *coordNum* and *rmsd*), for three different NaCl lattices of 128, 1024 and 8192 atoms. The solid lines show the times per MD step, including the calculation of each colvar and of its gradients, inverse gradients and Jacobian term (only the colvar and its gradients for *coordNum*). The dashed lines show the times per MD step when only the value of colvar is computed. The dotted black lines show times per MD step when no colvars are computed.

between the centres-of-mass of the Na^+ and Cl^- atoms within the unit cell; the radius of gyration of the entire unit cell; the coordination number between the Na^+ and Cl^- atoms (computed including periodic boundary conditions); and the RMSD with respect to the original configuration. All colvars were computed on groups of atoms of different size, ranging from two to the entire unit cell. The four chosen colvars represent components of Classes I, II, IV and VI: we did not test components of Classes III and V, because their functional forms are analogous to Classes IV and VI, respectively.

The measured times per MD step are plotted in Figure 4 as a function of the number of atoms in each colvar. Class I and Class II colvars are virtually identical in computational cost, due to the fact that they both involve linear loops over the set of atomic positions. Class VI colvars such as *rmsd* feature a linear cost as well, but also include a higher overhead due to the calculation of the rotation matrix [28]. Class III colvars such as *coordNum* are the most expensive, because they involve $N(N-1)/2$ calculations of mutual distances between two atoms. For all colvars except those of Class III, the cost of computing the colvar and all of its derivatives is marginally higher than that of computing only the colvar.

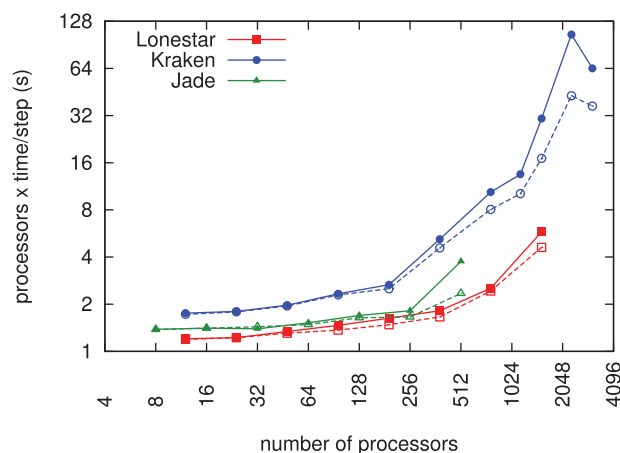


Figure 5. Benchmark results for ApoA1 on three different supercomputers: the TACC InfiniBand Linux cluster ‘Lonestar’ (squares), the NICS Cray XT5 supercomputer ‘Kraken’ (circles) and the CINES SGI Altix supercomputer ‘Jade’ (triangles). The empty symbols indicate the timings for MD simulations without colvars defined, filled symbols the timings for the calculation of a three-dimensional PMF along the position of the centre-of-mass of the protein C_α carbons (looping over 392 atoms, or 1176 individual Cartesian components).

Finally, we tested a macromolecular system in a parallel calculation, the protein–lipid apoA1 complex traditionally used for NAMD scaling tests (Figure 5). We defined three colvars, which are the X , Y and Z coordinates of the centre-of-mass of the C_α atoms of the two protein chains (392 atoms, or 1176 individual Cartesian components), defined as three separate *distanceZ*-based colvars. A metadynamics bias was applied to the three variables (X , Y , Z), using a cubic grid of size 100 Å and spacing 0.5 Å (200 grid points in each direction, or 8 million in total). This represents a relatively demanding example, aiming at describing conformations of large supramolecular assemblies at high detail. We tested this calculation on three high-performance computing resources: the InfiniBand Linux cluster ‘Lonestar’ at the Texas Advanced Computing Center (TACC), with two six-core ‘Westmere’ processors per node; the Cray XT5 ‘Kraken’ supercomputer at the National Institute for Computational Sciences (NICS) with two six-core AMD ‘Istanbul’ processors per node; and the SGI Altix supercomputer ‘Jade’ at the Centre Informatique National de l’Enseignement Supérieur (CINES) with two quad-core Intel ‘Harpertown’ processors per node, connected by an InfiniBand network.

Although measurable at node counts above 64, the overhead raised above a few percentage points only when the number of CPU cores used was well beyond the linear-scaling regime for the ‘raw’ performance without colvars (empty symbols in Figure 5). At the best configuration (768 cores of the ‘Lonestar’ supercomputer) and using a 2 fs integration time step, we measured a peak simulation speed of 3 ns/hour.

3.4. Multiple-replica parallelism

Due to the increase in the number of computing units in modern simulation resources, the potential to accelerate the simulation throughput is higher. However, communication between computational nodes occurs through hardware whose transfer bandwidths do not increase proportionately: the degree of parallelisation that can be achieved is therefore limited. One popular approach to further increase sampling is to combine colvar-based algorithms with a parallelism between many communicating replicas of the system [10–12,70–72]. In the case where all replicas share the same macroscopic parameters (temperature, pressure, etc.) and only differ in their microscopic degrees of freedom, interpretation of the results is especially straightforward. One of these algorithms is the ‘multiple-walker’ approach to the metadynamics algorithm [70]: by this approach, the potential $V_{\text{meta}}(\xi)$ is constructed from multiple equivalent replicas that concurrently explore the configurational space:

$$V_{\text{meta}}(\xi) = \sum_{r=1}^{N_{\text{rep}}} V_{\text{meta}}^{(r)}(\xi). \quad (23)$$

Unlike ‘embarrassingly parallel’ approaches such as free-energy perturbation [58] or umbrella sampling [1], the adaptive nature of the metadynamics algorithm is preserved.

Our implementation of the multiple-walker algorithm relies on a flexible communication system, wherein the individual biasing potentials $V_{\text{meta}}^{(r)}(\xi)$ and their gradients $\partial V_{\text{meta}}^{(r)}(\xi)/\partial \xi$ are shared by all replicas through an asynchronous communication. The communication is based on temporary files, with a network communication under development. This effectively allows us to exploit the full computational power of the largest supercomputers. It is often observed that the performance of an MD simulation fluctuates with time, depending on many factors (bursts in the network traffic, operating system jitter, time spent on input/output, etc.). Therefore, enforcing strict synchronisation between replicas may damage the parallel performance of the entire set. The approach chosen here eliminates this problem without making use of auxiliary parameterisation of the density of states to achieve asynchronous coupling when using distributed computing resources [73–75].

To test the performance as a function of the number of replicas, we used the apoA1 system simulated by NAMD version 2.9 [18,19]. We tested a metadynamics run with the same configuration as in Figure 5, reduced, however, to two variables (X and Y) for comparison with the version of Hamiltonian exchange [11] recently implemented in NAMD with MPI communication between replicas. Each replica was run on 1536 CPU cores of the Cray XK6 ‘Jaguar’ supercomputer. The performance of a single replica did not differ significantly between the two approaches, except for the difference in the build of the

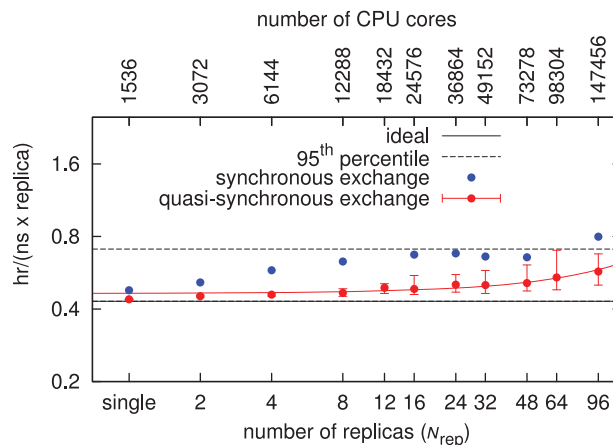


Figure 6. Performance of the multiple-replica communication scheme for colvar biases. Shown are the average single-replica performance $p(\text{single})$ (solid black line) and its 95th percentile (dashed line). The blue lines delimit the 95% interval of the instantaneous performance of an individual replica. The red points indicate $p(N_{\text{rep}})$, the median performance of each set of replicas: error bars indicate the 95% interval of the individual performances around $p(N_{\text{rep}})$. The blue points indicate the performance of the Hamiltonian exchange approach: error bars are absent due to synchronisation between replicas. The line indicates a least-squares fit with the formula $p(N_{\text{rep}}) = p(\text{single})(1 - \gamma(N_{\text{rep}} - 1))$, with $\gamma \simeq 0.0028$. The total number of CPU cores used for each test calculation is shown on the top axis.

Charm++ communication library used by NAMD. In the metadynamics run, each replica inserted a new Gaussian hill every 1000 MD steps and communicated it to the other replicas at the same frequency. In the Hamiltonian exchange run, synchronisation and exchange of the harmonic centres between replicas was also performed every 1000 steps.

In Figure 6 we plot the performance as a function of the number of replicas. Due to the fluctuations over time of the single-replica performance $p(\text{single})$, synchronisation causes the entire set to run at the speed of the slowest replicas. Therefore, the combined performance $p(N_{\text{rep}})$ may degrade depending on the distribution of $p(\text{single})$. In our test, $p(N_{\text{rep}})$ was roughly equal to the 95th percentile of $p(\text{single})$ when N_{rep} approaches 20 (Figure 6). In the metadynamics run, loss of ideal scaling occurred instead at higher N_{rep} , suggesting that only the latencies involved in file access added to the cost of communication. With the current file-based communication, the simulation retains good scaling up to $\sim 150,000$ computing cores of the Jaguar supercomputer, equalling 1.6 petaflops of peak performance and roughly half of the entire machine. At this configuration, a multiple-walker metadynamics calculation on apoA1 can sample up to $3.7 \mu\text{s}$ of aggregated simulation data per day. By performing communication between replicas with the same loose synchronisation as described so far but over the network, N_{rep} could be increased up to several hundreds, proportionally increasing the

aggregated simulation data sampled per day up to hundreds of microseconds.

Conclusions

We have presented a highly efficient and general implementation of colvar-based MD calculations. Classic as well as recent numerical methods are implemented. The wide and easily extensible sets of variables allow for modelling diverse phenomena. The software framework is designed to allow method developers to rapidly gain access to a computationally efficient implementation, embedded in popular community programs. Single-replica and multiple-replica versions of the implemented algorithms are tuned to exploit the massively parallel design of current supercomputing resources. The software architecture framework is designed to enable new applications of advanced MD simulations in the biological, chemical and materials sciences, whose increasing predictive power allows one to target problems inaccessible to experimental investigation.

Acknowledgements

The authors thank Klaus Schulten and James Phillips for their help in extending NAMD, and Christopher Harrison and Christophe Chipot for many constructive discussions. Axel Kohlmeyer is gratefully acknowledged for linking the software with LAMMPS, and Li Li for contributing the well-tempered metadynamics feature. Finally, we are indebted to the hundreds of users of the colvars module, whose feedback continues to be instrumental to the functionality and robustness of the software since its initial release. Part of this work was supported by NSF grant CHE-1212416. Access to supercomputing resources was provided by the NSF-XSEDE program (allocation TG-MCA93S020) and the GENCI-CINES program (grant number 2012-076198).

References

- [1] G.M. Torrie and J.P. Valleau, *J. Comput. Phys.* **23**, 187 (1977).
- [2] E.A. Carter, G. Ciccotti, J.T. Hynes, and R. Kapral, *Chem. Phys. Lett.* **156**, 472 (1989).
- [3] T. Huber, A.E. Torda, and W.F. van Gunsteren, *J. Comput. Aided Mol. Des.* **8**, 695 (1994).
- [4] H. Grubmüller, *Phys. Rev. E* **52**(3), 2893 (1995).
- [5] E. Darve and A. Pohorille, *J. Chem. Phys.* **115**, 9169 (2001).
- [6] L. Rosso and M.E. Tuckerman, *Mol. Sim.* **28**, 91 (2002).
- [7] A. Laio and M. Parrinello, *Proc. Natl. Acad. Sci. USA* **99**, 12562 (2002).
- [8] L. Maragliano and E. Vanden-Eijnden, *Chem. Phys. Lett.* **426**, 168175 (2006).
- [9] C. Chipot and A. Pohorille, editors, *Free Energy Calculations. Theory and Applications in Chemistry and Biology* (Springer-Verlag, Berlin, 2006).
- [10] Y. Sugita and Y. Okamoto, *Chem. Phys. Lett.* **314**, 141 (1999).
- [11] Y. Sugita, A. Kitao, and Y. Okamoto, *J. Chem. Phys.* **113** (15), 6042 (2000).
- [12] H. Fukunishi, O. Watanabe, and S. Takada, *J. Chem. Phys.* **116** (20), 9058 (2002).
- [13] C. Dellago, P.G. Bolhuis, F.S. Csajka, and D. Chandler, *J. Chem. Phys.* **108** (5), 1964 (1998).
- [14] E. Weinan, W.Q. Ren, and E. Vanden-Eijnden, *Phys. Rev. B* **66** (5) 052301 (2002).
- [15] A.K. Faradjian and R. Elber, *J. Chem. Phys.* **120** (23), 10880 (2004).
- [16] A.F. Voter, *J. Chem. Phys.* **106**, 4665 (1997).
- [17] D. Hamelberg, H. Mongan, and J.A. McCammon, *J. Chem. Phys.* **120** (24), 11919 (2004).
- [18] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, and K. Schulten, *J. Comput. Chem.* **26**, 1781 (2005).
- [19] M. Bhandarkar, A. Bhatele, E. Bohm, R. Brunner, F. Bue-lens, C. Chipot, A. Dalke, S. Dixit, G. Fiorin, P. Freddolino, P. Grayson, J. Gullingsrud, A. Gursoy, D. Hardy, C. Harrison, J. Hénin, W. Humphrey, D. Hurwitz, N. Krawetz, S. Kumar, D. Kunzman, C. Lee, R. McGreevy, C. Mei, M. Nelson, J. Phillips, O. Sarood, A. Shinozaki, D. Tanner, G. Zheng, and F. Zhu, *NAMD User's Guide, version 2.8* (University of Illinois at Urbana-Champaign, Urbana, IL, 2011).
- [20] S. Plimpton, *J. Comput. Phys.*, **117** (1), 1 (1995).
- [21] M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R.A. Broglia, and M. Parrinello, *Comp. Phys. Comm.* **180** (10), 1961 (2009).
- [22] J.G. Kirkwood, *J. Chem. Phys.* **3**, 300 (1935).
- [23] W.K. den Otter, *J. Chem. Phys.* **112**, 7283 (2000).
- [24] G. Ciccotti, R. Kapral, and E. Vanden-Eijnden, *Chem. Phys. Chem.* **6** (9), 1809 (2005).
- [25] J. Hénin, G. Fiorin, C. Chipot, and M.L. Klein, *J. Chem. Theory Comput.* **6** (1), 35 (2010).
- [26] A. Barducci, G. Bussi, and M. Parrinello, *Phys. Rev. Lett.* **100** (2), 020603 (2008).
- [27] A.C. Pan, D. Sezer, and B. Roux, *J. Phys. Chem. B* **112** (11), 3432 (2008).
- [28] E.A. Coutias, C. Seok, and K.A. Dill, *J. Comput. Chem.* **25** (15), 1849 (2004).
- [29] D.C. Rapaport, *J. Comput. Phys.* **60** (2), 306 (1985).
- [30] N.S. Martys and R.D. Mountain, *Phys. Rev. E* **59** (3, Part b), 3733 (1999).
- [31] T.F. Miller, M. Eleftheriou, P. Pattnaik, A. Ndirango, D. Newns, and G.J. Martyna, *J. Chem. Phys.* **116** (20), 8649 (2002).
- [32] H. Kamberaj, R.J. Low, and M.P. Neal, *J. Chem. Phys.* **122** (22), 224114 (2005).
- [33] A.V. Akimov, A.V. Nemukhin, A.A. Moskovsky, A.B. Kolomeisky, and J.M. Tour, *J. Chem. Theory Comput.* **4** (4), 652 (2008).
- [34] D. Sezer, J.H. Freed, and B. Roux, *J. Chem. Phys.* **128** (16), 5755 (2008).
- [35] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, and M.L. Klein, *J. Chem. Phys.* **79**, 926 (1983).
- [36] W.D. Cornell, P. Cieplak, C.I. Bayly, I.R. Gould, K.M. Merz, Jr, D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.C. Caldwell, and P.A. Kollman, *J. Am. Chem. Soc.* **117**, 5179 (1995).
- [37] A.D. MacKerell, D. Bashford, M. Bellott, R.L. Dunbrack, J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus, *J. Phys. Chem. B* **102** (18), 3586 (1998).
- [38] W.L. Jorgensen, D.S. Maxwell, and J. Tirado-Rives, *J. Am. Chem. Soc.* **118** (45), 11225 (1996).
- [39] L.D. Schuler, X. Daura, and W.F. van Gunsteren, *J. Comput. Chem.* **22** (11), 1205 (2001).

- [40] A.K. Rappe, C.J. Casewit, K.S. Colwell, W.A. Goddard, and W.M. Skiff, *J. Am. Chem. Soc.* **114** (25), 10024 (1992).
- [41] W. Shinoda, R. Devane, and M.L. Klein, *Mol. Sim.* **33** (1–2), 27 (2007).
- [42] A. Amadei, A.B.M. Linssen, and H.J.C. Berendsen, *Proteins* **17** (4), 412 (1993).
- [43] C.F. Abrams and E. Vanden-Eijnden, *Proc. Natl. Acad. Sci. USA* **107** (11), 4961 (2010).
- [44] N.M. Glykos, *J. Comput. Chem.* **27** (14), 1765 (2006).
- [45] D. Alemani, F. Collu, M. Cascella, and M. Dal Peraro, *J. Chem. Theory Comput.* **6** (1), 315 (2010).
- [46] M. Iannuzzi, A. Laio, and M. Parrinello, *Phys. Rev. Lett.* **90** (23), 238302 (2003).
- [47] Y. Mu, P.H. Nguyen, and G. Stock, *Proteins* **58** (1), 45 (2005).
- [48] A. Altis, P.H. Nguyen, R. Hegger, and G. Stock, *J. Chem. Phys.* **126** (24), 244111 (2007).
- [49] J. Wang, Y. Wu, C. Ma, G. Fiorin, J. Wang, L.H. Pinto, R.A. Lamb, M.L. Klein, and W.F. DeGrado, *Proc. Natl. Acad. Sci. USA* **110** (4), 1315 (2013).
- [50] G. Brannigan, D.N. LeBard, J. Hénin, R.G. Eickenhoff, and M.L. Klein, *Proc. Natl. Acad. Sci. USA* **107** (32), 14122 (2010).
- [51] K.T. O'Neil and W.F. DeGrado, *Science* **250** (4981), 646 (1990).
- [52] K. Daniilidis, *Int. J. Robot. Res.* **18** (3), 286 (1999).
- [53] C. Hu, M.Q.H. Meng, M. Mandal, and P.X. Liu, *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, June 25–28, 2006, Luoyang, China, 2006.
- [54] G. Lipari and A. Szabo, *J. Am. Chem. Soc.* **104**, 4546 (1982).
- [55] Y. Deng and B. Roux, *J. Phys. Chem. B* **113** (8), 2234 (2009).
- [56] J.C. Gumbart, B. Roux, and C. Chipot, *J. Chem. Theory Comput.* **9**, 794 (2013).
- [57] S. Izrailev, S. Stepaniants, M. Balsera, Y. Oono, and K. Schulten, *Biophys. J.* **72** (4), 1568 (1997).
- [58] R.W. Zwanzig, *J. Chem. Phys.* **22**, 1420 (1954).
- [59] J. Schlitter, M. Engels, and P. Krüger, *J. Mol. Graph.* **12** (2), 84 (1994).
- [60] E. Darve, D. Rodríguez-Gómez, and A. Pohorille, *J. Chem. Phys.* **128** (14), 144120 (2008).
- [61] J. Hénin and C. Chipot, *J. Chem. Phys.* **121**, 2904 (2004).
- [62] T. Lelièvre, M. Rousset, and G. Stoltz, *J. Chem. Phys.* **126** (13), 134111 (2007).
- [63] T. Lelièvre, M. Rousset, and G. Stoltz, *Free Energy Computations: A Mathematical Perspective* (Imperial College Press, London, 2010).
- [64] L. Zheng and W. Yang, *J. Chem. Theory Comput.* **8** (3), 810 (2012).
- [65] L. Zheng, M. Chen, and W. Yang, *Proc. Natl. Acad. Sci. USA* **105** (51), 20227 (2008).
- [66] A. Onufriev, D. Bashford, and D.A. Case, *Proteins* **55**, 383 (2004).
- [67] E. Lindahl, B. Hess, and D. Van der Spoel, *J. Mol. Mod.* **7**, 306 (2001).
- [68] W. Kabsch, *Acta Cryst.* **34** (5), 827 (1978).
- [69] U. Essman, L. Perela, M.L. Berkowitz, T. Darden, H. Lee, and L.G. Pedersen, *J. Chem. Phys.* **103**, 8577 (1995).
- [70] P. Raither, A. Laio, F. L. Gervasio, C. Micheletti, and M. Parrinello, *J. Phys. Chem. B* **110** (8), 3533 (2006).
- [71] G. Bussi, F.L. Gervasio, A. Laio, and M. Parrinello, *J. Am. Chem. Soc.* **128** (41), 13435 (2006).
- [72] S. Piana and A. Laio, *J. Phys. Chem. B* **111** (17), 4553 (2007).
- [73] M. Hagen, B. Kim, P. Liu, R.A. Friesner, and B.J. Berne, *J. Phys. Chem. B* **111** (6), 1416 (2007).
- [74] E. Gallicchio, R.M. Levy, and M. Parashar, *J. Comput. Chem.* **29** (5), 788 (2008).
- [75] S. Rauscher, C. Neale, and R. Pomès, *J. Chem. Theory Comput.* **5** (10), 2640 (2009).

Appendix: Tables of colvar components

In the following, for each implemented colvar component $z(\mathbf{X})$, we list its range of values, its atomic gradients $\nabla_{\mathbf{X}} z(\mathbf{X})$ and, when available analytically, the inverse gradients $\frac{\partial \mathbf{X}}{\partial z}$ and the Jacobian term $\frac{\partial \ln|J|}{\partial z}$. In a few cases, the infinity symbol ∞ is used to indicate that the range of values of a function is not limited by the functional form, but only by the dimensions of the system and by its boundary conditions.

Table A1. Class I colvar components, based on distances between the centres-of-mass $\mathbf{x}_1^C, \mathbf{x}_2^C, \dots$ of several atomic groups. The vectors $\Delta \mathbf{x}^{i,j}$ are the differences $(\mathbf{x}_j^C - \mathbf{x}_i^C)$, and the symbol \times indicates an outer product. The axis \mathbf{e} can be defined as a fixed unit vector, or based on the distance vector between \mathbf{x}_2^C and the centre-of-mass of a third group \mathbf{x}_3^C , as in $\mathbf{e} = (\mathbf{x}_3^C - \mathbf{x}_2^C) / \|\mathbf{x}_3^C - \mathbf{x}_2^C\|$. Gradients $\nabla_{\mathbf{x}} z(\mathbf{X})$ are written as derivatives with respect to $\mathbf{x}_1^C, \mathbf{x}_2^C, \dots$ in curly braces, $\{\nabla_{\mathbf{x}_1^C} z(\mathbf{X}), \nabla_{\mathbf{x}_2^C} z(\mathbf{X}), \dots\}$; individual atomic gradients consist of one of these terms multiplied by $\partial \mathbf{x}_k^C / \partial \mathbf{x}_i$. Analytical expressions for $\partial \mathbf{X} / \partial z$ and $\partial \ln |J| / \partial z$ are currently not available for *distanceDir*. K is a normalisation constant analytically computed to obey Equation (4). Well-known, yet lengthy expressions for the gradients of the *angle* and of the *dihedral* (torsional angle) components are omitted for brevity. The computational overhead to compute these variables is $\mathcal{O}(N)$.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{x}} z(\mathbf{X})$	Inverse gradient $\frac{\partial \mathbf{X}}{\partial z}$	Jacobian term $\frac{\partial \ln J }{\partial z}$
<i>distance</i> $[0: \infty)$	$\ \Delta \mathbf{x}\ \equiv \ \mathbf{x}_2^C - \mathbf{x}_1^C\ $	$\frac{1}{\ \Delta \mathbf{x}\ } \{-\Delta \mathbf{x}, \Delta \mathbf{x}\}$	$\frac{1}{2 \ \Delta \mathbf{x}\ } \{-\Delta \mathbf{x}, \Delta \mathbf{x}\}$	$\frac{2}{\ \Delta \mathbf{x}\ }$
<i>distanceDir</i> (\mathbb{S}^2)	$\frac{\Delta \mathbf{x}}{\ \Delta \mathbf{x}\ }$	$\left(-\frac{1}{\ \Delta \mathbf{x}\ ^3} \begin{pmatrix} \Delta x_1 & \Delta x_1 & \Delta x_1 \\ \Delta x_2 & \Delta x_2 & \Delta x_2 \\ \Delta x_3 & \Delta x_3 & \Delta x_3 \end{pmatrix} + \frac{1}{\ \Delta \mathbf{x}\ ^2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) \{-\Delta \mathbf{x}, \Delta \mathbf{x}\}$		
<i>distanceZ</i> $(-\infty: \infty)$	$(\Delta \mathbf{x}) \cdot \mathbf{e}$	$\{-\mathbf{e}, \mathbf{e}\}$	$\frac{1}{2} \{-\mathbf{e}, \mathbf{e}\}$	0
<i>distanceXY</i> $[0: \infty)$	$\ \Delta \mathbf{x}_\perp\ \equiv \ \Delta \mathbf{x} - (\Delta \mathbf{x} \cdot \mathbf{e}) \mathbf{e}\ $	$\frac{1}{\ \Delta \mathbf{x}_\perp\ } \{-\Delta \mathbf{x}_\perp, \Delta \mathbf{x}_\perp\}$	$\frac{1}{2 \ \Delta \mathbf{x}_\perp\ } \{-\Delta \mathbf{x}_\perp, \Delta \mathbf{x}_\perp\}$	$\frac{1}{\ \Delta \mathbf{x}_\perp\ }$
<i>angle</i> $[0: 180^\circ]$	$\cos^{-1} \left(\frac{\Delta \mathbf{x}^{2,1} \cdot \Delta \mathbf{x}^{2,3}}{\ \Delta \mathbf{x}^{2,1}\ \ \Delta \mathbf{x}^{2,3}\ } \right)$		$K \left\{ \frac{\partial \theta}{\partial \mathbf{x}_1^C}, 0, \frac{\partial \theta}{\partial \mathbf{x}_3^C} \right\}$	$\cot(\theta)$
<i>dihedral</i> $[-180^\circ: 180^\circ)$	$\tan^{-1} \left(\frac{(\Delta \mathbf{x}^{1,2} \times \Delta \mathbf{x}^{2,3}) \cdot \Delta \mathbf{x}^{3,4} \ \Delta \mathbf{x}^{2,3}\ }{(\Delta \mathbf{x}^{1,2} \times \Delta \mathbf{x}^{2,3}) \cdot (\Delta \mathbf{x}^{2,3} \times \Delta \mathbf{x}^{3,4})} \right)$		$K \left\{ \frac{\partial \omega}{\partial \mathbf{x}_1^C}, 0, 0, \frac{\partial \omega}{\partial \mathbf{x}_4^C} \right\}$	0

Table A2. Analytical expressions for Class II components, describing internal structure. The gradient $\nabla_{\mathbf{x}} z(\mathbf{X})$ and inverse gradient $\frac{\partial \mathbf{X}}{\partial z}$ are expressed in terms of the individual atom's position \mathbf{x}_i . When atomic masses m_i are included in the expression, \mathbf{x}^C is the centre-of-mass $(\sum_i m_i)^{-1} \sum_i \mathbf{x}_i m_i$; otherwise, it is the centre-of-geometry, $N^{-1} \sum_i \mathbf{x}_i$. Analytical expressions for the inverse gradient and Jacobian terms are currently available only for quantities that are not mass-weighted. The computational overhead for these components is $\mathcal{O}(N)$.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{x}_i} z(\mathbf{X})$	Inverse gradient $\frac{\partial \mathbf{x}_i}{\partial z}$	Jacobian term $\frac{\partial \ln J }{\partial z}$
<i>gyration</i> $[0: \infty)$	$\left(\frac{1}{N} \sum_i (\mathbf{x}_i - \mathbf{x}^C)^2 \right)^{1/2}$	$\frac{1}{zN} (\mathbf{x}_i - \mathbf{x}^C)$	$\frac{1}{z} (\mathbf{x}_i - \mathbf{x}^C)$	$\frac{3N-4}{z}$
<i>inertia</i> $[0: \infty)$	$\left(\frac{1}{\sum_j m_j} \sum_i m_i (\mathbf{x}_i - \mathbf{x}^C)^2 \right)^{1/2}$	$\frac{m_i}{z \sum_j m_j} (\mathbf{x}_i - \mathbf{x}^C)$		
<i>inertiaZ</i> $[0: \infty)$	$\sum_i m_i ((\mathbf{x}_i - \mathbf{x}^C) \cdot \mathbf{e})^2$	$2m_i (\mathbf{x}_i - \mathbf{x}^C)$		

Table A3. Analytical expressions for Class III colvar components, based on the distances between many pairs of atoms of one macromolecule. Atomic pairs are selected based on the chemical structure of the macromolecule (in the cases listed, a polypeptide chain). The computational overhead to compute these variables is $\mathcal{O}(N)$.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{x}} z(\mathbf{X})$
<i>alpha</i> (0: 1)	Angle term + H-bond term (Equation (7))	Equation (7)
<i>dihedralPC</i> (0: 1)	Backbone dihedral terms (Equation (8))	Equation (8)

Table A4. Analytical expressions for Class IV colvar components, based on multiple pairs of distances; \mathbf{d}^{ij} is $\mathbf{x}_2 - \mathbf{x}_1$, the distance vector between atom i and atom j ; d^0 is the user-defined isotropic ‘cutoff’ distance, whereas $(\mathbf{d}^0)^{-1}$ denotes a triplet of inverse cutoff distances in the three directions, defined as $(1/d_x^0, 1/d_y^0, 1/d_z^0)$. Expressions such as $\{-\mathbf{d}^{ij}, \mathbf{d}^{ij}\}$ indicate the two derivatives with respect to the positions of atoms i and j , respectively. The computational overhead for these components is $\mathcal{O}(N^2)$. In the expressions of the gradients of *coordNum* (isotropic and anisotropic versions), $(\|\mathbf{d}^{ij}\|/d^0)$ and $(\mathbf{d}^{ij} \cdot (\mathbf{d}^0)^{-1})$ are shortened to (\cdot) . The self-coordination number of one group of atoms (*selfCoordNum*) uses the same function as *coordNum*, but excludes the terms between each atom and itself with $\mathbf{d}^{ii} = 0$.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{X}} z(\mathbf{X})$
<i>distanceInv</i> (0: ∞)	$\left(\frac{1}{N_A N_B} \sum_{i,j} \left(\frac{1}{\ \mathbf{d}^{ij}\ } \right)^n \right)^{-1/n}$	$\frac{1}{n N_A N_B} \left(\frac{1}{N_A N_B} \sum_{i,j} \left(\frac{1}{\ \mathbf{d}^{ij}\ } \right)^n \right)^{-(n+1)/n} \frac{1}{\ \mathbf{d}^{ij}\ ^{n+2}} \{-\mathbf{d}^{ij}, \mathbf{d}^{ij}\}$
<i>coordNum</i> (0: $N_A N_B$)	$\sum_{i,j} \frac{1 - (\ \mathbf{d}^{ij}\ (d^0)^{-1})^n}{1 - (\ \mathbf{d}^{ij}\ (d^0)^{-1})^m}$	$\left(\frac{-1}{1 - (\cdot)^n} n (\cdot)^{n-1} - \frac{1 - (\cdot)^n}{1 - (\cdot)^m} m (\cdot)^{m-1} \right) \frac{1}{\ \mathbf{d}^{ij}\ d^0} \{-\mathbf{d}^{ij}, \mathbf{d}^{ij}\}$
<i>selfCoordNum</i> (0: N^2)		
<i>coordNum</i> (anis.) (0: $N_A N_B$)	$\sum_{i,j} \frac{1 - (\mathbf{d}^{ij} \cdot (\mathbf{d}^0)^{-1})^n}{1 - (\mathbf{d}^{ij} \cdot (\mathbf{d}^0)^{-1})^m}$	$\left(\frac{-1}{1 - (\cdot)^n} n (\cdot)^{n-1} - \frac{1 - (\cdot)^n}{1 - (\cdot)^m} m (\cdot)^{m-1} \right) \{- (\mathbf{d}^0)^{-1}, (\mathbf{d}^0)^{-1}\}$

Table A5. Analytical expressions for Class V colvar components (those based on pseudo-rigid orientations of a group of atoms). Analytical expressions for $\partial \mathbf{X} / \partial z$ and $\partial \ln |J| / \partial z$ are currently not available. All components are expressed as a function of the quaternion $\mathbf{q}(\mathbf{X}) = (q_0(\mathbf{X}), q_1(\mathbf{X}), q_2(\mathbf{X}), q_3(\mathbf{X}))$ and its derivatives (Equations (18) and (21)). $\mathbf{q}(\mathbf{X})$ describes the optimal rotation that, once applied to a set of reference coordinates \mathbf{X}^0 , minimises their distance from the current coordinates \mathbf{X} . \mathbf{X}^0 and \mathbf{X} are centred around zero prior computing \mathbf{q} , by subtracting their geometric centres. $\mathbf{q} = (q_1, q_2, q_3)$ is the vector portion of \mathbf{q} , and \mathbf{e} is a user-defined unit vector. The computational overhead of these components is $\mathcal{O}(N)$.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{X}} z(\mathbf{X})$
<i>orientation</i> (\mathbb{S}^3)	$\mathbf{q} = (q_0, q_1, q_2, q_3)$	$\nabla_{\mathbf{X}} \mathbf{q} = (\nabla_{\mathbf{X}} q_0, \nabla_{\mathbf{X}} q_1, \nabla_{\mathbf{X}} q_2, \nabla_{\mathbf{X}} q_3)$ (Equation (21))
<i>orientationAngle</i> [0: 180°]	$\theta = 2 \cos^{-1}(q_0)$	$\frac{-2}{\sqrt{1 - q_0^2}} \nabla_{\mathbf{X}} q_0$
<i>orientationProj</i> [-1: 1]	$p = \cos(\theta) = 2 q_0^2 - 1$	$4 \nabla_{\mathbf{X}} q_0$
<i>spinAngle</i> [-180°: 180°]	$\phi = 2 \tan^{-1} \left(\frac{\mathbf{q} \cdot \mathbf{e}}{q_0} \right)$	$\frac{2}{q_0^2 + (\mathbf{q} \cdot \mathbf{e})^2} \left((\mathbf{q} \cdot \mathbf{e}) \nabla_{\mathbf{X}} q_0 - \sum_{\alpha=1}^3 e_{\alpha} q_0 \nabla_{\mathbf{X}} q_{\alpha} \right)$
<i>tilt</i> [-1: 1]	$t = \cos(\omega)$	$\frac{4 q_0 \nabla_{\mathbf{X}} q_0 - \frac{4 q_0 (\mathbf{q} \cdot \mathbf{e})}{q_0^2 + (\mathbf{q} \cdot \mathbf{e})^2} \left((\mathbf{q} \cdot \mathbf{e}) \nabla_{\mathbf{X}} q_0 - \sum_{\alpha=1}^3 e_{\alpha} q_0 \nabla_{\mathbf{X}} q_{\alpha} \right)}{\left(\cos \left(\frac{\phi}{2} \right) \right)^2}$

Table A6. Analytical expressions for Class VI components, describing changes in the internal structure. The gradient $\nabla_{\mathbf{X}} z(\mathbf{X})$ and inverse gradient $\frac{\partial \mathbf{X}}{\partial z}$ are expressed in terms of the individual atom's position \mathbf{x}_i . \mathbf{x}^C is the centre of the group of atoms: if masses appear in the expression, \mathbf{x}^C is the centre-of-mass $(\sum_i m_i)^{-1} \sum_i \mathbf{x}_i m_i$; otherwise, it is the centre-of-geometry, $N^{-1} \sum_i \mathbf{x}_i$. \mathbf{X}^{ref} is the set of reference coordinates (centred on the origin) and \mathbf{X} are the instantaneous centred coordinates. $R(\mathbf{X})$ (also indicated as simply R) is the optimal rotation matrix that minimises the RMSD between $R(\mathbf{X})\mathbf{X}^{\text{ref}}$ and \mathbf{X} . The computational overhead for these components is typically $\mathcal{O}(N)$. In the Jacobian term of RMSD, $\nabla_{\mathbf{X}} \cdot (R(\mathbf{X})\mathbf{X}^{\text{ref}})$ expands to $\sum_{i=1}^N \sum_{\alpha=1}^3 \sum_{\beta=1}^3 \partial R_{\alpha\beta} / \partial x_{i\alpha} x_{i\beta}^{\text{ref}}$, where the subscripts α and β , varying between 1 and 3, indicate respectively Cartesian components of vectors \mathbf{x}_i and \mathbf{x}_i^0 , and the corresponding elements of the rotation matrix R . Partial derivatives of $R_{\alpha\beta}$ are obtained by differentiating Equation (18) with respect to the four components of the quaternion \mathbf{q} , then substituting Equation (21). In the Jacobian term of the *eigenvector*, the term $(\nabla_{\mathbf{X}} R(\mathbf{X}))\mathbf{V} \cdot \mathbf{X} = \sum_{i=1}^N \sum_{\alpha=1}^3 v_{i\alpha} \sum_{\beta=1}^3 (\nabla_{\mathbf{x}_i} R_{\alpha\beta}) x_{i\beta}$ averages to zero in real-life cases, can be omitted from the gradient in numerical applications without noticeable changes in the results, except for the fact that the simplified expression is *more* stable in situations where the optimal rotation becomes ill-defined (poor structural alignment). Accordingly, the simplified expression is used by default in the implementation.

	Definition of $z(\mathbf{X})$	Gradient $\nabla_{\mathbf{x}_i} z(\mathbf{X})$	Inverse gradient $\frac{\partial \mathbf{x}_i}{\partial z}$	Jacobian term $\frac{\partial \ln J }{\partial z}$
$rmsd [0: \infty)$	$\left(\frac{1}{N} \sum_i (\mathbf{x}_i - R \mathbf{x}_i^{\text{ref}})^2 \right)^{1/2}$	$\frac{1}{Nf} (\mathbf{x}_i - R \mathbf{x}_i^{\text{ref}})$	$\frac{1}{f} (\mathbf{x}_i - R \mathbf{x}_i^{\text{ref}})$	$\frac{1}{f} (3N - 4 - \nabla_{\mathbf{X}} \cdot (R(\mathbf{X})\mathbf{X}^{\text{ref}}))$
$eigenvector (-\infty: \infty)$	$\sum_i R \mathbf{v}_i \cdot (\mathbf{x}_i - R \mathbf{x}_i^{\text{ref}})$	$R \mathbf{v}_i - ((\nabla_{\mathbf{x}_i} R)\mathbf{V}) \cdot \mathbf{X}$	$R \mathbf{v}_i$	$\nabla_{\mathbf{X}} \cdot R(\mathbf{X})\mathbf{V}$