THE JOURNAL OF PHYSICAL CHEMISTRY B
A JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

Open Access
This article is licensed under CC-BY 4.0

pubs.acs.org/JPCB
Article

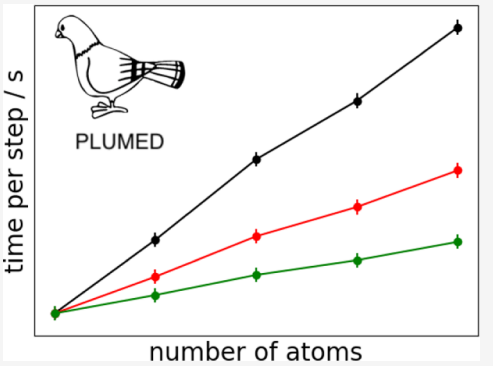
Making PLUMED Fly: A Tutorial on Optimizing Performance

Daniele Rapetti, Massimiliano Bonomi,* Carlo Camilloni,* Giovanni Bussi,* and Gareth A. Tribello*

Cite This: https://doi.org/10.1021/acs.jpcb.5c07562

Read Online
ACCESS | Metrics & More | Article Recommendations

ABSTRACT: PLUMED is an open-source software package that is widely used for analyzing and enhancing molecular dynamics simulations that works in conjunction with most available molecular dynamics softwares. While the computational cost of PLUMED calculations is typically negligible compared to the molecular dynamics code's force evaluation, the software is increasingly being employed to determine complex descriptors that are more computationally demanding. For these applications performance optimization becomes critical. In this tutorial, we describe a recently implemented tool that can be used to reliably measure code performance. We then use this tool to generate detailed performance benchmarks that show how calculations of large-numbers of distances, angles or torsions can be optimized by using vector-based commands rather than individual scalar operations. We then present benchmarks that illustrate how to optimize calculations of atomic order parameters and secondary structure variables. Throughout the tutorial and in our implementations we endeavor to explain the algorithmic tricks that are being used to optimize the calculations. We hope this allows others to understand these prescriptions and deploy them in their own calculations.



Downloaded via 95.248.237.174 on February 26, 2026 at 05:55:46 (UTC).
See https://pubs.acs.org/sharingguidelines for options on how to legitimately share published articles.

1. INTRODUCTION

PLUMED (https://www.plumed.org)[1] is an open-source software package that can be used to analyze and enhance molecular dynamics (MD) trajectories. Rather than operating as a monolithic software package, PLUMED serves as a framework for researchers who are using and developing advanced sampling techniques to share ideas. Consequently, in addition to the code, we also maintain a repository for sharing the inputs that have been used in publications that employ PLUMED (https://www.plumed-nest.org)[2] and a site for sharing tutorials that explain how to use its features (https://www.plumed-tutorials.org).[3]

As shown in Figure 1, PLUMED is designed to be inserted into MD codes and to work alongside them. It works by receiving the atomic positions from the underlying MD code, calculating various quantities from these positions and, if necessary, adjusting the forces that are acting upon the atoms. As we will discuss in the background section of this paper, PLUMED is written in a way that makes adding functionalities to calculate new quantities from these positions straightforward. This, combined with PLUMED's interoperability, has enabled developers to contribute their methodologies[4−37] within a unified ecosystem while maintaining compatibility across different molecular dynamics engines.[38−56] By establishing common interfaces and documentation standards, PLUMED has transformed how enhanced sampling methods are disseminated, validated, and adopted and effectively democratized access to cutting-edge simulation techniques and accelerated methodological innovation in the field.

Typically the computational cost of running PLUMED alongside an MD code is small. If PLUMED is being used to calculate a simple function of a small number of atomic positions, the cost of this calculation is negligible when compared to the cost associated with calculating the atomic forces. However, PLUMED is increasingly being used to perform more computationally expensive calculations. Given that in such calculations PLUMED can have a significant effect on performance, a tutorial paper that explains how to get the best performance from PLUMED feels timely. In the following sections we will thus explain some of the more complicated calculations that PLUMED can be used to perform and will then discuss various approaches that can be used to make the parts of these calculations that are performed in PLUMED run more quickly. We begin this survey in Section 5 by discussing collective variables that are functions of sums of distances, angles or torsions. Section 7 then discusses the calculation of symmetry functions and the tricks that can be used when calculating such functions in particular part of the simulation cell. We then discuss

Received: November 4, 2025
Revised: January 27, 2026
Accepted: February 2, 2026

ACS Publications
© XXXX The Authors. Published by American Chemical Society

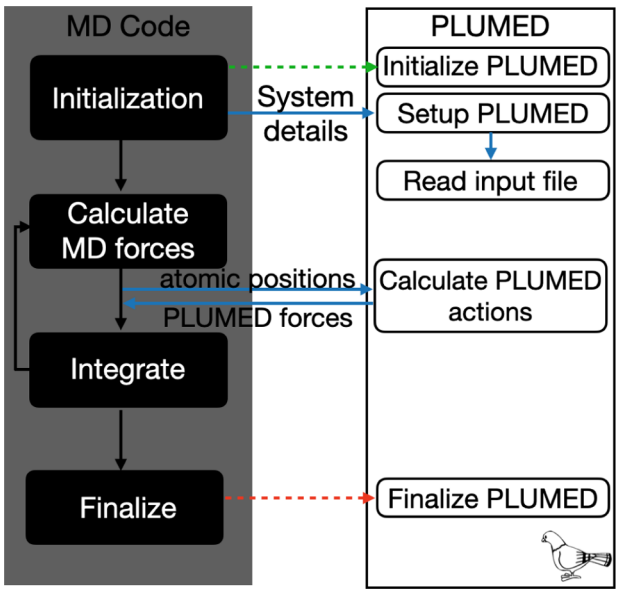


Figure 1. Interaction of PLUMED with molecular dynamics codes. PLUMED is able to calculate functions of the atomic positions and apply forces to atoms by passing data to and from the underlying MD code.

secondary structure variables in Section 9 before finishing with a discussion of the Steinhardt order parameters that are often used in studies of crystal nucleation in Section 10. Sections 3 and 4, which precede this survey, provide a brief introduction to PLUMED and an explanation of how the benchmarks that we discuss in the rest of the paper have been generated.

2. PREREQUISITES

To repeat the calculations described in this paper you need to configure the development version of PLUMED with the `--enable-modules=all` flag. PLUMED can then be compiled and run in the usual way. Our calculations were run on CINECA's LEONARDO GPU partition. PLUMED was compiled using gcc 12.2.0 and OpenMPI 4.1.6.

The timings for PLUMED that are reported in later figures are averages from 10 simulations. Error bars are used to indicate the standard deviation in all figures but these error bars are often smaller than the sizes of the points used to indicate the average.

3. BACKGROUND

Input to PLUMED typically consists of a single file that provides instructions on what PLUMED should calculate. An example input that illustrates how we can use PLUMED to add a restraint that forces the vector connecting atoms 1 and 2 to point in a direction that is perpendicular to the 111 direction of the lab frame is shown below.

```
# Calculate the vector connecting atoms 1 and 2
d: DISTANCE ATOMS=1,2 COMPONENTS
# Calculate the modulus of the vector connecting atom 1 and 2
dm: CUSTOM ARG=d.x,d.y,d.z FUNC=sqrt(x*x+y*y+z*z) PERIODIC=NO
# Calculate the director of the vector connecting atoms 1 and 2
ux: CUSTOM ARG=d.x,dm FUNC=x/y PERIODIC=NO
uy: CUSTOM ARG=d.y,dm FUNC=x/y PERIODIC=NO
uz: CUSTOM ARG=d.z,dm FUNC=x/y PERIODIC=NO
# Find the dot product between the director of the vector connecting
# atoms 1 and 2 and the 111 direction.
cc: COMBINE ARG=ux,uy,uz COEFFICIENTS=1,1,1 NORMALIZE PERIODIC=NO
# Calculate the angle between the vector connecting atoms 1 and 2
# and the 111 direction
ang: CUSTOM ARG=cc FUNC=acos(x) PERIODIC=NO
# Add an harmonic restraint to force this angle to remain close to 90 degrees
res: RESTRAINT ARG=ang AT=pi/2 KAPPA=100
```

Although this input performs a calculation that users would likely not perform, it does nicely illustrate PLUMED's philosophy. Each line in the input above defines an action and, by passing values between these actions, as illustrated in Figure 2 for the above input, the input defines a chain of operations that should be performed on the input positions. Notice that we also run through the action list backward as illustrated in the right panel of Figure 2 in order to evaluate the forces that our actions apply on the atomic positions. Also notice that, because the vector connecting the two atoms here is evaluated using the minimal image convention, it depends on the simulation box size. As such, when we do the backward propagation of the forces additional forces are added on the positions of the two atoms and on the cell vectors. These forces on the cell vectors contribute to the internal pressure of the system.

The input above also illustrates how we routinely use PLUMED's `CUSTOM` action, which relies on the Lepton library[57,58] that was developed by Peter Eastman and that we extracted from OpenMM.[40] This action provides users with a way of specifying arbitrary functions to be applied on the input values. In the above input one can thus see how the director of the vector connecting atoms 1 and 2 is computed from the components of the unnormalized vector and how the

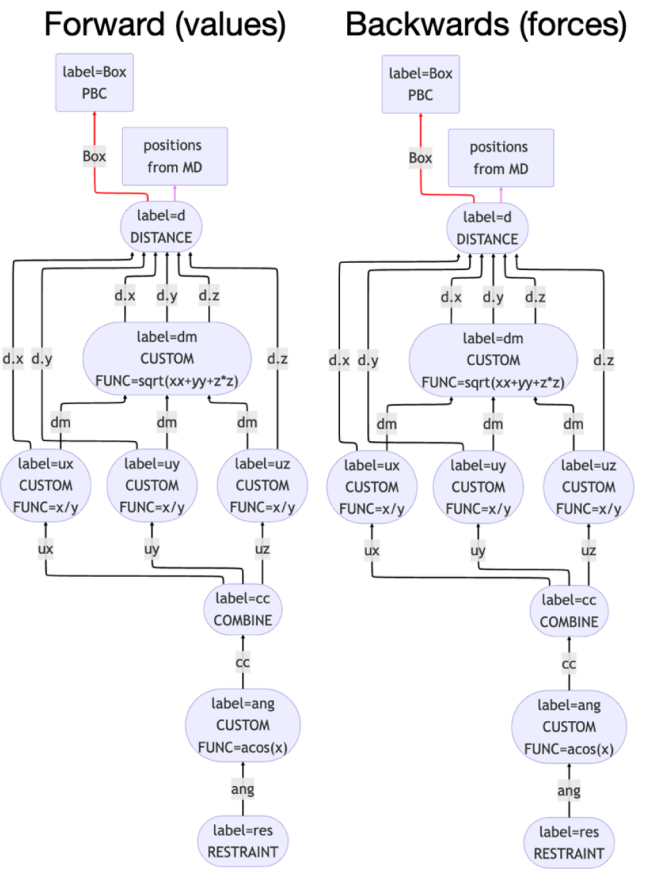


Figure 2. Data communications between PLUMED actions. The left panel illustrates how the constituent actions in the first example input in this paper evaluate the bias function. The right panel shows how data is passed between actions when forces are evaluated using the chain rule. These figures were made by using the PLUMED command `plumed show_graph`, which outputs a Mermaid diagram.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

angle between this vector and the (111) direction is computed by calculating the arccosine of a dot product.

Each action in a PLUMED input has an associated label. For the inputs in this paper these labels are the strings that appear before the colon. As indicated in the left panel of Figure 2, using these labels in the input to the ARG keyword of later actions allows one to reuse quantities calculated by earlier actions. In the input above, the quantities that are passed between the actions are all scalars. However, from PLUMED 2.10 onward you can also pass vectors in the same way. The following example illustrates how this passing of vectors works in practice. To make clear the types of value being passed we write the labels for quantities that are vectors in blue and labels for quantities that are scalars in black in this as well as all the inputs that follow.

```
# Calculate vectors that define the orientation of a collection of molecules
d: DISTANCE ...
   COMPONENTS
   ATOMS1=1,2 ATOMS2=3,4 ATOMS3=5,6 ATOMS4=7,8 # you can add more pairs here
...
# Now calculate the modulus of all the vectors above
dm: CUSTOM ARG=d.x,d.y,d.z FUNC=sqrt(x*x+y*y+z*z) PERIODIC=NO
# Calculate the directors of all the vectors
ux: CUSTOM ARG=d.x,dm FUNC=x/y PERIODIC=NO
uy: CUSTOM ARG=d.y,dm FUNC=x/y PERIODIC=NO
uz: CUSTOM ARG=d.z,dm FUNC=x/y PERIODIC=NO
# Calculate the mean from the three vectors above
mx: MEAN ARG=ux PERIODIC=NO
my: MEAN ARG=uy PERIODIC=NO
mz: MEAN ARG=uz PERIODIC=NO
# And calculate the modulus of the average
cv: CUSTOM ARG=ux,uy,uz FUNC=sqrt(x*x+y*y+z*z) PERIODIC=NO
```

The final quantity calculated in this input file is an order parameter that has been used to study liquid crystals.[59] Each pair of atoms specified in the distance command of the input gives an orientation for one of the molecules in the liquid crystal. The scalar quantity **cv** is thus equal to one if all the molecules in the liquid crystal have the same orientation and zero if all these molecules all have wildly different orientations.

To keep input files short, we provide shortcut commands. So, for example, you can calculate **cv** from the above input by using the single command:

```
cv: FERRONEMATIC_ORDER MOLECULE_STARTS=1,3,5,7  MOLECULE_ENDS=2,4,6,8
```

When PLUMED encounters this command it automatically generates the longer input file above and then uses it to perform the calculation. This approach has three advantages:

1. It reduces the amount of code that needs to be maintained
2. It allows us to quickly document what the FERRONEMATIC_ORDER command is computing by showing the longer version of the command that this input expands into within the PLUMED manual.
3. The long version of the command is also reported in the PLUMED log file, which facilitates the identification of problems.

As we illustrate in the remainder of this paper, the tools described above for quickly prototyping and documenting methods allow for cross fertilization of methods, ideas and approaches across different simulation communities. Furthermore, by reusing the same actions as much as possible we can provide the universal recommendations for optimizing performance that are the focus of the rest of this tutorial paper.

4. HOW TO EXAMINE PLUMED'S PERFORMANCE

Before discussing our prescriptions for improving the performance of PLUMED, it is worth explaining how the measures of PLUMED's performance that we have quoted in this tutorial have been generated. As we mentioned in the introduction, the computational expense associated with the calculations that PLUMED is performing is often negligible when compared with the calculation of the atomic forces. Furthermore, even when the calculations PLUMED performs have a non-negligiable contribution to the total simulation time, potential nonreproducibilities in the performance of the MD code might add noise and make PLUMED performance difficult to measure. It is thus suboptimal to run PLUMED alongside an MD code to measure its performance during the development stage. Furthermore, although one can run stand alone analyses of trajectories using PLUMED's driver utility, we often find that the time for such calculations is dominated by reading the trajectory. Consequently, using the `plumed driver` command to benchmark PLUMED is also misguided.

For these reasons, in PLUMED 2.10 we introduced a command line tool called `plumed benchmark` for reliably measuring performance across different variants of the PLUMED library. To get the graphs of performance in this paper we have used variations on the following command when employing this tool:

```
plumed benchmark --plumed pl.dat --natoms 1000 --atom-distribution sc
```

This command instructs PLUMED to repeatedly perform the calculations in the input file called `pl.dat` for a system of 1000 atoms that are arranged in an simple cubic structure. Consequently, the same set of positions are passed to PLUMED on every step but these positions are stored in memory so there is no need to do any molecular dynamics or disk access.

`plumed benchmark` has a number of features that may be useful for code developers who are worried about performance. Please note, first and foremost, that you can read the atomic positions that should be passed to PLUMED from most of the available trajectory formats. Consequently, if you are developing some exotic method to examine proteins or other complex molecules you can benchmark using a structure that is more relevant to the problem at hand than a simple cubic crystal by using a command such as:

```
plumed benchmark --plumed pl.dat --mf_pdb frame.pdb
```

PLUMED benchmark also allows you to run with multiple versions of PLUMED in parallel as illustrated below:

```
plumed-runtime benchmark --kernel /path/to/libplumedkernel.so:this
```

When you use this command, PLUMED alternates between performing the calculations using the version of PLUMED that is in the system PATH and the version of PLUMED in `/path/to/libplumedkernel.so`. The alternation is implemented to minimize the impact of the computer's load on the relative performance of the two versions that are being compared. Once the calculation is finished timings for the calculations with the two (or more) versions of the code are output to the log. The values PLUMED benchmark reports try to offset the initialization cost by not including it in the timings, and report an error in the relative performance of different PLUMED versions estimated using bootstrap.[60] Running such benchmark calculations to compare stable and development versions of the code is obviously useful if you are working on trying to optimize performance. However, it is important to remember that the timings output by PLUMED

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

benchmark will not tell you how long production calculations will take. Real-world performance will depend on technicalities related to the underlying MD code calculations, such as transfer of atoms from/to the GPU, use of caches, etc. We would always recommend that users fine-tune input files in an as-realistic-as-possible scenario, which often means running a short version of your production trajectory.

5. CALCULATING MULTIPLE DISTANCES, ANGLES, AND TORSIONS

There are two ways to use PLUMED to calculate the three distances between atom 1 and atoms 2, 3, and 4. You can use three actions that each pass out a single scalar as in the input below:

```
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=1,3
d3: DISTANCE ATOMS=1,4
```

Or you can use one action that passes out a 3 dimensional vector as in the input below.

```
d: DISTANCE ATOMS1=1,2 ATOMS2=1,3 ATOMS3=1,4
```

Similar pairs of options are available through the ANGLE and TORSION commands if you are calculating multiple angles or torsions.

When the number of distances being computed in these inputs is small, Figure 3 suggests that you will likely get very similar performance from these two options. However, if you are calculating a larger number of distances, the second option will provide much better performance as there are fewer virtual function calls for this second option and because the calculation of the distances in this second option can be parallelized using OpenMP and MPI. The crossovers in the top panel of Figure 3 suggest that using the second input becomes particularly important for getting the best performance out of PLUMED once you are computing approximately 100 distances. For sizes greater than this the cost of running the calculation with multiple OpenMP threads is cheaper than running the calculation on a single thread. If you have forces acting upon the distances using the input that allows for multi threading becomes important to performance once you are computing 50 or more distances (Figure 3 lower panel).

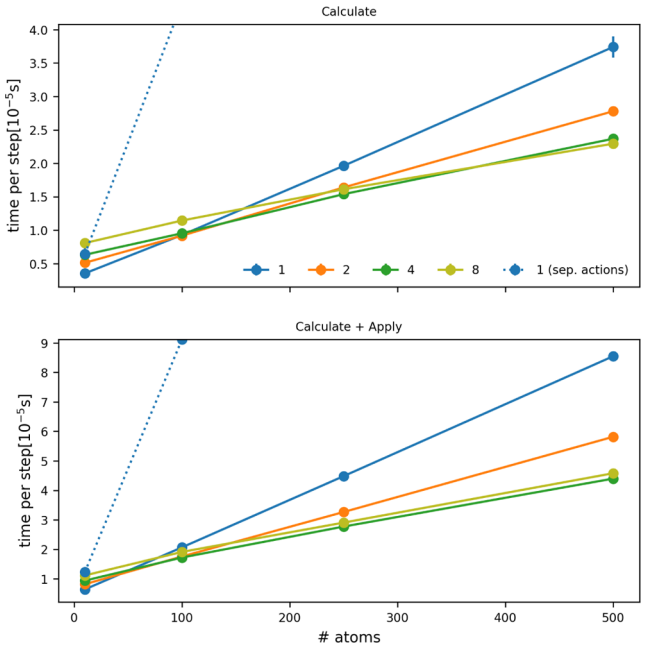


Figure 3. Time per step as a function of the number of distances that are being computed. The blue, orange, light and dark green lines indicate the cost of using a single DISTANCE command to calculate the vector of distances using 1, 2, 4, and 8 OpenMP threads, respectively. Meanwhile, the dotted line shows the cost of using N separate DISTANCE commands. The top panel indicates the cost of calculating the distances only, while the bottom panel indicates the additional cost that comes if you apply a force on the computed distances and also need to calculate derivatives.

The input that was used to generate the scaling plots in Figure 3 is shown below:

```
d: DISTANCE ATOMS1=1,2 ATOMS2=2,3 ATOMS3=3,4 # etc
s: SUM ARG=d PERIODIC=NO
RESTRAINT ARG=s KAPPA=1 AT=0
PRINT ARG=s FILE=colvar
```

This input tells PLUMED to calculate distances for every kth and $(k + 1)$th atom pair in the system. Consequently, if there are n atoms in the system $n - 1$ distances will be computed if we use the input above. These distances are then all added together and a restraint is applied on this sum. Similar inputs that used all $(n - 2)$ sets containing the kth, $(k + 1)$th and $(k + 2)$th atoms were used to benchmark the ANGLE command, while the $(n - 3)$ sets containing the kth, $(k + 1)$th, $(k + 2)$th and $(k + 3)$th atoms were used to benchmark the TORSION command. The results obtained from these calculations are shown in Figure 4.

The cost of these calculations increases linearly with the number of atoms as would be expected (Figure 4). Furthermore, having a force on these quantities roughly doubles the cost of the calculation, which, given that PLUMED recalculates all the distances/angles/torsions and their derivatives when applying forces using the chain rule, is to be expected. Most importantly, however, for the largest systems studied you can get a roughly factor 5 speed up by using 32 OpenMP threads rather than a single thread.

6. OPENMP VERSUS MPI

In the previous section we noted that PLUMED calculations can be parallelized using OpenMP or MPI. We focused on presenting our benchmarks with different numbers of OpenMP threads rather than different numbers of MPI processes because of the results shown in the left panel of Figure 5. To generate the lines in this figure we ran the TORSION benchmark that was introduced in the previous section using 8 OpenMP threads, 8 MPI processes, a pair of 4 OpenMP threads that communicate via MPI and four pairs of OpenMP threads that communicate via MPI. You get the best performance when you use pure OpenMP parallelism (solid light green line).

For these relatively small calculations, the cost per step decreases when you use up to 8 MPI processors (right panel Figure 5). However, when 16 or 32 MPI processes are used the cost of the calculation is increased by the additional communication so using fewer MPI processes is more efficient.

We also found that calculations running over N MPI processors each of which are running M OpenMP threads are often slower than calculations that simply run on M OpenMP threads. This is certainly the case for $N = 2$ and $M = 4$ but is not the case for $N = 4$ and $M = 2$ (dashed and solid orange and green lines left panel Figure 5). Consequently, if your MD

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

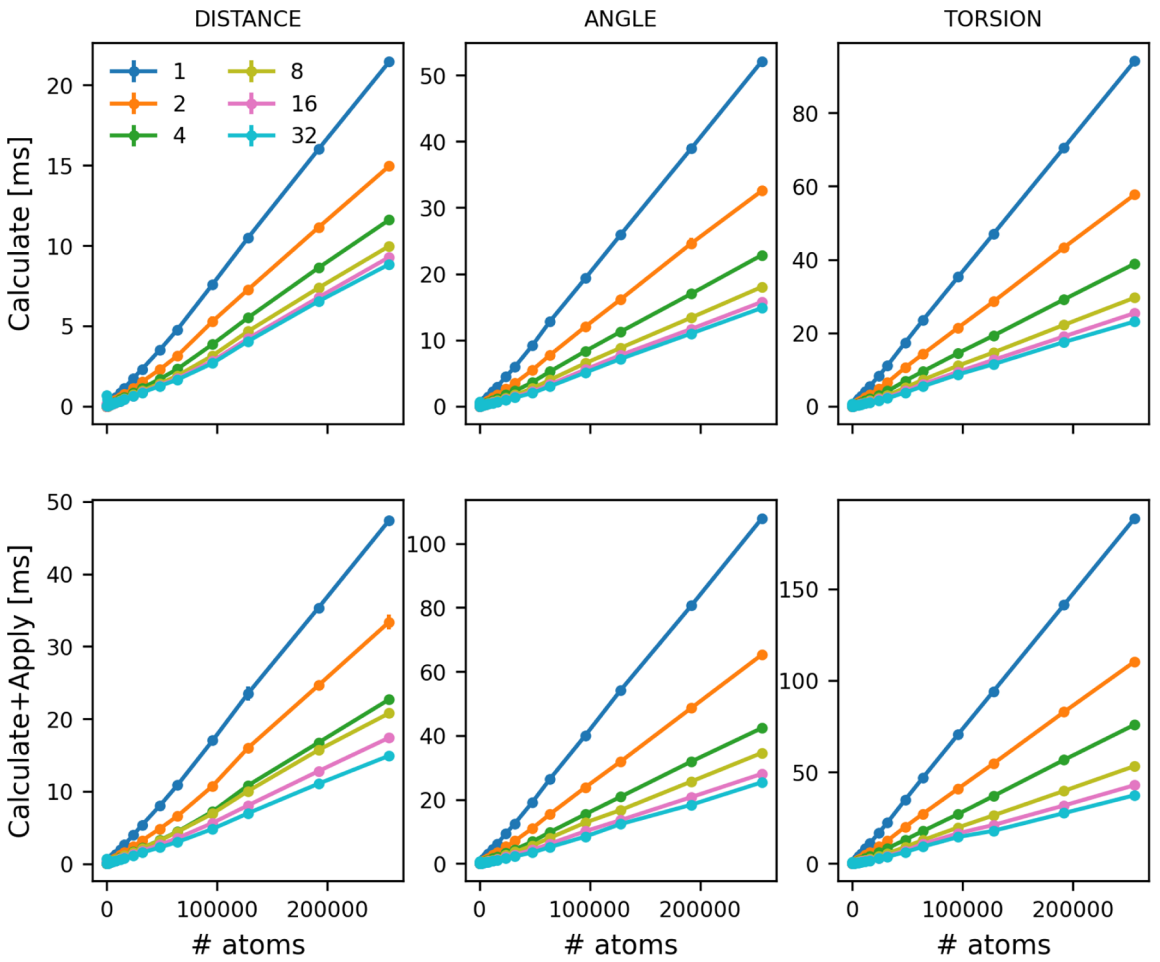


Figure 4. Time taken for a single PLUMED step as a function of the number of distances (left), angles (center) and torsions (right) that are being computed. Cost for just calculating these quantities (top panels). Cost for calculating and applying a force on the variables (bottom panels). Calculations were run on 1−32 OpenMP threads. The legend indicates what number of threads was used to produce each of the lines.

code is running using a combination of OpenMP and MPI it may be worth using the SERIAL flag in the PLUMED input to turn on off all MPI parallelism in PLUMED. Having completely separate instances of the PLUMED calculations on each of the MPI processes is often faster than dividing the calculations between the MPI processes and then communicating the data to all nodes.

7. WORKING WITH SYMMETRY FUNCTIONS

When studying phenomena such as crystal nucleation and growth using symmetry functions is commonplace.[8,61−63] These functions have the general form

$$s_i = \frac{1}{\sum_{j=1}^{N} \sigma(r_{ij})} \sum_{j=1}^{N} f(x_{ij}, y_{ij}, z_{ij}) \sigma(r_{ij}) \quad (1)$$

where (x_{ij}, y_{ij}, z_{ij}) is the vector connecting atoms i and j, r_{ij} is this vector's modulus and σ is a continuous switching function that is one when its argument is small and 0 when its argument is large. An example input that illustrates how such a function can be calculated using PLUMED is shown below:

```
# Calculate four NxN matrices called cmap.w, cmap.x, cmap.y and cmap.z
# Element ij of the matrix cmap.w is equal to sigma(r_ij)
# Element ij of the matrices cmap.x, cmap.y and cmap.z are
# equal to x_ij, y_ij and z_ij respectively.
cmap: CONTACT_MATRIX ...
  GROUP=@mdatoms SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
  COMPONENTS
...
# Calculate a matrix r whose ij element is equal to r_ij
r: CUSTOM ...
   ARG=cmap.x,cmap.y,cmap.z
   FUNC=sqrt(x*x+y*y+z*z)
   PERIODIC=NO
...
# Calculate a matrix called f whose ij element is equal to the
# quantity inside the sum of the numerator above.
f: CUSTOM ...
    ARG=cmap.w,cmap.x,cmap.y,cmap.z,r VAR=w,x,y,z,r
    FUNC=w*((x^4+y^4+z^4)/(r^4))
    PERIODIC=NO
...
# Evaluate the sum in the numerator of equation 1 by multiplying the matrix
# that is computed by the above action by a vector of ones.
ones: ONES SIZE=@natoms
numer: MATRIX_VECTOR_PRODUCT ARG=f,ones
# Evaluate the denominator in the expression above in a similar way
denom: MATRIX_VECTOR_PRODUCT ARG=cmap.w,ones
# And finally evaluate the values of the order parameter above
# for each of the atoms
op: CUSTOM ARG=numer,denom FUNC=x/y PERIODIC=NO
# Now calculate the mean of all the order parameters
mean: MEAN ARG=op PERIODIC=NO
# And add a bias
BIASVALUE ARG=mean
```

Notice that in inputs such as the one above we are now passing matrices between actions as well as scalars and vectors. We use red to distinguish the labels of matrices from the vectors and scalars. Further note that in the same way as we did for the FERRONEMATIC_ORDER parameter that was discussed in Section 2, we provide shortcuts that hide this

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

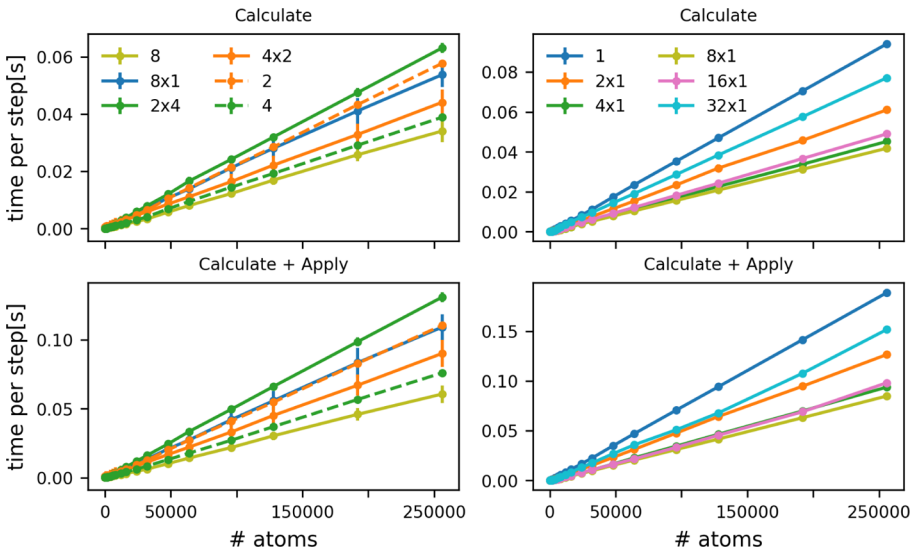


Figure 5. Time taken for a single PLUMED step as a function of the number of torsions that are being computed. The top panels show how the cost of calculating the torsions increases while the bottom panel shows how the cost of calculating the torsions and applying a force on these quantities changes. All the calculations that were used to generate the solid lines for graphs in the left column were run on 8 processors. For the light green line, all processors communicated via OpenMP, while the blue line shows the result that was obtained when communication between the 8 processors was managed using MPI. The dark green and orange lines show the results obtained when the two communication protocols are mixed. The orange line shows timings that are obtained by having four MPI processors that each run on two OpenMP threads, while the dark green line indicates the result that is obtained by having two MPI processors running on four OpenMP threads each. The orange and green dashed lines are results obtained when you run with 2 and 4 OpenMP threads, respectively. The lines on the graphs in the right column were obtained from calculations that were parallelized over 1 to 32 MPI processes.

complex input from casual users. Importantly, however, decomposing the calculation in the manner shown in the above input allows us to use the same or similar actions for many different symmetry functions. Furthermore, by optimizing these common actions we improve performance for many different symmetry function types.

The first and most important trick for optimizing this input is the use of the `D_MAX` parameter in the input to the switching function that is used in the `CONTACT_MATRIX` command. A `D_MAX` value can be set whenever you define a switching function in PLUMED. By setting this parameter of the switching function you are enforcing the value of the switching function to be zero for all $r > d_{\max}$ by using the stretching and scaling function that is computed from the switching function, θ, as follows.

$$\sigma(r_{ij}) = \frac{\theta(r_{ij}) - \theta(d_{\max})}{\theta(0) - \theta(d_{\max})} \tag{2}$$

Consequently, when we evaluate $\sigma(r_{ij})$ in the expression above we are not computing the usual rational switching function

$$\theta(r_{ij}) = \frac{1}{1 + \left(\frac{r_{ij}}{r_0}\right)^6} \tag{3}$$

$\sigma(r_{ij})$ is instead evaluated by inserting the value obtained for $\theta(r_{ij})$ from eq 3 into eq 2.

The fact that $\sigma(r_{ij})$ is guaranteed to be zero for all $r_{ij} > d_{\max}$ ensures that we can use the cell structures and linked list strategy that is discussed in[64] and illustrated in Figure 6 to optimize the calculation of the contact matrix. This strategy works by first dividing the simulation cell into cubic boxes that each have a side length of $d_{\max}$. The box each of the input atoms resides in is then identified. When we evaluate the ith row of the contact matrix we only evaluate element i, j if atom j is in the same box as atom i or one of the 26 boxes that are adjacent to the box that contains atom i. When `D_MAX` is

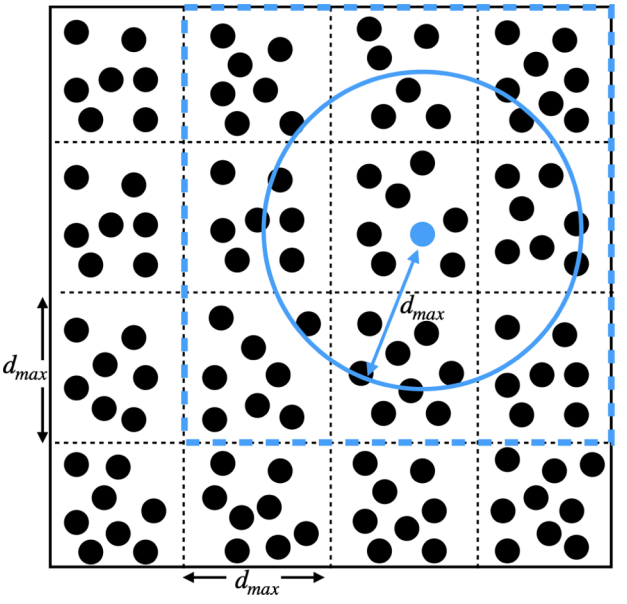


Figure 6. Cell structures and linked list technique that is introduced in ref 64 and employed within PLUMED. The cell box is divided into a set of cubes with a side length of $d_{\max}$. The cell each atom resides within is then determined at the start of the calculation. This reduces computational expense when we compute contact matrices because we can determine the neighbors of the blue atom by iterating over the atoms in the cell the blue atom resides in and the atoms in this cell's immediate neighbors. The distance between the blue atom and any atom that is not in the same cell or one of its neighbors is guaranteed to be greater than $d_{\max}$.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

used the calculation of the symmetry functions thus scales linearly and not quadratically with the number of atoms. Furthermore, because we are normally only interested in the structure in the first coordination sphere around the atoms, the D_MAX value can be set to a relatively small value. For the example calculations in this tutorial, which are run on an ideal simple cubic structure with a lattice parameter of 1 nm, D_MAX is set equal to 2 nm so the sum in eq 1 runs over the 18 atoms that are 1, $\sqrt{2}$ or $\sqrt{3}$ nm from the central atom. The boxes used in PLUMED are thus considerably smaller than those one would be using when exploiting similar tricks for evaluating the interatomic forces in an MD code.

Using D_MAX in the way described above also ensures that we can use sparse matrix storage for the **cmap.w**, **cmap.x**, **cmap.y**, **cmap.z**, **r** and **f** matrices in the above input and sparse matrix algebra when applying functions to the elements of the matrix in the CUSTOM actions and when multiplying these matrices by vectors in the MATRIX_VECTOR_PRODUCT actions to further improve performance.

When using an input such as the one above you can parallelize the calculation of all actions using OpenMP and MPI. Simulations were performed to determine how the time it takes PLUMED to perform a calculation using 1−32 OpenMP threads with the input above depends on the number of atoms that are input to the CONTACT_MATRIX command (Figure 7). You can see that adding a force on the symmetry functions increases the cost of the calculation by roughly a factor of 3. However, for the largest systems, using OpenMP reduces the cost of the calculation of the forces by a factor of 3. Even so, the cost of this calculation, when run with the large numbers of atoms that have been used here, is likely too great for it to be used as a CV in an MD simulation.

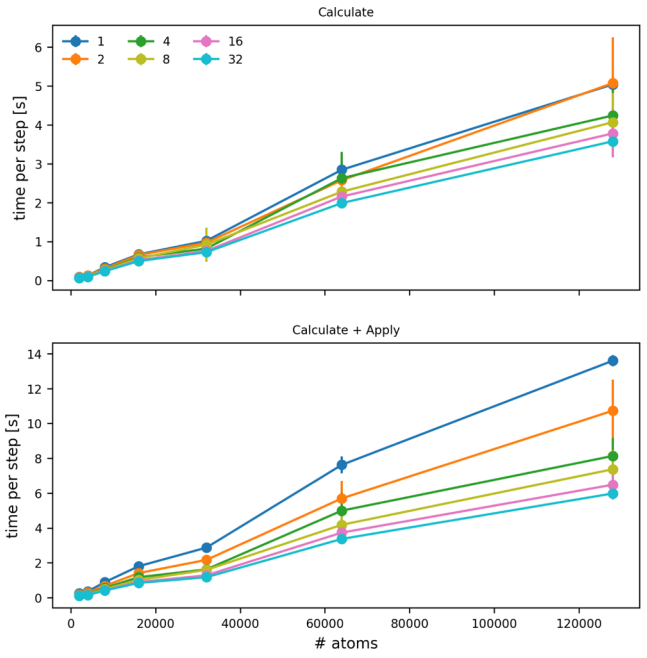


Figure 7. Cost of a single PLUMED step as a function of the number of atoms for which symmetry functions are being evaluated. The top panel shows how the cost of calculating the symmetry functions changes, while the bottom panel shows how the cost of calculating the symmetry function and applying a force upon them changes. The various lines show the costs when the calculation is run on the numbers of OpenMP threads indicated in the legend.

However, the implementation discussed here and the implementations of other expensive quantities discussed in this paper can be used to generate training data for a cheaper-to-evaluate neural network as discussed in.[65]

8. EVALUATING SYMMETRY FUNCTIONS IN A PARTICULAR PART OF THE BOX USING THE MASK KEYWORD

To reduce the computational expense associated with the calculation of symmetry functions some developers sometimes choose to only evaluate the values of symmetry functions for those atoms in a particular part of the box.[61] This approach makes particular sense if one is examining nucleation at a surface[66] or if one is using a restraint to prevent a seed from dissolving.[67] This approach would also be necessary if one were computing symmetry functions when using the methods for running at constant chemical potential discussed in.[68]

The problem when using such approaches is that the atoms within the region of interest, for which the symmetry function has to be evaluated, changes dynamically as the simulation progresses and atoms exchange in and out of the region of interest. We consequently need some way for dynamically indicating the set of atoms for which the symmetry functions need to be evaluated. The following example input illustrates how the MASK keyword can be used for precisely this purpose:

```
ones: ONES SIZE=@natoms
# Create an atom that is fixed at the origin
center: FIXEDATOM AT=0,0,0
# Determine if each of the atoms is within a sphere of radius 1.5 nm that is
# centered on the point (0,0,0)
w: INSPHERE ...
   ATOMS=@mdatoms CENTER=center
   RADIUS={RATIONAL D_0=24.9 R_0=0.01 D_MAX=25}
...
# Now evaluate the order parameters
cmap: CONTACT_MATRIX ...
 GROUP=@mdatoms SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
 COMPONENTS MASK=w
...
r: CUSTOM ...
  ARG=cmap.x,cmap.y,cmap.z
  FUNC=sqrt(x*x+y*y+z*z)
  PERIODIC=NO
...
f: CUSTOM ...
   ARG=cmap.w,cmap.x,cmap.y,cmap.z,r
   VAR=w,x,y,z,r
   FUNC=w*((x^4+y^4+z^4)/(r^4))
   PERIODIC=NO
...
numer: MATRIX_VECTOR_PRODUCT ARG=f,ones
denom: MATRIX_VECTOR_PRODUCT ARG=cmap.w,ones
# Evaluate the order parameter multiplied by the vector of ones and zeros
# that tells you whether or not each atom is in the region of interest
op: CUSTOM ARG=w,numer,denom FUNC=x*(y/z) PERIODIC=NO
opsum: SUM ARG=op PERIODIC=NO
vsum: SUM ARG=w PERIODIC=NO
# Evaluate the average value of the order parameter for those atoms that
# are in the region of interest.
mean: CUSTOM ARG=opsum,vsum FUNC=x/y PERIODIC=NO
BIASVALUE ARG=mean
```

The general form for the order parameter that is being evaluated here is given by the following expression

$$\xi = \frac{\sum_i w(x_i) s_i}{\sum_i w(x_i)}$$

In this expression s_i is the symmetry function that is defined in eq 1. $w(x_i)$ is then a differentiable function of the position, x_i, of atom i that is one if the atom is in the region of interest and zero otherwise. In the input above this $w(x_i)$ function is a switching function that acts upon the distance between atom i and the origin of the lab frame. The ith element of the vector, **w**, is thus one if x_i is within a sphere centered on the origin and zero otherwise.

The important thing to note in this input is that the vector **w** is passed to the CONTACT_MATRIX action through the

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

MASK keyword even though it is not needed to calculate the contact matrix. Passing this vector in this way ensures that the CONTACT_MATRIX only calculates the *i*th row of the matrix if the *i*th element in **w** is nonzero. In other words, by passing the vector **w** through the MASK keyword we ensure that s_i values are only computed for those atoms that are in the sphere of interest.

To determine the effect this trick has on computational performance we used PLUMED running on 16 OpenMP threads to calculate the average value of the symmetry function in a spherical subregion of a system of 120,000 atoms (Figure 8). The bottom *x*-axis in this figure indicates the radius of the sphere in which the symmetry function is being evaluated, while the top axis is then used indicate the number of atoms that are within a sphere of this radius. You can clearly see how the cost of the calculation is reduced as the radius of the spherical region of interest decreases and the number of atoms for which eq 1 is being evaluated decreases.

9. ANOTHER USE FOR THE MASK KEYWORD

The example provided in the previous section illustrates one application of a common approach for working with vectors and matrices whose elements are a product of a part that is cheap to evaluate and a part that is more expensive to evaluate. As illustrated above, if you are working with such objects you first evaluate the object with the computationally cheap elements and determine if any of the elements of this vector are zero. You then use the value from this cheap action as a mask that tells the more computationally expensive action that there are elements of its output that do not need to be computed.

Another place where this approach is used in PLUMED is in the implementation of the STRANDS_CUTOFF keyword in the ANTIBETARMSD and PARABETARMSD actions.[69] The following example is a typical input that uses this keyword:

```
MOLINFO STRUCTURE=protein.pdb
ab: ANTIBETARMSD ...
   RESIDUES=all TYPE=OPTIMAL
   STRANDS_CUTOFF=1.0 R_0=0.1 NN=8 MM=12
...
BIASVALUE ARG=ab
```

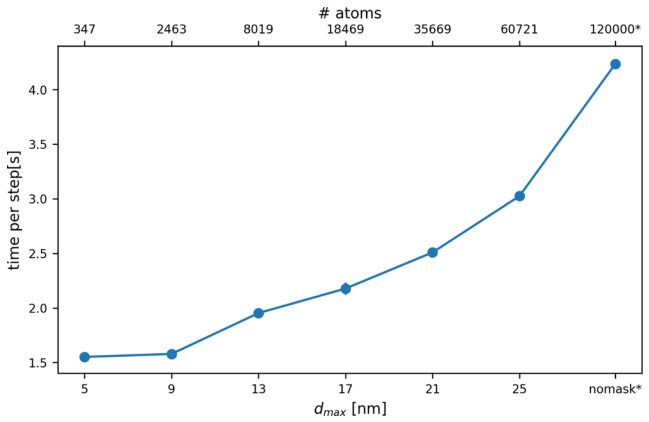


Figure 8. Cost of a single PLUMED step as a function of the volume of the spherical region in which you are evaluating symmetry functions for the atoms. The bottom *x*-axis indicates the radius of the spherical region in which the symmetry functions are being evaluated, while the top *x* axis indicates the number of atoms for which symmetry functions are being evaluated.

The ANTIBETARMSD command that is used here is an example of a shortcut action. When PLUMED reads this input it converts it into the input for a set of actions that together compute the ANTIBETARMSD collective variable which is defined as follows

$$s = \sum_i \frac{1 - \left(\frac{R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})}{r_0}\right)^8}{1 - \left(\frac{R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})}{r_0}\right)^{12}} \tag{4}$$

In this expression the sum runs over all the six residue segments of protein that could conceivably form an antiparallel β sheet and $\mathbf{X}_i$ is the positions of the 30 backbone atoms in each of these residue segments. $\mathbf{X}_{\text{ref}}$ is the positions of the 30 backbone atoms in an ideal antiparallel β sheet so $R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})$ is the root-mean-square deviation (RMSD) distance between the instantaneous configuration of the backbone atoms in the *i*th residue segment and the ideal structure for an antiparallel β sheet. The sum in eq 4 thus counts how many segments of the protein resemble an antiparallel β sheet.

PLUMED assumes that every pair of three residue segments that are separated by more than six residues along the chain can form an antiparallel beta sheets (Figure 10). This action is thus computationally expensive because number of potential antiparallel beta sheets scales quadratically with the number of residues.

The particular set of actions that are used to compute the ANTIBETARMSD collective variable function and the way the values are passed between them are shown in Figure 9. Notice the dashed line that connects the CUSTOM action with label **ab_cut** and the SECONDARY_STRUCTURE_RMSD action with label **ab_rmsd**. This line illustrates that the vector **ab_cut** is being used as a MASK in the second action. Each element in this vector is only equal to one if the distance between the C alpha atoms of the two central residues of the three-residue segments that we are comparing to an idealized antiparallel β sheet is less than a cutoff. If this distance is larger than the cutoff then the element is zero. Consequently, by using this vector as a mask on the SECONDARY_STRUCTURE_RMSD action we ensure that the expensive calculation of $R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})$ in eq 4 above is only performed for a small subset of the residues in the protein, which could potentially form an antiparallel β sheet.

To understand why this works consider the atoms involved in five of the $R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})$ values that we would have to calculate to obtain ANTIBETARMSD without STRANDS_CUTOFF that are shown in Figure 10. In evaluating eq 4 we need to consider whether the atoms in the blue rectangle form an antiparallel β sheet with each of the three-residue segments shown in the green and red rectangles. However, if we compute the distances between the yellow atom and each of the pink atoms we immediately see that the configurations formed by the residues in the blue rectangle and each of the red rectangles cannot possibly resemble an antiparallel β sheet as the central atoms in the residues are far too far apart. To compute eq 4 we thus only need to compute the two $R(\mathbf{X}_i, \mathbf{X}_{\text{ref}})$ values that involve the atoms in the blue rectangle and the atoms in the two green rectangles.

By using the STRANDS_CUTOFF keyword correctly we can improve the performance of the ANTIBETARMSD action by a factor of 2 for a small protein system with only 180

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

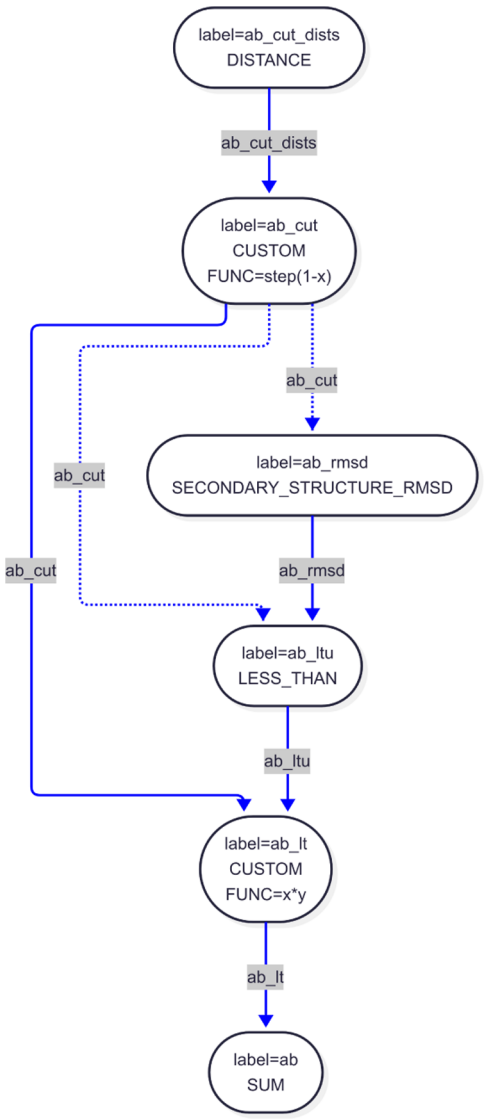


Figure 9. Actions used to evaluate secondary structure variables.

residues (Figure 11). We have plotted the performance of the version of this CV that was implemented in the paper where Pietrucci and Laio[69] first introduced this variable which used the DRMSD to measure the distances between the instantaneous and reference structure in Figure 11. A revised version of this CV that we implemented in PLUMED, that uses the RMSD in place of the DRMSD is also available. The RMSD version of this CV is slightly cheaper than the DRMSD version but the difference in cost is marginal.

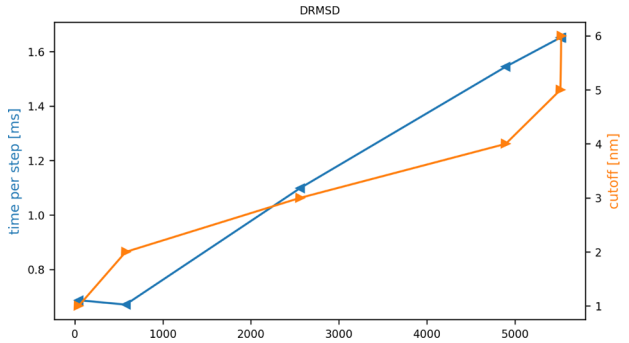


Figure 11. Lowering the STRANDS_CUTOFF value reduces the computational cost of the ANTIBETARMSD action. The orange line and right axis give the values we used for this cutoff. As discussed in the text, when you use a smaller STRANDS_CUTOFF you need to do fewer expensive RMSD or DRMSD calculations. The x-axis indicates how many of these DRMSD calculations are being performed for each cutoff. The blue line then shows how the time to perform these calculations changed as we increased the STRANDS_CUTOFF value. You can see that reducing this quantity to a reasonable value of 1.0 nm reduces the cost of the calculation by more than a factor of 2 as you move from needing to calculate over 5000 D/RMSD values to having to calculate less than 1000.

10. STEINHARDT PARAMETERS

Steinhardt parameters[4,70,71] are a key component of many approaches for studying nucleation in atomic systems. This approach is often claimed to be superior to the approach based on the symmetry function that was introduced earlier because it accounts for rotational invariance. These rotational invariances are accounted for by computing all $(2l + 1)$ spherical harmonics, Y_l^m, for a particular l value using

$$q_{lm}(i) = \sum_{j=1}^{N} \sigma(r_{ij}) Y_l^m(\theta_{ij}, \phi_{ij}) \quad \text{where} \quad \theta_{ij} = \cos^{-1}\left(\frac{z_{ij}}{r_{ij}}\right)$$

$$\text{and} \quad e^{i\phi_{ij}} = \frac{x_{ij}}{r_{ij}} + i\frac{y_{ij}}{r_{ij}}$$

Notice that this equation resembles eq 1, which was introduced in the section on symmetry functions. In writing it we have thus used the symbols that were defined when we introduced that earlier equation.

From these $(2l + 1)$ complex numbers one can then compute a modulus using

$$Q_l(i) = \frac{|q_l(i)|}{\sum_j \sigma(r_{ij})} \quad \text{where} \quad |q_l(i)| = \sqrt{\sum_{m=-l}^{l} q_{lm}(i)^* q_{lm}(i)} \tag{5}$$

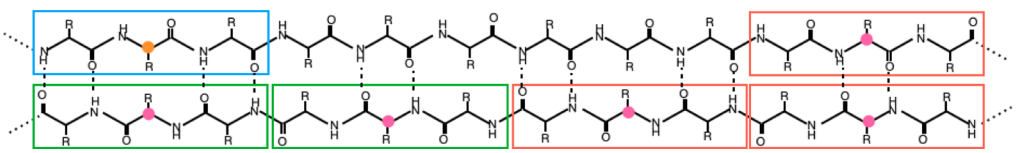


Figure 10. Method via which the STRANDS_CUTOFF keyword of ANTIBETARMSD improves the performance of this action. PLUMED needs to determine if the three residue segment in the blue rectangle forms an antiparallel β sheet with each of the three residues in each of the red and green rectangles by computing $R(\mathbf{X}_i, \mathbf{X}_{ref})$. However, before computing these $R(\mathbf{X}_i, \mathbf{X}_{ref})$ values, PLUMED computes the distance between the yellow atom and each of the pink atoms. The value of $R(\mathbf{X}_i, \mathbf{X}_{ref})$ is then only computed if this distance is less than the STRANDS_CUTOFF value. Consequently, we compute two $R(\mathbf{X}_i, \mathbf{X}_{ref})$ values rather than five values as the central atom in the three-residue segments that are in red rectangles are too far from the three-residue segment in the blue rectangle to possibly form an antiparallel β sheet.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

for each of the individual atoms. Alternatively, one can compute the following product of the $(2l + 1)$ spherical harmonics on atoms i and j.[72]

$$Q(i,j) = \sum_{m=-l}^{l} \left(\frac{q_{lm}(i)}{|q_l(i)|} \right)^* \left(\frac{q_{lm}(j)}{|q_l(j)|} \right) \tag{6}$$

The advantage of this second approach over the first is that a comparison of the orientations of the coordination spheres around atoms i and j is performed. In other words, the dot product that is evaluated in the expression above is only large if the same $q_{lm}(i)$ and $q_{lm}(j)$ values are large on both atom i and atom j. This second approach is thus less affected if there are a significant number of $q_{lm}(i)$ components with moderately large values than the first.

The example input below illustrates how we can use PLUMED to calculate the average $Q_6(i)$ value for all the atoms in the system.

```
q6: Q6 SPECIES=@mdatoms SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
mean: MEAN ARG=q6 PERIODIC=NO
BIASVALUE ARG=mean
```

The input here is a shortcut once again. An expanded version that does something similar to this shortcut is provided in the next example input below. We ran calculations to determine how the computational expense associated with evaluating this variable changes with system size (Figure 12). A comparison of Figures 7 and 12 illustrates that computing Q_6 is roughly six times more expensive than computing a symmetry function.

As discussed above, order parameters that use eq 6 in place of eq 5 usually demonstrate superior performance at distinguishing between atoms that are part of the solid and liquid phases. A typical order parameter for an atom that is computed using eq 6 is given by

$$s_i = \frac{\sum_{j=1}^{n} \sigma(r_{ij}) Q(i,j)}{\sum_{j=1}^{n} \sigma(r_{ij})} \tag{7}$$

To compute the average value of this order parameter for all the atoms in the system using PLUMED we would use the following input.

```
# Compute sigma(r_ij) and the (x_ij,y_ij,z_ij) vectors
q6mat: CONTACT_MATRIX ...
 GROUP=@mdatoms COMPONENTS
 SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
...
# Evaluate the Y_l^m values for each bond
q6sh: SPHERICAL_HARMONIC ARG=q6mat.x,q6mat.y,q6mat.z,q6mat.w L=6
# Calculate the vector of q_lm(i) values
aones: ONES SIZE=@natoms
q6sp: MATRIX_VECTOR_PRODUCT ARG=q6sh.*,aones
# Evaluate the |q_l(i)| values
q6norm2: COMBINE ...
  PERIODIC=NO ARG=q6sp.*
  POWERS=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
...
q6norm: CUSTOM ARG=q6norm2 FUNC=sqrt(x) PERIODIC=NO
# Construct a matrix in which the ith row contains the set of q_lm(i) values
vecs: VSTACK ARG=q6sp.*
# Divide each of the q_lm(i) values in the matrix we just calculated by |q_l(i)|
ones: ONES SIZE=26
normmat: OUTER_PRODUCT ARG=q6norm,ones
uvecs: CUSTOM ARG=vecs,normmat FUNC=x/y PERIODIC=NO
# Calculate a matrix containing all the Q(i,j) values
# Notice that we use a MASK here so Q(i,j) is not calculated
# if atoms i and j are further apart than d_max
uvecsT: TRANSPOSE ARG=uvecs
dpmat: MATRIX_PRODUCT ARG=uvecs,uvecsT MASK=q6mat.w
prod: CUSTOM ARG=q6mat.w,dpmat FUNC=x*y PERIODIC=NO
# Evaluate the numerator in the expression for s_i above
numer: MATRIX_VECTOR_PRODUCT ARG=prod,aones
# Evaluate the coordination numbers
denom: MATRIX_VECTOR_PRODUCT ARG=q6mat.w,aones
# These are the s_i values
lq6: CUSTOM ARG=numer,denom FUNC=x/y PERIODIC=NO
mean: MEAN ARG=lq6 PERIODIC=NO
BIASVALUE ARG=mean
```

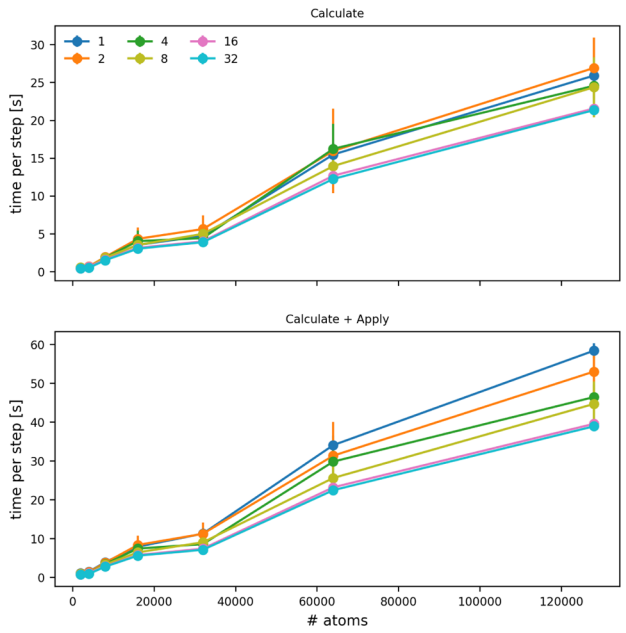


Figure 12. Cost of a single PLUMED step as a function of the number of atoms for which Q_6 parameters are being evaluated. The top panel shows how the cost of calculating the Q_6 parameters changes, while the bottom panel shows how the cost of calculating Q_6 and applying a force upon it changes. The various lines show the costs when the calculation is run on the numbers of OpenMP threads indicated in the legend.

In previous versions of PLUMED the computational expense associated with doing the calculations in the input above was much greater than what is required to compute eq 5. However, a comparison between Figures 12 and 13 shows that there is no longer a large difference in the cost associated with computing eqs 5 and 7. In other words, there is no longer a large computational penalty if you compute eq 7 instead of 5.

Computing eq 7 was expensive in earlier versions of PLUMED because the distances that are used in the CONTACT_MATRIX action were computed twice. These calculations are still done twice if you use the LOCAL_Q6 shortcut that is provided within PLUMED to maintain the older syntax for this command. By breaking up the various actions within PLUMED and allowing one to reuse expensive-to-compute values computed in one action in other parts of the input we have also improved the performance of the code.

Now suppose that you want to compute the average value of the quantity that is defined in eq 7 for the subset of atoms that are in a particular part of the box. We cannot use the INSPHERE action in the input to the MASK keyword for the CONTACT_MAP action with label **cmap** from the above input in the way that was illustrated in Section 8 because, as illustrated in Figure 14, we need to evaluate $q_{lm}(i)$ values for atoms that are not within the region of interest in order to evaluate eq 7 for all the atoms in the region of interest.

We can still use a MASK to reduce the expense of this calculation as is illustrated by the input that is shown below:

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

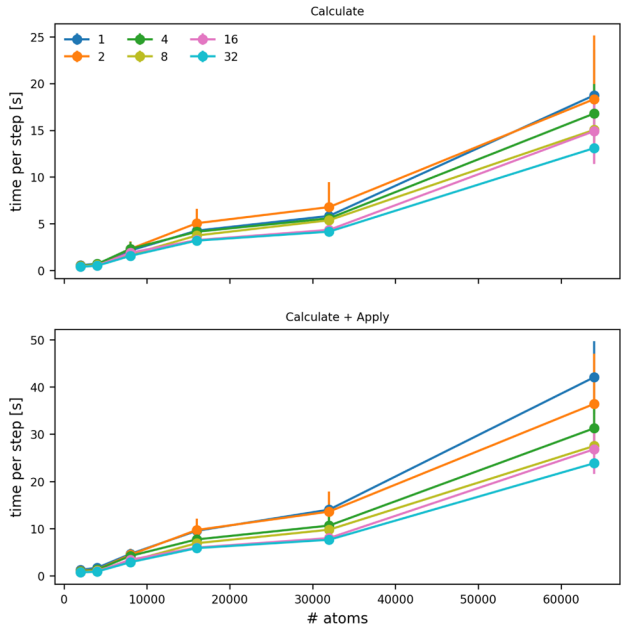


Figure 13. Cost of a single PLUMED step as a function of the number of atoms for which the quantity defined in eq 7 is being computed. The top panel shows how the cost of calculating this quantity changes, while the bottom panel shows how the cost of calculating this quantity and applying a force upon it changes. The various lines show the costs when the calculation is run on the numbers of OpenMP threads indicated in the legend.

```
aones: ONES SIZE=@natoms
# Create an atom that is fixed at the origin
center: FIXEDATOM AT=0,0,0
# Determine if each of the atoms is within a sphere of radius 1.5 nm that is
# centered on the point (0,0,0)
w: INSPHERE ...
   ATOMS=@mdatoms CENTER=center
   RADIUS={RATIONAL D_0=15.9 R_0=0.01 D_MAX=16}
...
# Compute the contact matrix for the atoms that are within the sphere
lq6mat: CONTACT_MATRIX ...
 GROUP=@mdatoms
 SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
 MASK=w
...
# Construct a matrix in which every row is equal to the vector w
volmat: OUTER_PRODUCT ARG=w,aones MASK=lq6mat
# Element i,j of this matrix is non-zero if r_ij is less than d_max
# and atom i is in the region of interest.
bondsmat: CUSTOM ARG=lq6mat,volmat FUNC=x*y PERIODIC=NO
# Transposing the matrix above and multiplying it by a vector of ones
# results in a vector in which element i is only non zero if there is a bond
# between it and one of the atoms in the region of interest.
bondsmatT: TRANSPOSE ARG=bondsmat
bonds: MATRIX_VECTOR_PRODUCT ARG=bondsmatT,aones
# The set of atoms for which we need to compute q_lm(i) values includes
# the atoms in the region of interest and the the atoms that are bonded
# to atoms in the region of interest.
q6mask: CUSTOM ARG=bonds,w FUNC=x+y PERIODIC=NO
# In the next few lines we use the q6mask value in the input to the MASK
# keyword so we only calculate the q_lm(i)/|q_l(i)| values using the method
# described in the previous input for the subset of atoms that are required.
q6mat:  CONTACT_MATRIX ...
 GROUP=@mdatoms  COMPONENTS
 SWITCH={RATIONAL D_0=0.6 R_0=1 NN=6 MM=12 D_MAX=2.0}
 MASK=q6mask
...
q6sh: SPHERICAL_HARMONIC ARG=q6mat.x,q6mat.y,q6mat.z,q6mat.w L=6
q6sp: MATRIX_VECTOR_PRODUCT ARG=q6sh.*,aones
q6norm2: COMBINE ...
  PERIODIC=NO ARG=q6sp.* MASK=q6mask
  POWERS=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
...
q6norm: CUSTOM ARG=q6norm2 FUNC=sqrt(x) MASK=q6mask PERIODIC=NO
vecs: VSTACK ARG=q6sp.*
ones: ONES SIZE=26
normmat: OUTER_PRODUCT ARG=q6norm,ones
uvecs: CUSTOM ARG=vecs,normmat FUNC=x/y PERIODIC=NO
```

As you have to evaluate two CONTACT_MATRIX actions in the above input there will be cases where simply computing the values of s_i using eq 7 for all the atoms in the system and then computing the average value of this quantity in the region of interest without using the MASK keyword at all is computationally cheaper than using the input above. We thus ran calculations to determine how the length of time required to perform the calculation using 16 OpenMP threads for a system of 30,000 atoms depends on the radius of the spherical region (Figure 15). The bottom x-axis in this figure indicates the radius of the sphere in which eq 7 is being evaluated. The top axis then shows how many atoms we need to evaluate eq 5 for in order to evaluate eq 7 for the atoms in the spherical region as well as the number of atoms in the sphere. The result is as you would expect; namely, when the sphere is large the computational expense associated with calculating the two contact matrices ensures that using the input above is slower than simply calculating eq 5 for all atoms and averaging. However, when the sphere is smaller the reduction in computational cost that is associated with evaluating eq 5 for a smaller number of atoms easily makes up for the cost that is added by evaluating the contact matrix twice. It is thus considerably more computationally efficient to use the input above whenever the volume of interest is small.

10.1. Additional Tips

Having discussed some of the most recent optimizations added to the PLUMED codebase, we here report a checklist of other performance-optimization ideas that have been implemented and, that are discussed in previous papers or online tutorials:

- When using metadynamics you should employ the implementation of the bias that stores the potential on a grid. You do this by employing the GRID_MIN, GRID_MAX and GRID_BIN keywords in your METAD command as discussed in.[73] Using a grid in these calculations prevents the cost associated with evaluating the bias potential from increasing in the later parts of the trajectory when more Gaussian functions have been added.[74]
- At the time of writing, some collective variables in PLUMED use a standard neighbor list rather than the cell structures and linked list strategy from ref 64 A

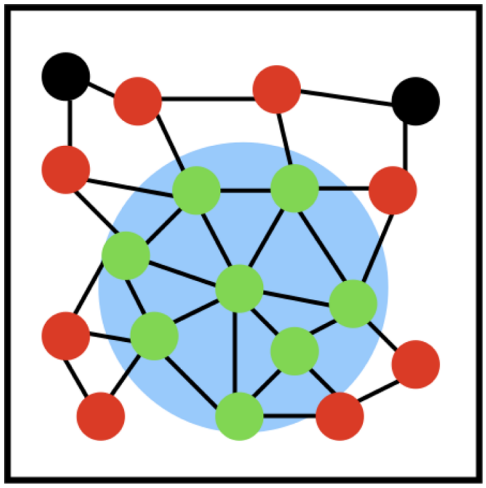


Figure 14. Optimizing the evaluation local Q6 parameters (eq 7) in a sphere. The lines indicate pairs of atoms that are within d_{max} of each other. The green circles are the atoms that are within the blue region and for which we need to evaluate eq 7. To evaluate eq 7 for these atoms we need to evaluate q_{lm} values for all the atoms that are shown in green and all the atoms shown in red that are within d_{max} of a green atom. Many of the atoms shown in red, for which we need to evaluate q_{lm}, are not within the blue circle.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

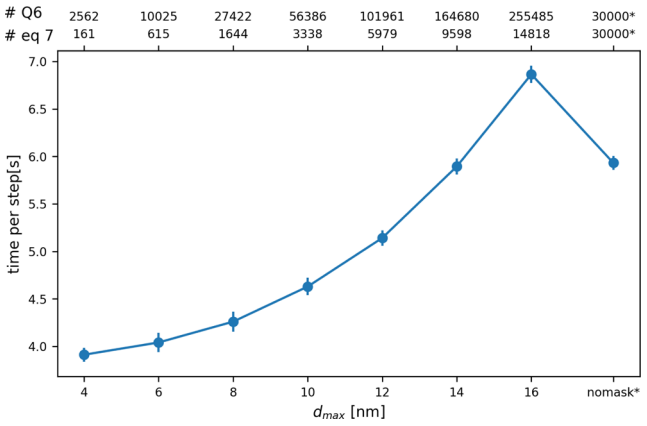


Figure 15. Cost of a single PLUMED step as a function of the radius of the spherical region in which the quantity defined in eq 7 is being calculated. The bottom x-axis indicates the radius of the spherical region in which eq 7 is being evaluated, while the numbers labeled #eq 7 on the top x axis indicate the number of atoms that are within it. The numbers labeled #Q6 are the number of atoms for which eq 5 must be evaluated.

notable example is `COORDINATION`. Typically the keywords `NL_CUTOFF` and `NL_STRIDE` are used to turn on the neighbor list. If you want to use this feature you will need to optimize the neighbor list parameters for both speed and correctness as discussed in.[73]

- Some biasing potentials act on collective variables that have a smooth dependence on the atomic coordinates. When using such bias potentials you can use the multiple-time-step implementation discussed in.[73,75]
- Some actions in PLUMED are able to modify global coordinates. Examples include `WHOLEMOLECULES` and `FIT_TO_TEMPLATE`. For these actions, PLUMED cannot track dependencies in an optimal way. This means that you should carefully choose the `STRIDE` parameters for these actions to have correct results and to minimize their impact on the overall performances.

11. CONCLUSION

In this tutorial, we have demonstrated how to perform reliable and reproducible benchmarks of PLUMED's performance using the recently introduced `plumed benchmark` tool. We have used this tool to present a series of benchmarks covering a diverse set of applications, from simple scalar quantities to more complex collective variables such as symmetry functions and Steinhardt parameters. These examples illustrate how performance can be optimized by employing vector-based operations, linked-list algorithms, and appropriate parallelization strategies.

We encourage developers who contribute new functionalities to PLUMED to follow a similar benchmarking approach. Providing benchmarks alongside contributed code not only helps ensure performance portability and transparency but also facilitates meaningful comparisons between implementations across different hardware and software environments. Performing these comparisons will become increasingly important as functionality is ported from the CPU to GPUs using frameworks such as CUDA, OpenACC, ArrayFire and SYCL. We also invite developers to explore alternative implementations of the vectorized calculations discussed here, for instance using emerging numerical frameworks such as JAX,[76] PyTorch,[76] or TensorFlow,[77] and to report and share their benchmarking results with the community. Additional development work and careful benchmarking would likely result in further improvements for all the methods discussed in this tutorial. Our hope is that by providing sufficient detail for readers to reimplement these functionalities elsewhere and benchmark new implementations against our own, we embrace the competitive and collaborative spirit that has always driven the best scientific software development—one that values both innovation and rigorous evaluation in equal measure.

Looking forward, we envision that benchmarking could be further integrated into PLUMED's development workflow. Automated benchmarking pipelines could regularly assess performance across multiple PLUMED versions and hardware configurations, generating plots similar to those shown here and enabling continuous monitoring of performance evolution. Such a system would not only streamline performance testing but also strengthen PLUMED's role as a transparent and reproducible platform for method development in molecular simulations.

AUTHOR INFORMATION

Corresponding Authors

Massimiliano Bonomi − *Institut Pasteur, Université Paris Cité, CNRS UMR3528, Computational Structural Biology Unit, Paris, France,* https://research.pasteur.fr/en/; orcid.org/0000-0002-7321-0004; Email: mbonomi@pasteur.fr

Carlo Camilloni − *Department of Biosciences, University of Milano, Milano 20133, Italy;* orcid.org/0000-0002-9923-8590; Email: carlo.camilloni@unimi.it

Giovanni Bussi − *Scuola Internazionale Superiore di Studi Avanzati (SISSA), Trieste 34136, Italy;* orcid.org/0000-0001-9216-5782; Email: bussi@sissa.it

Gareth A. Tribello − *Centre for Quantum Materials and Technologies, School of Mathematics and Physics, Queen's University Belfast, Belfast BT7 1NN, U.K.;* orcid.org/0000-0002-4763-9317; Email: g.tribello@qub.ac.uk

Author

Daniele Rapetti − *Scuola Internazionale Superiore di Studi Avanzati (SISSA), Trieste 34136, Italy;* orcid.org/0000-0002-7193-5220

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jpcb.5c07562

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

M.B. would like to acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101086685—bAIes). D.R. and G.B. acknowledge the Italian National Centre for HPC, Big Data, and Quantum Computing (grant No. CN00000013), founded within the Next Generation EU initiative. Lastly, all the authors thank Chris Chipot for the patience he demonstrated in editing this paper.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

REFERENCES

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

(2) The PLUMED consortium. Promoting transparency and reproducibility in enhanced molecular simulations. *Nat. Methods* **2019**, *16*, 670−673.

(3) Tribello, G. A.; Bonomi, M.; Bussi, G.; Camilloni, C.; Armstrong, B. I.; Arsiccio, A.; Aureli, S.; Ballabio, F.; Bernetti, M.; Bonati, L.; et al. PLUMED Tutorials: A collaborative, community-driven learning ecosystem. *J. Chem. Phys.* **2025**, *162*, 092501.

(4) Tribello, G. A.; Giberti, F.; Sosso, G. C.; Salvalaglio, M.; Parrinello, M. Analyzing and Driving Cluster Formation in Atomistic Simulations. *J. Chem. Theory Comput.* **2017**, *13*, 1317−1327.

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

(6) Klug, J.; Triguero, C.; Del Pópolo, M. G.; Tribello, G. A. Using Intrinsic Surfaces To Calculate the Free-Energy Change When Nanoparticles Adsorb on Membranes. *J. Phys. Chem. B* **2018**, *122*, 6417−6422.

(7) Baldi, E.; Ceriotti, M.; Tribello, G. A. Extracting the Interfacial Free Energy and Anisotropy from a Smooth Fluctuating Dividing Surface. *J. Phys.:Condens. Matter* **2017**, *29*, 445001.

(8) Cheng, B.; Ceriotti, M.; Tribello, G. A. Classical nucleation theory predicts the shape of the nucleus in homogeneous solidification. *J. Chem. Phys.* **2020**, *152*, 044103.

(9) Santiso, E. E.; Trout, B. L. A general set of order parameters for molecular crystals. *J. Chem. Phys.* **2011**, *134*, 064109.

(10) Tribello, G. A.; Gasparotto, P. *Biomolecular Simulations: Methods and Protocols*; Bonomi, M., Camilloni, C., Eds.; Springer New York: New York, NY, 2019; pp 453−502.

(11) Tribello, G. A.; Gasparotto, P. Using Dimensionality Reduction to Analyze Protein Trajectories. *Front. Mol. Biosci.* **2019**, *6*, 46.

(12) Chen, H.; Fu, H.; Shao, X.; Chipot, C.; Cai, W. E. L. F. An Extended-Lagrangian Free Energy Calculation Module for Multiple Molecular Dynamics Engines. *J. Chem. Inf. Model.* **2018**, *58*, 1315−1318.

(13) Hocky, G. M.; Dannenhoffer-Lafage, T.; Voth, G. A. Coarse-Grained Directed Simulation. *J. Chem. Theory Comput.* **2017**, *13*, 4593−4603.

(14) Piaggi, P. M.; Parrinello, M. Calculation of phase diagrams in the multithermal-multibaric ensemble. *J. Chem. Phys.* **2019**, *150*, 244119.

(15) Hartmann, M. J.; Singh, Y.; Vanden-Eijnden, E.; Hocky, G. M. Infinite switch simulated tempering in force (FISST). *J. Chem. Phys.* **2020**, *152*, 244120.

(16) Raniolo, S.; Limongelli, V. Ligand binding free-energy calculations with funnel metadynamics. *Nat. Protoc.* **2020**, *15*, 2837−2866.

(17) Bonomi, M.; Camilloni, C. Integrative structural and dynamical biology with PLUMED-ISDB. *Bioinformatics* **2017**, *33*, 3999−4000.

(18) Morishita, T.; Nakamura, T.; Shinoda, W.; Ito, A. M. Isokinetic approach in logarithmic mean-force dynamics for on-the-fly free energy reconstruction. *Chem. Phys. Lett.* **2018**, *706*, 633−640.

(19) Di Bartolo, A. L.; Masone, D. Synaptotagmin-1 C2B domains cooperatively stabilize the fusion stalk via a master-servant mechanism. *Chem. Sci.* **2022**, *13*, 3437−3446.

(20) Invernizzi, M.; Parrinello, M. Exploration vs Convergence Speed in Adaptive-Bias Enhanced Sampling. *J. Chem. Theory Comput.* **2022**, *18*, 3988−3996.

(21) Gasparotto, P.; Meißner, R. H.; Ceriotti, M. Recognizing Local and Global Structural Motifs at the Atomic Scale. *J. Chem. Theory Comput.* **2018**, *14*, 486−498.

(22) Pipolo, S.; Salanne, M.; Ferlat, G.; Klotz, S.; Saitta, A. M.; Pietrucci, F. Navigating at Will on the Water Phase Diagram. *Phys. Rev. Lett.* **2017**, *119*, 245701.

(23) Bonati, L.; Rizzi, V.; Parrinello, M. Data-Driven Collective Variables for Enhanced Sampling. *J. Phys. Chem. Lett.* **2020**, *11*, 2998−3004.

(24) Bonati, L.; Piccini, G.; Parrinello, M. Deep learning the slow modes for rare events sampling. *Proc. Natl. Acad. Sci.* **2021**, *118*, No. e2113533118.

(25) Palazzesi, F.; Valsson, O.; Parrinello, M. Conformational Entropy as Collective Variable for Proteins. *J. Phys. Chem. Lett.* **2017**, *8*, 4752−4756.

(26) Gigli, L.; Goscinski, A.; Ceriotti, M.; Tribello, G. A. Modeling the ferroelectric phase transition in barium titanate with DFT accuracy and converged sampling. *Phys. Rev. B* **2024**, *110*, 024101.

(27) Pietrucci, F.; Andreoni, W. Graph Theory Meets Ab Initio Molecular Dynamics: Atomic Structures and Transformations at the Nanoscale. *Phys. Rev. Lett.* **2011**, *107*, 085504.

(28) Valsson, O.; Parrinello, M. Variational Approach to Enhanced Sampling and Free Energy Calculations. *Phys. Rev. Lett.* **2014**, *113*, 090601.

(29) Giberti, F.; Cheng, B.; Tribello, G. A.; Ceriotti, M. Iterative Unbiasing of Quasi-Equilibrium Sampling. *J. Chem. Theory Comput.* **2020**, *16*, 100−107.

(30) Giberti, F.; Tribello, G. A.; Ceriotti, M. Global Free-Energy Landscapes as a Smoothly Joined Collection of Local Maps. *J. Chem. Theory Comput.* **2021**, *17*, 3292−3308.

(31) Hoff, S. E.; Thomasen, F. E.; Lindorff-Larsen, K.; Bonomi, M. Accurate model and ensemble refinement using cryo-electron microscopy maps and Bayesian inference. *PLoS Comput. Biol.* **2024**, *20*, No. e1012180.

(32) Schnapka, V.; Morozova, T. I.; Sen, S.; Bonomi, M. Atomic resolution ensembles of intrinsically disordered proteins with Alphafold. *Nat. Commun.* **2026**, DOI: 10.1038/s41467-026-69172-y.

(33) Panei, F. P.; Gkeka, P.; Bonomi, M. Identifying small-molecules binding sites in RNA conformational ensembles with SHAMAN. *Nat. Commun.* **2024**, *15*, 5725.

(34) Hsu, W.-T.; Piomponi, V.; Merz, P. T.; Bussi, G.; Shirts, M. R. Alchemical metadynamics: Adding alchemical variables to metadynamics to enhance sampling in free energy calculations. *J. Chem. Theory Comput.* **2023**, *19*, 1805−1817.

(35) Frohlking, T.; Mlynsky, V.; Janecek, M.; Kuhrová, P.; Krepl, M.; Banás, P.; Sponer, J.; Bussi, G. Automatic learning of hydrogen-bond fixes in the AMBER RNA force field. *J. Chem. Theory Comput.* **2022**, *18*, 4490−4502.

(36) Cesari, A.; Gil-Ley, A.; Bussi, G. Combining simulations and solution experiments as a paradigm for RNA force field refinement. *J. Chem. Theory Comput.* **2016**, *12*, 6192−6200.

(37) Bottaro, S.; Banas, P.; Sponer, J.; Bussi, G. Free energy landscape of GAGA and UUCG RNA tetraloops. *J. Phys. Chem. Lett.* **2016**, *7*, 4032−4038.

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

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

(40) Eastman, P.; et al. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *J. Phys. Chem. B* **2024**, *128*, 109−116.

(41) Coretti, A.; Bacon, C.; Berthin, R.; Serva, A.; Scalfi, L.; Chubak, I.; Goloviznina, K.; Haefele, M.; Marin-Laflèche, A.; Rotenberg, B.; Bonella, S.; Salanne, M. MetalWalls Simulating electrochemical interfaces between polarizable electrolytes and metallic electrodes. *J. Chem. Phys.* **2022**, *157*, 184801.

(42) Giorgino, T. PLUMED-GUI: An environment for the interactive development of molecular dynamics analysis and biasing scripts. *Comput. Phys. Commun.* **2014**, *185*, 1109−1114.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX

(43) Case, D. A.; et al. AmberTools. *J. Chem. Inf. Model.* **2023**, *63*, 6183−6191.
(44) Case, D. A.; et al. Recent Developments in Amber Biomolecular Simulations. *J. Chem. Inf. Model.* **2025**, *65*, 7835−7843.
(45) Kühne, T. D.; et al. CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. *J. Chem. Phys.* **2020**, *152*, 194103.
(46) Giannozzi, P.; et al. Advanced capabilities for materials modelling with QUANTUM ESPRESSO. *J. Phys.:Condens. Matter* **2017**, *29*, 465901.
(47) Giannozzi, P.; et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys.:Condens. Matter* **2009**, *21* (19pp), 395502.
(48) Giannozzi, P.; Baseggio, O.; Bonfà, P.; Brunato, D.; Car, R.; Carnimeo, I.; Cavazzoni, C.; de Gironcoli, S.; Delugas, P.; Ferrari Ruffino, F.; Ferretti, A.; Marzari, N.; Timrov, I.; Urru, A.; Baroni, S. Quantum ESPRESSO toward the exascale. *J. Chem. Phys.* **2020**, *152*, 154105.
(49) Devereux, H. L.; Cockrell, C.; Elena, A. M.; Bush, I.; Chalk, A. B. G.; Madge, J.; Scivetti, I.; Wilkins, J. S.; Todorov, I. T.; Smith, W.; Trachenko, K. DL_POLY 5: Calculation of system properties on the fly for very large systems via massive parallelism. *arXiv* **2025**, arXiv.2503.07526.
(50) Kapil, V.; et al. i-PI 2.0: A universal force engine for advanced molecular simulations. *Comput. Phys. Commun.* **2019**, *236*, 214−223.
(51) Hourahine, B.; et al. Recent Developments in DFTB+, a Software Package for Efficient Atomistic Quantum Mechanical Simulations. *J. Phys. Chem. A* **2025**, *129*, 5373−5390.
(52) Hjorth Larsen, A.; et al. The atomic simulation environment—a Python library for working with atoms. *J. Phys.:Condens. Matter* **2017**, *29*, 273002.
(53) Fan, Z.; Wang, Y.; Ying, P.; Song, K.; Wang, J.; Wang, Y.; Zeng, Z.; Xu, K.; Lindgren, E.; Rahm, J. M.; et al. GPUMD: A package for constructing accurate machine-learned potentials and performing highly efficient atomistic simulations. *J. Chem. Phys.* **2022**, *157*, 114801.
(54) Phillips, J. C.; Hardy, D. J.; Maia, J. D. C.; Stone, J. E.; Ribeiro, J. V.; Bernardi, R. C.; Buch, R.; Fiorin, G.; Hénin, J.; Jiang, W.; et al. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *J. Chem. Phys.* **2020**, *153*, 044130.
(55) Swenson, D. W. H.; Prinz, J.-H.; Noe, F.; Chodera, J. D.; Bolhuis, P. G. OpenPathSampling: A Python Framework for Path Sampling Simulations. 1. Basics. *J. Chem. Theory Comput.* **2019**, *15*, 813−836.
(56) Doerr, S.; Harvey, M. J.; Noé, F.; De Fabritiis, G. HTMD: High-Throughput Molecular Dynamics for Molecular Discovery. *J. Chem. Theory Comput.* **2016**, *12*, 1845−1852.
(57) Eastman, P. Lepton Mathematical Expression Parser. https://simtk.org/projects/lepton, 2014 (accessed 10 07 2025).
(58) Eastman, P.; Pande, V. *GPU Computing Gems Jade, Edition.*; mei, W., Hwu, W., Eds.; *Applications of GPU Computing Series*; Morgan Kaufmann: Boston, 2012; pp 399−407.
(59) Eppenga, R.; Frenkel, D. Monte Carlo study of the isotropic and nematic phases of infinitely thin hard platelets. *Mol. Phys.* **1984**, *52*, 1303−1334.
(60) Efron, B. Bootstrap Methods: Another Look at the Jackknife. *Ann. Stat.* **1979**, *7*, 1−26.
(61) Angioletti-Uberti, S.; Ceriotti, M.; Lee, P. D.; Finnis, M. W. Solid-liquid interface free energy through metadynamics simulations. *Phys. Rev. B* **2010**, *81*, 125416.
(62) Cheng, B.; Tribello, G. A.; Ceriotti, M. Solid-liquid interfacial free energy out of equilibrium. *Phys. Rev. B* **2015**, *92*, 180102.
(63) Cheng, B.; Tribello, G. A.; Ceriotti, M. The Gibbs free energy of homogeneous nucleation: From atomistic nuclei to the planar limit. *J. Chem. Phys.* **2017**, *147*, 104707.
(64) Allen, M. P.; Tildesley, D. J. *Computer simulation of liquids*; Clarendon Press: USA, 1989.
(65) Dietrich, F. M.; Advincula, X. R.; Gobbo, G.; Bellucci, M. A.; Salvalaglio, M. Machine Learning Nucleation Collective Variables with Graph Neural Networks. *J. Chem. Theory Comput.* **2024**, *20*, 1600−1611.
(66) Sosso, G. C.; Li, T.; Donadio, D.; Tribello, G. A.; Michaelides, A. Microscopic Mechanism and Kinetics of Ice Formation at Complex Interfaces: Zooming in on Kaolinite. *J. Phys. Chem. Lett.* **2016**, *7*, 2350−2355.
(67) Blow, K. E.; Sosso, G. C.; Quigley, D. You reap what you sow: On the impact of nuclei morphology on seeded molecular dynamics simulations. *J. Chem. Phys.* **2025**, *162*, 184503.
(68) Perego, C.; Salvalaglio, M.; Parrinello, M. Molecular dynamics simulations of solutions at constant chemical potential. *J. Chem. Phys.* **2015**, *142*, 144113.
(69) Pietrucci, F.; Laio, A. A. Collective Variable for the Efficient Exploration of Protein Beta-Sheet Structures: Application to SH3 and GB1. *J. Chem. Theory Comput.* **2009**, *5*, 2197−2201.
(70) Steinhardt, P. J.; Nelson, D. R.; Ronchetti, M. Bond-orientational order in liquids and glasses. *Phys. Rev. B* **1983**, *28*, 784−805.
(71) Lechner, W.; Dellago, C. Accurate determination of crystal structures based on averaged local bond order parameters. *J. Chem. Phys.* **2008**, *129*, 114707.
(72) Rein ten Wolde, P.; Ruiz-Montero, M. J.; Frenkel, D. Numerical calculation of the rate of crystal nucleation in a Lennard-Jones system at moderate undercooling. *J. Chem. Phys.* **1996**, *104*, 9932−9947.
(73) Bonomi, M. PLUMED Masterclass 21.7: Optimising PLUMED performances. https://www.plumed-tutorials.org/lessons/21/007/data/NAVIGATION.html, 2021 (accessed 10 07 2025).
(74) Babin, V.; Roland, C.; Sagui, C. Adaptively biased molecular dynamics for free energy calculations. *J. Chem. Phys.* **2008**, *128*, 134101.
(75) Ferrarotti, M. J.; Bottaro, S.; Pérez-Villa, A.; Bussi, G. Accurate Multiple Time Step in Biased Molecular Simulations. *J. Chem. Theory Comput.* **2015**, *11*, 139−146.
(76) Paszke, A.; et al. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2019.
(77) Abadi, M.; et al. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*: Savannah, GA, 2016, pp 265−283.

https://doi.org/10.1021/acs.jpcb.5c07562
J. Phys. Chem. B XXXX, XXX, XXX−XXX