







# PyRETIS 3: Conquering rare and slow events without boundaries

Wouter Vervust<sup>1</sup>  | Daniel T. Zhang<sup>2</sup>  | An Ghysels<sup>1</sup>  | Sander Roet<sup>3</sup>  | Titus S. van Erp<sup>2</sup>  | Enrico Riccardi<sup>4</sup> 

<sup>1</sup>IBiTech–BioMMedA Group, Ghent University, Ghent, Belgium

<sup>2</sup>Department of Chemistry, Norwegian University of Science and Technology, Trondheim, Norway

<sup>3</sup>Department of Chemistry, Utrecht University, Utrecht, The Netherlands

<sup>4</sup>Department of Energy Resources, University of Stavanger, Stavanger, Norway

## Correspondence

Enrico Riccardi, Department of Energy Resources, University of Stavanger, Stavanger, Norway.

Email: [enrico.riccardi@uis.no](mailto:enrico.riccardi@uis.no)

## Abstract

We present and discuss the advancements made in PyRETIS 3, the third instalment of our Python library for an efficient and user-friendly rare event simulation, focused to execute molecular simulations with replica exchange transition interface sampling (RETIS) and its variations. Apart from a general rewiring of the internal code towards a more modular structure, several recently developed sampling strategies have been implemented. These include recently developed Monte Carlo moves to increase path decorrelation and convergence rate, and new ensemble definitions to handle the challenges of long-lived metastable states and transitions with unbounded reactant and product states. Additionally, the post-analysis software PyVisa is now embedded in the main code, allowing fast use of machine-learning algorithms for clustering and visualising collective variables in the simulation data.

## KEYWORDS

kinetics, path sampling, PyRETIS, Python, rare event, slow event

## 1 | INTRODUCTION

The constant increase in high-performance computing (HPC) power enables molecular simulations to consider an increasing number of particles and a significantly longer simulated time. These hardware advancements have been complemented by the development of more efficient algorithms and software, substantially amplifying the effectiveness and predictive capacity of these methods. Despite these advancements, the study of rare transitions remains a computational challenge, as conventional simulations often capture insufficient transition events for statistical kinetic analysis.

For a numerical example, consider the dissociation rate of the drug molecule imatinib from the kinase protein ABL, which is approximately  $10^{-3} \text{ s}^{-1}$ .<sup>1</sup> Using a solvated simulation box of about 50,000 atoms, one can simulate up to 300 ns per day using a recent A100 GPU with an AMD EPYC CPU on the GROMACS MD software (version 2021.3).<sup>2</sup> Assuming dissociation events follow a Poisson process, one expects to observe an unbinding event after

9126 millennia of simulation time. Clearly, an increase of simulation speeds alone, albeit several orders of magnitude, will not suffice to extract kinetic information from biologically and chemically relevant systems.

In response, rare event simulations have emerged as a pivotal algorithmic advancement, enhancing the capabilities of molecular dynamics simulations for studying transition events.<sup>3</sup> In particular, the replica exchange transition interface sampling (RETIS) technique has proven to be one of the most accurate and efficient techniques for computing exact quantitative unbiased dynamical properties.<sup>4</sup> Yet, the necessity persists for intuitive and user-friendly software that can broaden the utilisation of these advanced simulation techniques, aiming to establish them as a standard tool accessible to non-specialists.

With this rationale, we developed PyRETIS, a Python library dedicated to efficiently simulating rare events in molecular systems based on the RETIS algorithm. Since its first release in 2017,<sup>5</sup> PyRETIS has undergone significant improvements, including extended features and algorithmic refinement in PyRETIS 2 in 2020.<sup>6</sup> To the best of our knowledge, PyRETIS is currently one of only two publicly available

Wouter Vervust and Daniel T. Zhang share the first authorship of the present work.

path sampling codes capable of executing RETIS, the other being Open Path Sampling (OPS) code,<sup>7,8</sup> which was released in 2019.

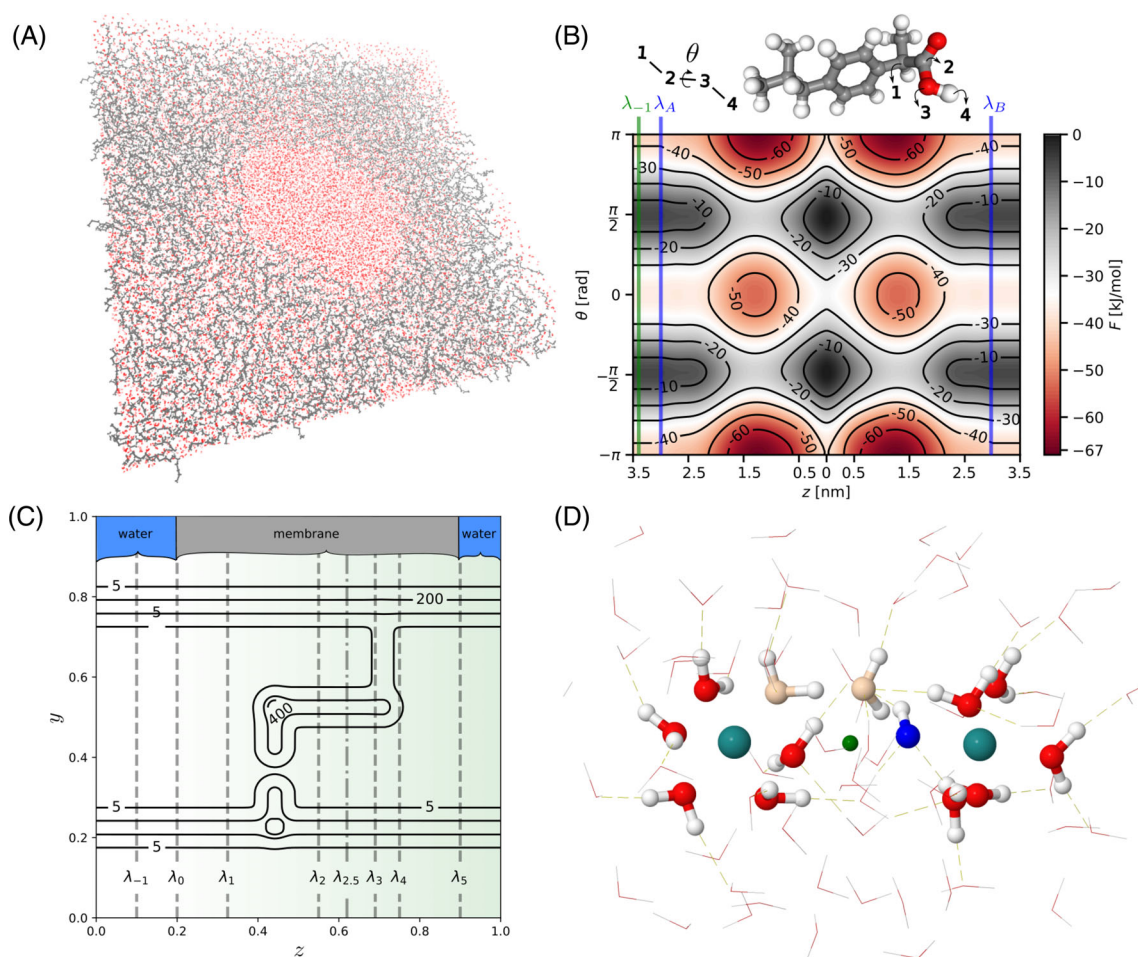
PyRETIS, in its different releases, has been shown effective in multiple studies on a rather broad set of topics ranging from chemical reactions,<sup>9–12</sup> the adoption of a bacterial protein to DNA,<sup>13</sup> thin film breakage,<sup>14,15</sup> the solid-solid transition between the wurtzite and rock salt crystal structures,<sup>16</sup> and the oxygen permeation through membranes.<sup>17</sup> These comprehensive studies have served as a sturdy foundation, guiding the software's development over the past three years. A graphical representation of the latest works is included in Figure 1.

The latest iteration introduced in this article, PyRETIS 3, incorporates novel structural and algorithmic enhancements. Notably, this new release boasts an optimised architecture designed for parallel simulations and features an interface with machine learning algorithms, simplifying the post-analysis of simulation results. Its modular structure is visually depicted in the accompanying flowchart (Figure 2). A comprehensive discussion of the theory supporting the software's development can be found in the literature.<sup>3,4,17,21–24</sup>

## 2 | ALGORITHMIC DEVELOPMENTS

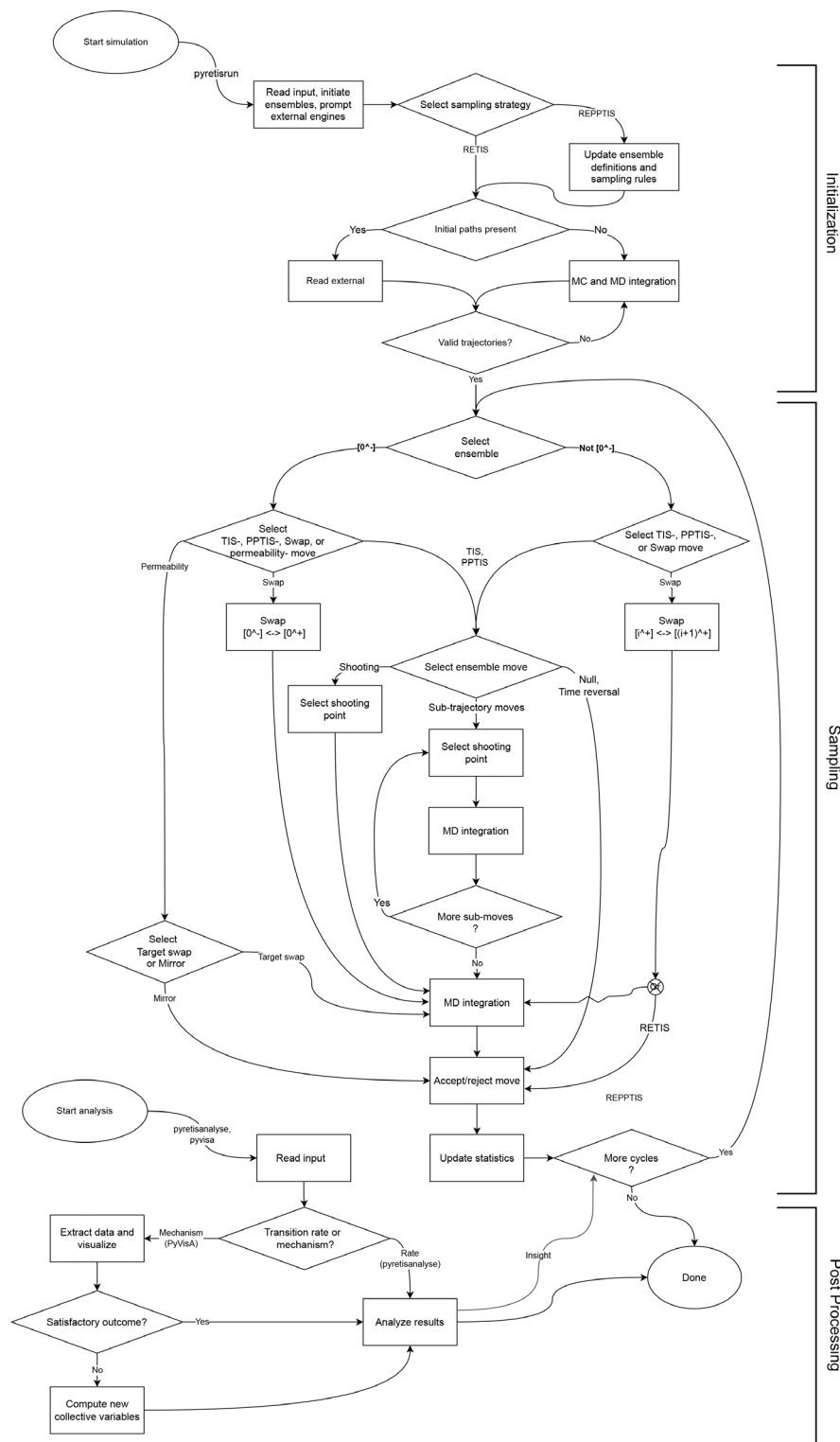
### 2.1 | PPTIS: Partial path transition interface sampling

In replica exchange transition interface sampling (RETIS), the sampling is conducted by generating a large number of trajectories that are accepted or rejected based on the Metropolis-Hastings law.<sup>25,26</sup> To use RETIS, a set of interfaces along a main collective variable has to be positioned, where each of these interfaces defines an ensemble. A trajectory belongs to the path ensembles  $[i^+]$  when it starts at  $\lambda_0$  ( $=\lambda_A$ , the boundary around stable state definition for the reactant state A), crosses the specific interface  $\lambda_i$  and reaches either the product state at  $\lambda_B$  or completely returns to the reactant state at  $\lambda_0$ , as visualized in Figure 3A. However, when the transition is not only rare, but also slow, as paths may get stuck in local metastable states, the average path lengths may extend beyond tens or hundreds of nanoseconds for some of the RETIS ensembles.



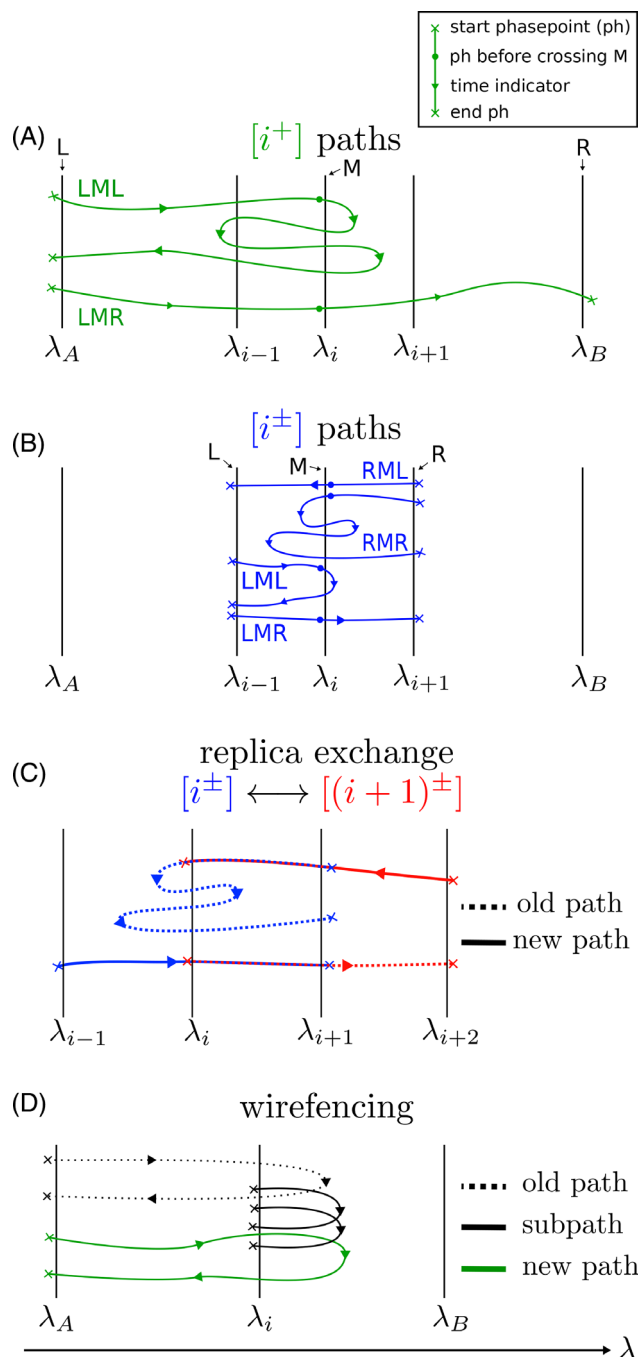
**FIGURE 1** Snapshots and descriptive representations of the latest works performed using the PyRETIS simulation library. (A) Thin film breakage<sup>14,15</sup> was investigated with force field dynamics using the GROMACS engine.<sup>2</sup> (B) The permeability of ibuprofen through a bilayer membrane<sup>18</sup> was estimated using the PyRETIS internal engine. (C) The increase in ergodicity by addition of the replica exchange move to PPTIS was demonstrated using a maze system.<sup>18</sup> (D) The electron transfer reaction between two ruthenium (2+/3+) ions<sup>15,19</sup> was investigated with *ab initio* dynamics using the CP2K<sup>20</sup> engine.

FIGURE 2 Flowchart of PyRETIS 3 logic.



To address this limitation, we have implemented partial path transition interface sampling (PPTIS)<sup>27</sup> into the PyRETIS code. With this method, the path ensembles  $[i^{\pm}]$  contain paths that cross  $\lambda_i$  and start and end on neighbouring interfaces  $\lambda_{i-1}$  or  $\lambda_{i+1}$ , as visualized in Figure 3B. From the paths, the local crossing probabilities  $p_i^{\pm}$  and  $p_i^{\mp}$  can be determined, and with a recursive relation,<sup>27</sup> the global crossing probability  $P_A(\lambda_B|\lambda_A)$  can be estimated based on the assumption of memory loss.

In PyRETIS 3, users can opt for a PPTIS simulation with a simple keyword in the input.rst file, as described in the online documentation. In the summary file pathensemble.txt, the path labels of the  $[i^{\pm}]$  ensemble can now take on four different path-types for accepted paths: LML or LMR (as in a TIS simulation), and also RMR or RML (Figure 3B). Here, 'L', 'M', and 'R' denote the left, middle, and right interfaces within a designated path ensemble. In post-analysis, the ratio of different path-types determines local crossing probabilities,



**FIGURE 3** (A) Trajectories of a TIS path ensemble  $[i^+]$  start from the reactant interface  $\lambda_A$  (L), cross the specific interface  $\lambda_i$  (M) and reach either the product interface  $\lambda_B$  (R) or completely return to  $\lambda_A$ . (B) Trajectories of a PPTIS path ensemble  $[i^\pm]$  start from either the left-neighbouring (L) or right-neighbouring (R) interface of  $\lambda_i$  (M), after which they cross M, and return to either L or R. (C) Schematic representation of a replica exchange move in a REPPTIS simulation. The RMR path of  $[i^\pm]$  (blue dotted) is extended backward in time to create a new RML path of  $[(i+1)^\pm]$  (solid red). Likewise, the LMR path of  $[(i+1)^\pm]$  (red dotted) is extended backwards in time to create a new LMR path of  $[i^\pm]$  (blue solid). (D) Schematic representation of a WF move in a RETIS simulation. A new path (solid green) is generated from an old path (dotted black) via a chain of subtrajectories (solid black), which can significantly improve path decorrelation compared to the standard shooting move.

for example,  $p_i^\pm = \text{LMR}/(\text{LMR} + \text{LML})$  denotes the local probability of reaching the right interface  $\lambda_{i+1}$  rather than the left interface  $\lambda_{i-1}$ , given that a path has crossed the middle interface  $\lambda_i$  after having crossed the left interface  $\lambda_{i-1}$ .

## 2.2 | REPPTIS: Replica exchange partial path transition interface sampling

Akin to RETIS enhancing the transition interface sampling (TIS)<sup>28</sup> efficiency through replica exchange,<sup>29–31</sup> similar advancements in PPTIS can lead to the REPPTIS method.<sup>18</sup> The inclusion of replica exchange in REPPTIS relies on MC moves designed to exchange path segments among adjacent path ensembles. These segments are then extended until they meet the acceptance criteria of a neighbouring ensemble, an example of which is shown in Figure 3C. As shown in Reference 18, these swaps notably facilitate the exploration in regions partitioned by energetic or kinetic barriers orthogonal to the order parameter  $\lambda$ , thereby substantially improving the sampling's ergodicity

The acceptance of a swap hinges on the pairing of path-types (LMR, LML, RML, or RMR) and the selected propagation directions (forward or backward in time). For example, an LMR path in the  $[2^\pm]$  ensemble can be extended into  $[3^\pm]$  by forward propagation in time, while backward propagation would push the path towards the  $[1^\pm]$  ensemble. If propagation directions are incompatible with the path-types, the replica exchange move is immediately rejected without requiring MD steps, resulting in an SWD entry ('Swap Wrong Direction') in the pathensemble.txt files of the relevant ensembles. Next, if propagation directions are compatible with path-types, path segments are extended until they cross either the left or right interface of the new ensemble.

Contrary to swaps between the RETIS  $[i^+]$  ensembles, swaps between REPPTIS ensembles require MD integration in the extension step. In RETIS, only the swap between  $[0^-]$  and  $[0^+]$  require MD integration. Fortunately, in both RETIS and REPPTIS, any required MD integration commences only upon confirmation of swap acceptance. Consequently, the CPU time spent on MD integration is almost never wasted. The only exception is when paths exceed the user-specified maximum path length. Such occurrences should be rare as (RE)PPTIS is designed to reduce path lengths. Frequent occurrences suggest that the user should either add more interfaces, increase the maximum path length parameter, or change the order parameter definition. An example PyRETIS 3 input file running REPPTIS with a certain swap frequency is shown in Figure 4. The user can refer to the online documentation for detailed instructions on setting up a REPPTIS simulation.

## 2.3 | Improved stone skipping & web throwing

To increase the simulation efficiency, advanced shooting moves in the form of stone skipping (SS) and web throwing (WT)<sup>23</sup> were already implemented in PyRETIS 2. Compared to standard shooting

```

Simulation
-----
task = repptis
steps = 10000
interfaces = [-0.5, -0.3, 0.0, 0.3, 0.5]
permeability = True
zero_left = -0.75

TIS settings
-----
freq = 0.0
maxlength = 100000

RETIS settings
-----
swapfreq = 0.2
swapsimul = True

```

**FIGURE 4** New keywords introduced to perform a REPPTIS simulation. Replica exchange moves are introduced by setting the ‘task’ keyword to repptis. Typically, one will not perform time-reversal moves in the PPTIS framework, which is accomplished by setting the ‘freq’ keyword of TIS settings to 0.0. The relative frequency of shooting moves to replica exchange moves is set by the ‘swapfreq’ keyword of the RETIS settings. In this example, 20 % of the moves will perform replica exchange. This input excerpt is tailored for a permeability simulation, as the ‘permeability’ keyword is set to True. The positioning of the accompanying  $\lambda_{-1}$  interface is done by setting the ‘zero\_left’ keyword to  $-0.75$ .

methods,<sup>32</sup> both SS and WT are more cost-efficient per MD step. They achieve this by leveraging a sequence of intermediate short paths (subpaths) that minimise correlations between previously existing and newly generated paths. After a number of subpaths have been completed, the last one is extended to become a new full path. Although SS and WT yield an increase in efficiency of more than one order of magnitude in case studies,<sup>23</sup> their practical implementation can be impeded when linked with external MD engines or when the calculation of the order parameter is expensive.

More specifically, new SS/WT subpaths must be launched from a randomly perturbed phase point (time slice) of the previous path/subpath that establishes a crossing with some relevant interface (like the main ensemble’s interface that always needs to be crossed). Typically, only the velocities are changed in this perturbation, but after this process, it should be verified that the perturbed phase point is still a crossing point: the other side of the interface should be reached after a time step forward or backward in time. If this is not the case, the move is not directly rejected, but new velocities should be generated. Especially if the time step considered by the RETIS algorithm actually consists of several MD steps, this requirement is not always easy to fulfil and multiple velocity randomization attempts can be required.

If a RETIS time step is a single MD step (the subcycle is set to 1), Reference 15 proposed two strategies to minimise the computational cost for doing the one-step crossing test. First, the velocity-Verlet integrator<sup>15,33</sup> can be reformulated to predict the next configuration point without the

associated expensive force calculation. Therefore, given a configuration-based order parameter and a relatively cheap velocity generation, the cost of testing reduces considerably such that the one-step crossing requirement can quickly be achieved. The second strategy is to modify the velocity generation procedure to increase the likelihood of pushing the next step across the interface. For example, for an  $N$  particle system, certain velocities can be kept or reversed instead of letting all  $3N$  velocity components be regenerated from a Maxwell-Boltzmann distribution.

However, in the case that one MD engine call implies performing several MD steps (i.e., subcycle is set to 10–2000), and/or when the order parameter calculation is expensive, the increased testing cost for the one-step crossing condition reduces the SS and WT computational efficiency. We therefore developed a third type of advanced shooting move that does not require the one-step crossing condition, which is implemented in PyRETIS 3 and discussed in the next section.

## 2.4 | Wire fencing

To circumvent the issues related to the one-step crossing condition, we formulated a third advanced shooting move within the subtrajectory family that we named “wire fencing” (WF).<sup>15</sup> In comparison to SS and WT, a new WF subpath can be launched from a configuration point of the previous path or subpath that lies between the path ensemble’s interface  $\lambda_i$  and  $\lambda_{cap}$ , a user-defined *cap interface*:  $\lambda_i < \lambda_{cap} \leq \lambda_B$ . It is therefore not restricted to crossing points of specific interfaces. In the cases of potential energy surfaces that have gradual regions close to state  $B$ , the shooting point selection can be restricted to only occur in steep regions by a suitable  $\lambda_{cap}$  placement. A schematic representation of the WF move is given in Figure 3D.

The WF move, like all advanced shooting moves, are best combined with the *high-acceptance* technique.<sup>15,23</sup> The high-acceptance scheme alters the sampled path distribution, a change that can be precisely adjusted through a reweighting scheme in the post-simulation analysis. In this specific formulation, the Metropolis-Hastings acceptance criteria solely reject paths that both commence and culminate at the interface  $\lambda_B$ .

To use SS, WT or WF in PyRETIS 3, the user can select, for each ensemble, the type of shooting-moves, the number of subpaths, the positions of the additional interfaces (like the cap-interface in WF), and whether to adopt the high-acceptance sampling scheme.

## 2.5 | The $\lambda_{-1}$ interface

In both RETIS and REPPTIS simulations, the flux term is computed from the average path lengths of the  $[0^-]$  and  $[0^+]$  ensembles. The former ensemble,  $[0^-]$ , is conventionally defined by a single interface and obeys different sampling rules compared to the other ensembles. In some modelling situations, it could be advantageous to restrict the  $[0^-]$  ensemble within two interfaces. A first example is when the reactant state  $A$  is unbound to the left, such as when it signifies an infinite reservoir or when it is barrierlessly linked to state  $B$  through periodic boundary conditions.



In such cases, while the rate might not be well defined, other dynamical properties—such as the permeability in a membrane system—are still ascertainable. A second example is when the reactant state  $[O^-]$  extends over a finite, but very large collective variable  $\lambda$  range. Here, it may be more strategic to focus on sampling the region closer to  $\lambda_0$ . In Reference 34, an additional interface  $\lambda_{-1}$  was introduced to the left of  $\lambda_0$ , and the new ensemble  $[O^-]$  was defined as the path ensemble containing trajectories that start and stop on  $\lambda_0$  or  $\lambda_{-1}$ . Consequently, the region  $\lambda < \lambda_{-1}$  is never sampled, hence avoiding periodic boundary crossings or waste of computer time in a non-relevant region of phase space.

A theoretical complication is that an adaptation is needed to match the  $[O^-]$  and  $[O^+]$  ensemble. Indeed, the number of paths that may be cut out from a very long equilibrium MD simulation and that belong to the standard  $[O^-]$  is equal to those of  $[O^+]$ , that is,  $N_{[O^-]} = N_{[O^+]}$ . The  $\lambda_{-1}$  interface increases the number of paths, however, as the trajectory is cut more often, and  $N_{[O^-]} > N_{[O^+]}$ . To get the correct permeability, a correction  $\xi$  is therefore needed,<sup>34</sup> where  $\xi = N_{[O^+]}/N_{[O^-]}$ . Fortunately, the factor  $\xi$  can be readily evaluated in the post-processing by PyRETIS-3.

## 2.6 | Mirror move and target-swap move

The permeability coefficient can directly be computed in PyRETIS 3. It can be obtained by considering the rate of transition while following a single permeant, referred to as the target, from left to right through a given region (e.g., a membrane). The presence of other permeants in the system affects the rate, as they are part of the surrounding environment. Transitions from right to left are prevented due to the presence of the  $\lambda_{-1}$  interface. However, for the sake of sampling efficiency, it would be beneficial to incorporate the statistics of other transitioning permeants and utilise transitions in both directions, including from right to left, whenever the membrane is (statistically) symmetric. This potential for sampling efficiency gain is achieved through the target-swap move and the mirror move, respectively.

The mirror move involves mirroring the  $z$  coordinates of all particles of all time slices of the previous path across an  $xy$ -mirror plane located between two periodic images of the membrane. Additionally, the  $z$ -component of the particle velocities is flipped. However, it is important to note that the mirror move must be accompanied by setting the  $\lambda_{-1}$  interface at an equal distance away from the mirror plane in the water slab as the  $\lambda_0$  interface, but on the opposite side. The mirror move solely applies to the  $[O^-]$  ensemble and has the consequence that a previous path ending at  $\lambda_{-1}$ , will end at  $\lambda_0$  after this move. It is worth noting the distinction from the time-reversal move, as the mirror move would lead to a switch to the opposite interface even if the previous path started and ended at the same side. As a result of the new path now ending at  $\lambda_0$ , it can be successfully swapped with a  $[O^+]$  path in the next MC step.

For code-technical reasons, PyRETIS-3 implements the mirror move in a slightly different but equivalent manner. Instead of changing the particle coordinates and velocities directly, PyRETIS-3

modifies the definition of the reaction coordinate (RC) by using the  $z$  of the target's position, and mirroring it across a mirror plane. This alternative implementation facilitates the integration of PyRETIS-3 with external molecular dynamics (MD) engines, which may have different methods for altering coordinates and velocities. The flag indicating the sign of  $z$  to be used in the RC definition is also exchanged during the replica exchange moves of the RETIS algorithm.

The target-swap move also plays a crucial role in improving sampling efficiency. This move randomly selects a permeant to be considered as the new target from all the time slices of the previous path. It considers permeants within the  $[O^-]$  boundaries, namely  $\lambda_{-1}$  and  $\lambda_0$ , at that time slice. Once a random time slice and permeant pair is chosen, the trajectory is traced both forward and backward in time until the new target reaches the boundaries. This process may involve extending or truncating the old trajectory along each time direction. Ultimately, the final trajectory is accepted or rejected based on a Metropolis decision, ensuring that detailed balance is maintained.

It is worth noting that even if the mirror move and the target swap move operate exclusively in the minus ensemble ( $[O^-]$ ), the enhanced exploration trickles down to the other ensembles through replica exchange swaps, improving the sampling across all ensembles. The effectiveness of both moves was distinctly demonstrated in a two-channel system where it significantly enhanced sampling in the collective variable space orthogonal to the reaction coordinate.<sup>34</sup>

## 3 | POST-PROCESSING

### 3.1 | Error analysis

Calculating statistical errors within a RETIS simulation is a non-trivial task due to the various types of correlation. In contrast to TIS, where path simulations are independent and standard error propagation rules can be used, RETIS introduces additional complexities. Although block averages can address correlations within a path ensemble, correlations extend across different path ensembles in RETIS due to path swapping, which results in shared data between the ensembles. The involvement of the  $[O^+]$  path ensemble in both the flux and crossing probability calculations, along with the expected correlation between path length and reaction progress, further hinders assuming error independence. Additionally, computing the total crossing probability using the weighted histogram analysis method (WHAM)<sup>21,35,36</sup> also increases the difficulty for dealing with correlations as it utilises overlaps in path ensembles to improve the accuracy.

The implementation of standard block averaging poses practical challenges. This issue is evident in REPPTIS, where  $p_i^{\pm}$  and  $p_i^{\mp}$  may be based on different numbers of sampled trajectories, rendering an absolute block length unsuitable. One potential solution is to adapt the block length to the relative size of the different data sets. For example, in a RETIS/REPPTIS analysis, the first 10% of each data file could be utilised to compute a rate. Subsequently, data between 10% and 20% from all files are employed to calculate a new rate, continuing this process until ten nearly independent estimates are obtained.

These rates can then be used to compute the standard deviation and, ultimately, the error.

In a practical implementation, the described approach can still be viable if the number of blocks (10 in the aforementioned example) is predetermined and fixed. However, block averaging techniques often demonstrate significant fluctuations in computed errors due to the arbitrary choice of block length or, equivalently, the number of blocks. It is therefore recommended to conduct error analysis using a range of block lengths. By evaluating the computed errors across different block lengths and observing the graph of computed error versus block length  $m$ , one can identify a plateau region where the errors stabilise. This specific region is usually identified for sufficiently large  $m$  values, where the blocks can be deemed uncorrelated yet remain small enough to maintain a substantial number of blocks. Taking the average of the computed errors within this plateau region is considered good practice, as it provides a more reliable estimate of the statistical uncertainty. Yet, partitioning all data files into many sets of relative blocks is cumbersome and time consuming.

We have, therefore, taken a more practical yet sound approach, that we refer to as recursive block errors, which surprisingly has not been widely mentioned as an alternative to standard block averaging. The recursive block errors approach is based on a single data file containing the running estimate as a function of performed MC cycles of the property under investigation, such as the rate, flux, crossing probability, or specific path ensemble properties like local crossing probabilities and path lengths. These running estimates are standard outputs from PyRETIS simulations.

Suppose the RETIS simulation consists of  $N$  MC cycles, where a cycle typically implies the update of each path ensemble by a MC move. Let  $k[n]$  with  $1 \leq n \leq N$  be the running estimate of the rate after  $n$  cycles have been completed. Given a block length of  $m$ , there are  $M = \text{int}(N/m)$  blocks. We now introduce the *recursive-block* value  $k_j^r$  for each block  $j = 1, 2, \dots, M$ . These values are implicitly defined by ensuring that the mean of the initial  $j$  *recursive-block* values matches the running estimate after  $jm$  MC cycles:

$$k[jm] = (k_1^r + k_2^r + k_3^r + \dots + k_j^r) / j \quad (1)$$

which leads to a recursive relation for  $k_j^r$ :

$$k_j^r = \begin{cases} jk[jm] - (k_1^r + k_2^r + k_3^r + \dots + k_{j-1}^r) & \text{if } j > 1 \\ k[m] & \text{if } j = 1 \end{cases} \quad (2)$$

From this, given a specific block length  $m$ , the following non-recursive relation can be extracted

$$k_j^r = jk[jm] - (j-1)k[(j-1)m] \quad (3)$$

These block values  $k_j^r$  are then used to compute the standard deviation between them and ultimately the estimated error for this particular block size  $m$ , just like is done with standard block averaging.

The great advantage of this simple relation is that the post-analysis can start using just the running estimate  $k[n]$  with  $1 \leq n \leq N$ ,

which subsequently can be reused to compute  $k_j^r$  for a large set of block sizes  $m$ . Consequently, this method enables an efficient computation of the estimated error concerning block size, facilitating the extraction of a final robust error estimate by averaging specifically within the identified plateau region. In the appendix, we show that this approach converges to the same error estimates as when standard-block averages are used.

## 3.2 | Permeability

It is challenging to compute the permeability  $P$  of permeants through a barrier (e.g., through a membrane), when a high free energy barrier needs to be overcome or when the permeants are trapped in a free energy well for an extended time.<sup>37</sup> An example of the former is the water permeation through phospholipid membranes,<sup>38–40</sup> and an example of the latter is the transport of oxygen molecules through membranes.<sup>41,42</sup> In Reference 34, it was shown that the permeability through a region can be estimated from the RETIS crossing probability  $P_A(\lambda_B|\lambda_A)$ . As an additional quantity for  $P$ , the average time that a path spends in a reference interval in the  $[0^-]$  or  $[0^+]$  ensemble needs to be calculated. Moreover, the  $\xi$  factor (see subsection on the  $\lambda_{-1}$  interface) also needs to be evaluated when a  $\lambda_{-1}$  interface is used. These two extra quantities were implemented in a straightforward way in the PyRETIS post-processing analysis tools.

With the PyRETIS implementation in Reference 34, a series of 1D examples of barriers with varying height and viscosity settings were tested. In addition, the permeation of particles through a 2D membrane with two distinct permeation pathways was successfully mapped out, where the use of the target-swap move and mirror move enhanced the sampling efficiency. When the permeation event is not only rare but also slow, the permeability can be assessed with the PPTIS or REPPTIS methodology.<sup>18</sup> The (RE)PPTIS implementation was used to investigate the permeation through a 2D maze, representing a membrane with two permeation pathways, where one has an entropic barrier and the other an energetic barrier. The computed (RE)PPTIS permeabilities were benchmarked against the RETIS permeabilities, where the replica exchange moves in REPPTIS could significantly reduce the effect of memory-loss in the partial path sampling method.

## 3.3 | PyVisA

PyVisA,<sup>43</sup> a post-processing visualisation and analysis tool, has been integrated into the PyRETIS 3 release, with its own executable and optional graphical user interfaces. The library permits to directly load and visualise simulation outcomes, reducing the data handling time and costs, as data can be checked and compressed remotely and visualised locally. The simulation results can be displayed as a whole or in sections. Different collective variables, simulation time, trajectory status, trajectory number can be directly grouped for different ensembles supporting the analysis and visualisation of simulation results. A customizable interface to improve the appearance of the reports has

been constructed and an interactive interface implemented. Furthermore, the selected data subset can be saved in ".hdf5", simple ".txt", or with a ".json" format. Images can also be exported directly as ".png". To provide further visualisation customizability, PyVisA can also output a minimalistic python script to regenerate the selected image from selected data. Clearly, the script can be then re-adapted to the best user convenience. If trajectory data (x-, y-, z-positions) are stored, each data-point of the displayed trajectory is linked by PyVisA to the original source, and a molecular 3D visualisation can be launched showing the molecular structure at the points of interest. The selected data can then be fed directly to a set of machine learning approaches such as clustering algorithms,<sup>44</sup> random forests,<sup>45</sup> calculation of the Pearson correlation matrix of coefficients,<sup>46,47</sup> and so forth. These approaches are provided by the scikit-learn python package<sup>48</sup> and the sampling data is internally wrangled such that it can directly be fed to these algorithms. PyVisA has been constructed to be executed independently while simulations are ongoing, allowing for intermediate descriptions and visualisations of the simulation outcomes. The new

panels to access the interactive functionality and the integrated machine learning modules are shown in Figure 5.

## 4 | OTHER UPDATES

### 4.1 | Code structure

In PyRETIS 2 users could define different settings for different ensembles, to allow for better customization in the design of sampling problems. In PyRETIS 3, the internal representation of ensembles has also been subdivided. Each ensemble is represented by an independent object, allowing multiple simulations to be performed contemporaneously. Each ensemble can also be, in principle, executed with a different external engine. The trajectory management has been optimised, where trajectory cleaning is now performed after every ensemble move rather than waiting for an entire MC cycle to finish. This significantly reduces storage space for large systems using many ensembles.

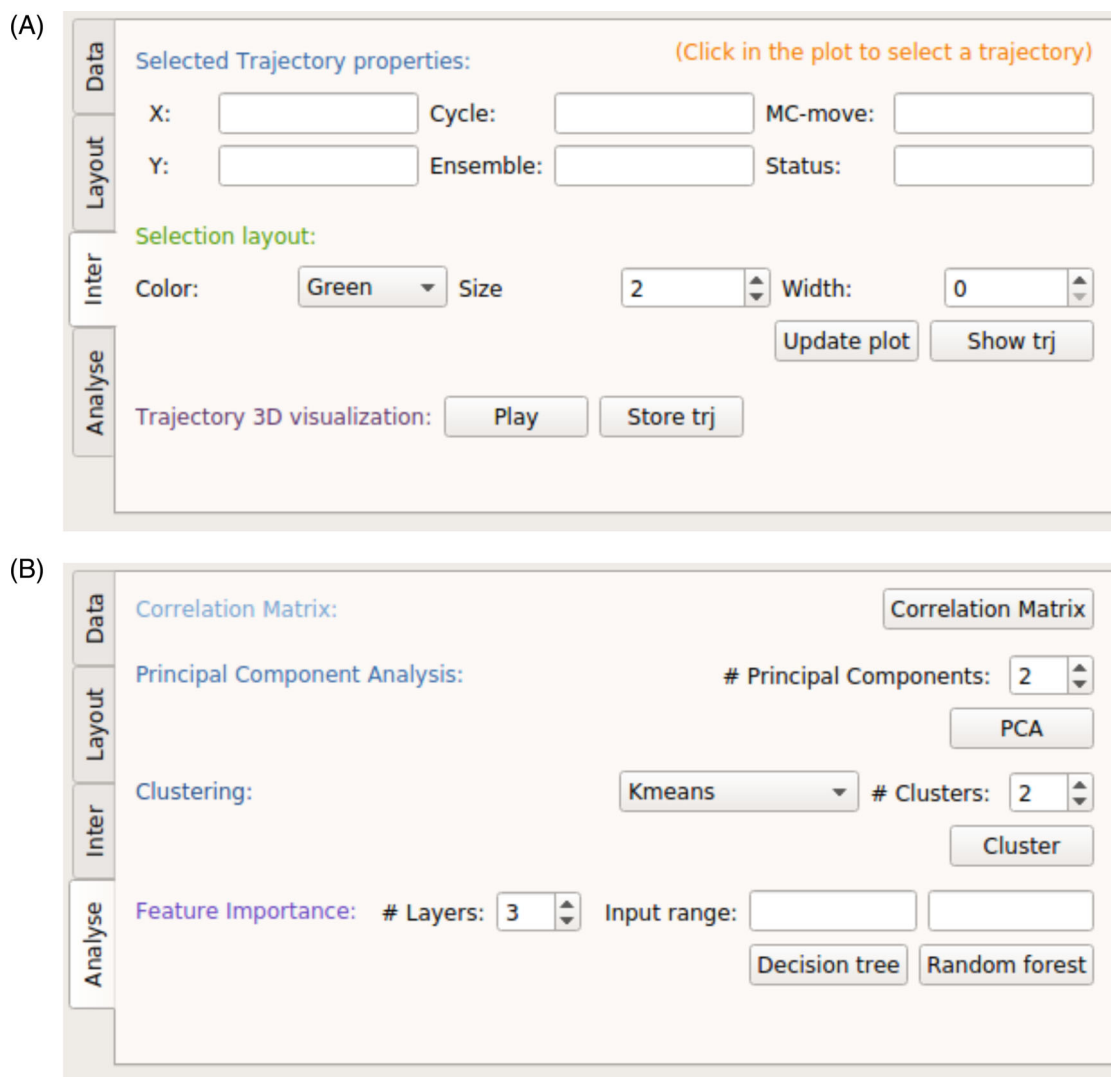


FIGURE 5 PyVisA interactive visualisation panel (A) and PyVisA machine learning model control panel (B).



The new code structure allows for effective parallel simulations to be performed in different ensembles, potentially speeding up sampling performance by optimising cluster usage. An effective simulation handler procedure to advantageously execute multiple parallel simulations has been recently published by Roet et al.<sup>49</sup> and it can be expected to be included in forthcoming software releases.

## 4.2 | Interface with the external engine Amsterdam modeling suite

The PyRETIS code has been developed with the intent to use external engines (e.g., GROMACS,<sup>2</sup> LAMMPS,<sup>50</sup> CP2K,<sup>20</sup> OpenMM<sup>51</sup>) and to facilitate the implementation of new ones. In collaboration with SCM (Software for Chemistry & Materials), we have established an interface between PyRETIS and their Amsterdam Modeling Suite (AMS).<sup>52</sup> This development allows PyRETIS 3 to utilise AMS as its MD engine, significantly expanding the range of dynamic simulations it can perform. The extended capabilities include the ability to conduct *ab initio* MD based on the Amsterdam Density Functional (ADF) package, ReaxFF,<sup>53</sup> and Density Functional Tight Binding (DFTB).<sup>54</sup>

These functionalities are already implemented and are currently available in the development branch of the PyRETIS code. At the time of submitting this paper, the documentation and unit testing for this feature were not fully completed. Nevertheless, the functionalities are fully operational. We encourage users interested in utilising this functionality to reach out to the PyRETIS developers for support and guidance on its usage.

## 5 | USER GUIDELINES

To maximize the efficiency of the sampling strategies offered by PyRETIS, we provide some guidelines on which simulation technique is best suited for which applications. In general, RETIS is a very efficient sampling strategy for rare events, which are characterized by long waiting periods followed by the actual transition that happens very quickly. REPPTIS, on the other hand, is designed to tackle slow events. A slow event can still be characterized by a long waiting period, after which the transition happens in a sluggish fashion, where a transition spends considerable time in metastable states along the reactive pathway.

In more technical terms, RETIS is most suitable for reaction mechanisms that are well-described by overcoming a single, large energetic and/or entropic barrier. If the reaction mechanism includes metastable states, a collection of RETIS simulations can still be used if the location of the free energy wells are well-known beforehand. A Markov state model can then be built from the collection of RETIS simulations, from which the global transition rate can be estimated. Examples of this include the permeation of small molecules (methanol, ethanol, etc.) through nanoporous materials or phospholipid bilayers. When the system becomes more complex, the optimal reaction coordinate quickly becomes elusive due to the increasing number of transition

states and metastable states. Consequently, (unknown) metastable states emerge along (unknown) orthogonal degrees of freedom concerning the selected reaction coordinate, for which REPPTIS is better suited. Examples include the association and dissociation of drug molecules to proteins, the permeation of small drug molecules through phospholipid bilayers, protein-protein interactions, and so forth.

## 6 | USER SUPPORT

The code has been updated with the latest python libraries and its dependencies on external packages have been minimized in order to increase its maintainability. Our software development has been inspired by the FAIR software principles. The code is fully open source and shared on GitLab and GitHub.<sup>55</sup> Documentation, tests, and examples are constantly updated to simplify the code usability by external users.

## 7 | CONCLUSIONS AND FUTURE WORK

We have released the third version of the PyRETIS code, which includes algorithmic developments that can greatly improve sampling efficiency. In particular, partial path transition interface sampling (PPTIS), replica exchange partial path transition interface sampling (REPPTIS), and advanced shooting moves (wire fencing, mirror move, and target-swap move) were implemented and previous implementations of the stone skipping and web throwing moves were improved. Additionally, PyRETIS 3 solidifies the  $\lambda_{-1}$  interface feature based on a new well defined and relevant path ensemble  $[0^{-}]$ , which improves the purely pragmatic implementation of the  $\lambda_{-1}$  interface option in PyRETIS 2. Within the  $[0^{-}]$  ensemble, two additional terms are calculated: the average time that a path in this ensemble spends in a reference region and the  $\xi$  factor. These two values, together with the computed RETIS crossing probability, allow one to compute the permeability coefficient in bounded and unbounded systems. The approach does also not require any additional flux calculations, which was the pragmatic solution suggested in the PyRETIS 2 for bounded systems. A direct approach to compute the permeability in unbounded systems was not available in PyRETIS 2.

In the post-processing phase of the code, we enhanced our error analysis by employing *recursive block* analysis. This method is particularly effective in handling intricate correlations, especially in scenarios where the final computed properties stem from a series of separate yet interdependent simulations, a characteristic notably pertinent to RETIS and REPPTIS. We have then further improved and formally included PyVisA as a part of the PyRETIS 3 release. The library provides an analysis and visualisation toolkit to facilitate the study of simulation output. In particular, the library prepares the data such that machine learning approaches (e.g., random forest, clustering) can be directly applied, providing enhanced insight on the results.

Further software development will focus on code parallelization. The recent introduction of  $\infty$ RETIS,<sup>49,56</sup> a novel RETIS variant

integrating asynchronous replica exchange with infinite swaps, represents a crucial advancement in line with this objective. This innovation effectively resolves the challenge posed by the imbalanced CPU costs associated with the (advanced) shooting moves, stemming from varying path lengths.

In the domain of force field development and data analysis, the synergy between path sampling and machine learning will continue to advance. For instance, leveraging RETIS simulations at the Ab Initio MD level can yield a pertinent training set of configuration points crucial for establishing a reactive force field through neural networks<sup>57–59</sup> tailored for specific chemical reactions. Once the force field is established, it can be reintegrated into the RETIS simulation, offering unprecedented convergence and reliability. Additionally, mechanistic analysis through committor analysis<sup>60,61</sup> and the predictive capacity analysis<sup>10,21</sup> is anticipated to provide deeper insights with the aid of machine learning tools. The objective is not solely to comprehend the occurrence of rare events like chemical reactions but also to understand methods for enhancing, directing, or impeding them. The PyRETIS development team aims to propel these advancements by offering new open-source computer codes that are publicly accessible, intuitive, and instrumental in tackling complex applications while advancing quantitative path sampling algorithms.

## ACKNOWLEDGMENTS

We acknowledge funding support from the Research Council of Norway for the 'Theolight' toppforsk project (Grant No. 275506). We also acknowledge funding of the FWO (project G002520N and project G094023N), BOF of Ghent University and the European Union (ERC Consolidator grant, 101086145 PASTIME).

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in pyretis at <https://gitlab.com/pyretis/pyretis.git>.

## ORCID

Wouter Vervust  <https://orcid.org/0000-0002-8714-3017>

Daniel T. Zhang  <https://orcid.org/0000-0002-0296-0860>

An Ghysels  <https://orcid.org/0000-0003-0015-2605>

Sander Roet  <https://orcid.org/0000-0003-0732-545X>

Titus S. van Erp  <https://orcid.org/0000-0001-6600-6657>

Enrico Riccardi  <https://orcid.org/0000-0003-1890-7113>

## REFERENCES

- [1] A. Lyczek, B.-T. Berger, A. M. Rangwala, Y. Paung, J. Tom, H. Philipose, J. Guo, S. K. Albanese, M. B. Robers, S. Knapp, et al., *Proc Natl Acad Sci* **2021**, *118*, e2111451118.
- [2] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, E. Lindahl, *SoftwareX* **2015**, *1–2*, 19.
- [3] T. S. van Erp, *Solvent effects on chemistry with alcohols*. Ph.D. thesis, Universiteit van Amsterdam, Netherlands **2003**.
- [4] T. S. van Erp, *Phys. Rev. Lett.* **2007**, *98*, 268301.
- [5] A. Lervik, E. Riccardi, T. S. van Erp, *J. Comput. Chem.* **2017**, *38*, 2439.
- [6] E. Riccardi, A. Lervik, S. Roet, O. Aarøen, T. S. van Erp, *J. Comput. Chem.* **2020**, *41*, 370.
- [7] D. W. H. Swenson, J.-H. Prinz, F. Noe, J. D. Chodera, P. G. Bolhuis, *J. Chem. Theory Comput.* **2019**, *15*, 813.
- [8] D. W. H. Swenson, J.-H. Prinz, F. Noe, J. D. Chodera, P. G. Bolhuis, *J. Chem. Theory Comput.* **2019**, *15*, 837.
- [9] M. Moqadam, E. Riccardi, T. T. Trinh, A. Lervik, T. S. van Erp, *Phys. Chem. Chem. Phys.* **2017**, *19*, 13361.
- [10] M. Moqadam, A. Lervik, E. Riccardi, V. Venkatraman, B. K. Alsberg, T. S. van Erp, *Proc. Natl. Acad. Sci. U. S. A.* **2018**, *115*, E4569.
- [11] S. Roet, C. D. Daub, E. Riccardi, *J. Chem. Theory Comput.* **2021**, *17*, 6193.
- [12] C. D. Daub, E. Riccardi, V. Hänninen, L. Halonen, *PeerJ Phys Chem* **2020**, *2*, e7.
- [13] E. Riccardi, E. C. van Mastbergen, W. W. Navarre, J. Vreeede, *PLoS Comput. Biol.* **2019**, *15*, e1006845.
- [14] O. Aarøen, E. Riccardi, T. S. van Erp, M. Sletmoen, *Colloids Surf., A* **2022**, *632*, 127808.
- [15] D. T. Zhang, E. Riccardi, T. S. van Erp, *J. Chem Phys* **2023**, *158*, 024113.
- [16] A. Lervik, I.-H. Svenum, Z. Wang, R. Cabriolu, E. Riccardi, S. Andersson, T. S. van Erp, *Phys. Chem. Chem. Phys.* **2022**, *24*, 8378.
- [17] E. Riccardi, A. Krämer, T. S. van Erp, A. Ghysels, *J Phys Chem B* **2020**, *125*, 193.
- [18] W. Vervust, D. T. Zhang, T. S. van Erp, A. Ghysels, *Biophys. J.* **2023**, *122*, 2960.
- [19] A. Tiwari, B. Ensing, *Faraday Discuss.* **2016**, *195*, 291.
- [20] J. Hutter, M. Iannuzzi, F. Schiffmann, J. VandeVondele, *WileyWIREs Comput Mol Sci* **2014**, *4*, 15.
- [21] T. S. van Erp, M. Moqadam, E. Riccardi, A. Lervik, *J. Chem. Theory Comput.* **2016**, *12*, 5398.
- [22] T. S. van Erp, T. P. Caremans, C. E. A. Kirschhock, J. A. Martens, *Phys. Chem. Chem. Phys.* **2007**, *9*, 1044.
- [23] E. Riccardi, O. Dahlen, T. S. van Erp, *J. Phys. Chem. Lett.* **2017**, *8*, 4456.
- [24] T. S. van Erp, *Epl* **2023**, *143*, 30001.
- [25] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, *J. Chem. Phys.* **1953**, *21*, 1087.
- [26] W. K. Hastings, *Biometrika* **1970**, *57*, 97.
- [27] D. Moroni, P. G. Bolhuis, T. S. van Erp, *J. Chem. Phys.* **2004**, *120*, 4055.
- [28] T. S. van Erp, D. Moroni, P. G. Bolhuis, *J. Chem. Phys.* **2003**, *118*, 7762.
- [29] R. H. Swendsen, J. S. Wang, *Phys. Rev. Lett.* **1986**, *57*, 2607.
- [30] E. Marinari, G. Parisi, *Europhys Lett.* **1992**, *19*, 451.
- [31] Y. Sugita, Y. Okamoto, *Chem. Phys. Lett.* **1999**, *314*, 141.
- [32] C. Dellago, P. G. Bolhuis, D. Chandler, *J. Chem. Phys.* **1998**, *108*, 9236.
- [33] D. Frenkel, B. Smit, *Understanding Mol. Simul.*, 2nd ed., Academic Press, San Diego, CA **2002**.
- [34] A. Ghysels, S. Roet, S. Davoudi, T. S. van Erp, *Phys. Rev. Res.* **2021**, *3*, 033068.
- [35] A. M. Ferrenberg, R. H. Swendsen, *Phys. Rev. Lett.* **1989**, *63*, 1195.
- [36] J. Rogal, W. Lechner, J. Juraszek, B. Ensing, P. G. Bolhuis, *J. Chem. Phys.* **2010**, *133*, 174109.
- [37] S. Davoudi, A. Ghysels, *J. Chem. Phys.* **2021**, *154*, 054106.
- [38] A. Krämer, A. Ghysels, E. Wang, R. M. Venable, J. B. Klauda, B. R. Brooks, R. W. Pastor, *J. Chem. Phys.* **2020**, *153*, 124107.
- [39] R. M. Venable, A. Krämer, R. W. Pastor, *Chem. Rev.* **2019**, *119*, 5954.
- [40] A. Ghysels, A. Krämer, R. Venable, W. Teague, E. Lyman, K. Gawrisch, R. W. Pastor, *Nat. Commun.* **2019**, *10*, 5616.
- [41] A. Ghysels, R. M. Venable, R. W. Pastor, G. Hummer, *J. Chem. Theory Comput.* **2017**, *13*, 2962.
- [42] O. De Vos, R. M. Venable, T. Van Hecke, G. Hummer, R. W. Pastor, A. Ghysels, *J. Chem. Theory Comput.* **2018**, *14*, 3811.
- [43] O. Aarøen, H. Klær, E. Riccardi, *J. Comput. Chem.* **2021**, *42*, 435.
- [44] R. Xu, D. Wunsch, *IEEE Trans. Neural Netw* **2005**, *16*, 645.
- [45] T. K. Ho, Random decision forests, Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, pp. 278–282 vol.1.
- [46] K. Pearson, *Proc R Soc London* **1895**, *58*, 240.
- [47] O. Dahlen, A. Lervik, O. Aarøen, R. Cabriolu, R. Lyng, T. S. van Erp, *Comput. Appl. Eng. Educ.* **2020**, *28*, 779.

- [48] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python. <http://www.scipy.org> **2001**.
- [49] S. Roet, D. T. Zhang, T. S. van Erp, *J Phys Chem A* **2022**, *126*, 8878.
- [50] S. Plimpton, *J. Comput. Phys.* **1995**, *117*, 1.
- [51] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, V. S. Pande, *J. Chem. Theory Comput.* **2013**, *9*, 461.
- [52] SCM, *Theoretical Chemistry*, Vrije Universiteit, Amsterdam, The Netherlands, Ams **2023**.
- [53] T. P. Senftle, S. Hong, M. M. Islam, S. B. Kylasa, Y. Zheng, Y. K. Shin, C. Junkermeier, R. Engel-Herbert, M. J. Janik, H. M. Aktulga, T. Verstraelen, A. Grama, A. C. T. van Duin, *Appl Future Directions Npj Comput Mater* **2016**, *2*, 15011.
- [54] J. G. Brandenburg, S. Grimme, *J. Phys. Chem. Lett.* **2014**, *5*, 1785.
- [55] E. Riccardi, S. Pantano, R. Potestio, *Interface Focus* **2019**, *9*, 20190005.
- [56] D. T. Zhang, L. Baldauf, S. Roet, A. Lervik, T. S. van Erp, *Proc. Natl. Acad. Sci. U. S. A.* **2024**, *121*, e2318731121.
- [57] J. Behler, M. Parrinello, *Phys. Rev. Lett.* **2007**, *98*, 146401.
- [58] A. P. Bartók, M. C. Payne, R. Kondor, G. Csányi, *Phys. Rev. Lett.* **2010**, *104*, 136403.
- [59] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kombluth, N. Molinari, T. E. Smidt, B. Kozinsky, *Nat. Commun.* **2022**, *13*, 2453.
- [60] A. Ma, A. R. Dinner, *J. Phys. Chem. B* **2005**, *109*, 6769.
- [61] H. Jung, R. Covino, A. Arjun, C. Leitold, C. Dellago, P. G. Bolhuis, G. Hummer, *Nat. Comput. Sci.* **2023**, *3*, 334.
- [62] F. G. Wang, D. P. Landau, *Phys. Rev. Lett.* **2001**, *86*, 2050.
- [63] A. Laio, F. L. Gervasio, *Rep. Prog. Phys.* **2008**, *71*, 126601.

**How to cite this article:** W. Vervust, D. T. Zhang, A. Ghysels, S. Roet, T. S. van Erp, E. Riccardi, *J. Comput. Chem.* **2024**, *45*(15), 1224. <https://doi.org/10.1002/jcc.27319>

## APPENDIX A: RECURSIVE BLOCK ERROR ANALYSIS

Considering Equations (1)–(3), it is easy to show that recursive block values,  $k_j^r$ , are identical to standard blocks averages,  $k_j^s$ , in case that the running estimate  $k[n]$  would be a true running average (not only a ‘running estimate’): a simple average of  $n$  data points. However, the rate estimate  $k[n]$  is not a running average in the strict sense. Instead, it is a running estimate, and it is a result obtained from different ensemble averages such as the path lengths of ensembles  $[0^-]$  and  $[0^+]$ , and the local crossing probabilities  $P_A(\lambda_{i+1}|\lambda_i)$  of ensemble  $[i^+]$  with  $i \geq 0$ . In that case, the recursive block values and the standard block values may not necessarily be identical. In this context, the term ‘the  $j$ th standard block value’ denotes the specific value derived by segregating each relevant dataset into blocks and specifically examining the information within the  $j$ th block of each dataset to compute the intended property. This is denoted by the superscript  $s$  in  $k_j^s$ , distinguishing it from a recursive block,  $k_j^r$ . When the recursive and standard blocks coincide (i.e., when  $k_j^r = k_j^s$ ), we represent this as  $k_j^r = k_j^s = k_j$ , omitting the superscript entirely.

To discuss the recursive block approach in a more generic context, let us assume that the evaluation of  $k[n]$  is based on different ensemble averages  $a[n], b[n], c[n], \dots$ , that is,  $k[n] = f(a[n], b[n], c[n], \dots)$

where  $f(\cdot)$  is the function that provides the rate from the ensemble averages. For these ensemble averages  $a[n]$ , and so forth, the running estimates are plain running averages and there is no difference between recursive and standard blocks:  $a_j^r = a_j^s = a_j$  and so forth.

However,  $k_j^s = f(a_j, b_j, c_j, \dots) \neq k_j^r$  for  $j > 1$ . Yet, for large enough blocks we can assume that each block  $j$  is close to its exact value  $a, b, c$ :  $a_j = a + \delta a_j$ ,  $b_j = b + \delta b_j$ ,  $c_j = c + \delta c_j$  such that in first order of  $\delta$ :

$$k_j^s = f(a_j, b_j, c_j, \dots) \approx f(a, b, c, \dots) + \left(\frac{\partial f}{\partial a}\right) \delta a_j + \left(\frac{\partial f}{\partial b}\right) \delta b_j + \left(\frac{\partial f}{\partial c}\right) \delta c_j + \dots \quad (\text{A1})$$

where the derivatives are evaluated at the exact value  $a, b, c$ . Now, let us compare this to the Taylor expansion of the  $j$ th recursive block starting from Equation (3):

$$\begin{aligned} k_j^r &= jf(a[jm], b[jm], c[jm], \dots) \\ &\quad - (j-1)f(a[(j-1)m], b[(j-1)m], c[(j-1)m], \dots) \\ &= jf\left(\frac{1}{j} \sum_{i=1}^j a_i, \frac{1}{j} \sum_{i=1}^j b_i, \frac{1}{j} \sum_{i=1}^j c_i, \dots\right) \\ &\quad - (j-1)f\left(\frac{1}{j-1} \sum_{i=1}^{j-1} a_i, \frac{1}{j-1} \sum_{i=1}^{j-1} b_i, \frac{1}{j-1} \sum_{i=1}^{j-1} c_i, \dots\right) \\ &= jf\left(a + \frac{1}{j} \sum_{i=1}^j \delta a_i, b + \frac{1}{j} \sum_{i=1}^j \delta b_i, c + \frac{1}{j} \sum_{i=1}^j \delta c_i, \dots\right) \\ &\quad - (j-1)f\left(a + \frac{1}{j-1} \sum_{i=1}^{j-1} \delta a_i, b + \frac{1}{j-1} \sum_{i=1}^{j-1} \delta b_i, c + \frac{1}{j-1} \sum_{i=1}^{j-1} \delta c_i, \dots\right) \end{aligned}$$

Applying Taylor expansions to  $f$  in this equation and simplifying the expression, we find

$$k_j^r \approx f(a, b, c, \dots) + \left(\frac{\partial f}{\partial a}\right) \delta a_j + \left(\frac{\partial f}{\partial b}\right) \delta b_j + \left(\frac{\partial f}{\partial c}\right) \delta c_j + \dots + \mathcal{O}(\delta^2) \quad (\text{A2})$$

From Equations (A1) and (A2), it is evident that the Taylor expansions are equivalent up to the first order in  $\delta$ . This suggests that with an increase in block length, the recursive blocks converge toward the properties of standard blocks, validating our approach. It is important to realise that truncating the Taylor expansion to the first order of  $\delta$  aligns with the standard error propagation practice, ensuring that our approach does not introduce any additional approximations beyond those already implied in other accepted methods.

It is interesting to note that the approach mentioned here, that is, running estimates and recursive blocks, can be effectively integrated with methods like Wang-Landau<sup>62</sup> or metadynamics.<sup>63</sup> In these techniques, a bias dynamically adapts throughout the sampling process until it converges and stabilises. Hence the bias is non-stationary and changes in the running estimate and in the running block averages. Nevertheless, as the sampling duration extends, the statistical impact of the bias' non-stationarity gradually diminishes. Therefore, when the number of MC cycles  $N$  becomes sufficiently large, the effect of non-stationarity will only affect a small fraction of the blocks or only influence a minor part of the initial block. This justifies the applicability of the recursive block method even in such adaptive biasing methods.