PyRETIS 2: An Improbability Drive for Rare Events

Enrico Riccardi (a),* Anders Lervik (a), Sander Roet (a), Ola Aarøen (b) and Titus S. van Erp (a,c)

The algorithmic development in the field of path sampling has made tremendous progress in recent years. Although the original transition path sampling method was mostly used as a qualitative tool to sample reaction paths, the more recent family of interface-based path sampling methods has paved the way for more quantitative rate calculation studies. Of the exact interface-based path sampling methods, the most efficient, but rather difficult to implement, has been the main motivation to develop the open-source Python-based computer library PyRETIS that was released in 2017. PyRETIS is designed to be easily interfaced with any molecular dynamics (MD) package using either classical or ab initio MD. In this study, we report on the principles and the software enhancements that are now included in PyRETIS 2, as well as the recent developments on the user interface, improvements of the efficiency via the implementation of new shooting moves, easier initialization procedures, analysis methods, and supported interfaced software. © 2019 The Authors. *Corresponding author: enrico.riccardi@ntnu.no

Introduction

Simulation of long time scales has been a major challenge in molecular simulations. Although the increase of system scale is straightforwardly parallelizable, extending simulation time is not. This is a severe problem as molecular dynamics (MD) typically requires femtoseconds time steps. Even if the computational evaluation of such a step is usually achieved within a fraction of second, it would take centuries of CPU time to compute 1 s of real time. This makes it nearly impossible to study processes like chemical reactions, phase transitions, and conformational changes with MD. These processes typically occur so infrequently that years of computer time are needed to observe even a single event.

The vast majority of methods, which have been developed for overcoming this problem, either alter the potential energy surface or the dynamics of the system (see, e.g., Refs. [1,2]). The use of Monte Carlo (MC) in path space is an approach that does not disturb the underlying physical dynamics, but generates unlikely molecular events like an improbability drive.[3,4] This approach is the essence of transition path sampling (TPS)[5–7] in which repetitively a new path is being generated from an old path via MC moves that obey detailed balance. The most important MC move is the so-called shooting move in which first a random time slice (comprising the phase point at a certain MD step) of the old path is selected and then stochastically modified. For instance, random disturbances could be applied to the velocities of that point. After that, this point is first propagated backward and then forward in time using a standard symplectic and time-reversible integrator for MD, Langevin, or Brownian dynamics. Moreover, as paths consist of time slices (phase points at discrete time steps), velocity Verlet,[8] and other reversible integration schemes should be preferred above leapfrog[9] as the latter provides velocities that are shifted in time by half a time step.[10] After the completion of these time integrations, the forward and backward trajectories are glued together resulting in a new continuous path that follows the natural dynamics of the system. This path is finally accepted or rejected based on a Metropolis–Hastings rule.[11,12]

Transition interface sampling (TIS)[13] introduced several fundamental key elements that made efficient quantitative path sampling possible. Firstly, TIS introduced the statistical path ensemble with flexible path length, reducing the redundant exploration of the stable regions. Secondly, a series of path ensembles were defined based on a set of hyperplanes (interfaces). These hyperplanes are generally defined by a value of the order parameter (reaction coordinate/progress coordinate) which is a function of the coordinates (and possibly velocities) of the system.

The first interface, called $\lambda_0$ or $\lambda_{0b}$, defines the region of the initial state. The last interface, called $\lambda_f$ or $\lambda_{fb}$, defines the region of the final state. The first interface is placed such that a straightforward MD simulation starting from the reactant side would cross this interface sufficiently frequently, which enables the determination of the flux through this plane by straightforward MD. The last interface is placed sufficiently far across the barrier so that each trajectory from $\lambda_0$ to $\lambda_f$ will not easily return to the

[a] E. Riccardi, A. Lervik, S. Roet, T. S. van Erp
Department of Chemistry, Norwegian University of Science and Technology, Høgskolen 5, 7491 Trondheim, Norway
E-mail: enrico.riccardi@ntnu.no

[b] O. Aarøen
Department of Biotechnology and Food Science, Norwegian University of Science and Technology, Høgskolen 5, 7491 Trondheim, Norway

[c] T. S. van Erp
Center for Molecular Modeling (CMM), Ghent University, Technologiepark 903, 9052 Zwijnaarde, Belgium

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2019 The Authors. *Corresponding author: enrico.riccardi@ntnu.no

Journal of Computational Chemistry published by Wiley Periodicals, Inc.
initial state \( A \) and, therefore, can be considered as a successful transition from the initial state to the final state. As long as the positioning of \( \lambda_0 \) and \( \lambda_i \) is reasonable and respects the above criteria, the final result will not largely be affected by their exact placement. The other interfaces are defined in between \( \lambda_0 \) and \( \lambda_i \) and their positions are solely based on efficiency arguments.

Once the interfaces are defined, the TIS rate equation is then given by

\[
\begin{align*}
    k_{AB} &= f_A \mathcal{P}_A(\lambda_B|\lambda_A) \\
    \mathcal{P}_A(\lambda|\lambda_i) &= \prod_{i=0}^{n-1} \mathcal{P}_A(\lambda_{i+1}|\lambda_i)
\end{align*}
\]  

(1)

where \( f_A \) is the flux through \( \lambda_A \) and \( \mathcal{P}_A(\lambda_B|\lambda_A) \) is the very small probability that a crossing with \( \lambda_A \) will lead to a crossing with \( \lambda_B \) without recrossing \( \lambda_A \). As this probability is generally extremely small, it cannot be calculated directly. However, it can be computed by the exact factorization in the second expression in eq. (1). One should realize that \( \mathcal{P}_A(\lambda_{i+1}|\lambda_i) \) is not just the probability to go from \( \lambda_i \) to \( \lambda_{i+1} \), but rather a history-dependent conditional probability.

Replica exchange TIS (RETiS) samples all path ensembles in parallel and applies replica exchange moves between those.\(^{[14]}\)

In addition, it replaces the MD simulation for the flux calculation with the ensemble \([0^*] \), which consists of paths that start and end at \( \lambda_0 \) but explore the reactant well region rather than the reaction barrier. For these features, RETIS is considerably more complex to implement in computer codes. This has been the main driving force to develop the open source PyRETI{\textsuperscript{5}5} library.

A year after the official disclosure of PyRETI{\textsuperscript{5}5}, the Open Path Sampling (OPS) library was released.\(^{[16,17]}\) The aims of PyRETI{\textsuperscript{5}5} and OPS are similar, but the libraries have been written with slightly different user communities in mind. OPS is more generic and allows the expert user to design different path ensembles. PyRETI{\textsuperscript{5}5}, on the other hand, has a stronger focus on the RETIS algorithm and a stronger emphasis on user-friendly accessibility (i.e., toward the nonexpert user). There are presently active collaborations between the two developer groups, which potentially could lead to a partial, or even full, merger of the two libraries in the future.

In this article, we discuss the code developments made in the release of PyRETI{\textsuperscript{5}5} 2. PyRETI{\textsuperscript{5}5} 1 was interfaced with GROMACS\(^{[18]}\) and CP2K\(^{[19]}\) for respectively, performing classical and ab initio MD. In PyRETI{\textsuperscript{5}5} 2, we improved the GROMACS interface and added interfaces with OpenMM\(^{[20,21]}\) and LAMMPS.\(^{[22]}\) Several structural improvements have been made to improve the readability and reliability of the code. The major ones are shortly described in this study. To improve the efficiency, the new MC moves in path space, Stone Skipping and Web Throwing developed by Riccardi et al.\(^{[23]}\) have been implemented. An easier initialization procedure has also been introduced such that trajectories, or simple snapshots, can now directly be read by PyRETI{\textsuperscript{5}5} 2 and used to initialize the RETIS simulation. In terms of outputs, a graphical user interface (GUI) to quickly inspect trajectories and density plots as functions of different descriptors (collective variables) has been constructed and added to the library.

### Algorithmic Improvements

#### Advanced path generating MC moves

To optimize the sampling efficiency, PyRETI{\textsuperscript{5}5} 2 allows a direct and intuitive selection of path sampling strategies. Two of the most promising and recent sampling methods, namely, Stone Skipping and Web Throwing,\(^{[23]}\) have now also been included in the code. Stone skipping and web throwing are two advanced MC-based moves that reduce correlations between successively generated paths. This implies that fewer generated trajectories are required to estimate crossing probabilities and rate constants within a desired error range. Therefore, despite that the execution of a single MC step is more costly than standard shooting, the sampling efficiency can increase considerably (for the case study of Ref. [23], the increase in efficiency was more than an order magnitude).

We expect that these new moves will become the default choice as they are intrinsically faster than standard shooting, though it will require some adaptations with how PyRETI{\textsuperscript{5}5} presently handles external engines, before this will be paid off in practice. The essential aspect of the new MC moves is that they launch a sequence of short subpaths via a shooting protocol that shoots from the ensemble-specific interface, that is, from a time slice just before or after the interface. The shooting move for creating a subpath propagates in one time direction only, though requires to cross the interface in one single time step backward in time or forward in time, depending on whether the selected shooting point is a point just beyond or before the interface. If this single-step crossing condition is not fulfilled, new random velocities should be generated until the condition is met. While this single-step crossing condition can be verified in principle for a single MD step without doing an actual force calculation,\(^{[23]}\) the single-step crossing test has to be carried out explicitly whenever a (PyRETI{\textsuperscript{5}5}) step consist of several MD steps generated by the external engine. This makes the repeated generation of random velocities, followed by the testing of the one-step crossing condition, potentially, a very expensive element of this MC move. If the dynamics is sufficiently stochastic, then this one-step crossing test can be avoided by maintaining the same two time slices before and after the interface and create subpaths via one-way shooting protocol from the point that is after the interface, that is, without changing the velocities. For deterministic dynamics and dynamics that is only moderately stochastic (e.g., underdamped Langevin dynamics), the new MC moves might not yet outperform standard shooting until a new approach for handling the external engines has been implemented.

#### The \([0^*] \) ensemble with additional confining interface

The \([0^*] \) ensemble was introduced by van Erp\(^{[14]}\) to replace the MD simulation for computing the flux in eq. (1) and to allow for replica exchange moves between all path ensembles. Paths in the \([0^*] \) ensemble start at \( \lambda_0 \) like all other paths in the other ensembles, but move away from the barrier exploring the reactant well. The path is terminated once it recrosses \( \lambda_0 \). As a result, the time slices of a valid path in the \([0^*] \) path ensemble
The trajectories depicted in the (λ, CV) plane generated via the shooting move in the [0⁺] ensemble, where CV is an additional collective variable different from λ. a) The standard situation of systems for which RETIS was originally designed. The free energy provides a natural boundary at the left side, which implies that the λ coordinate cannot get much smaller than λ₀ and respective path generation trials (indicated by the numbers 1, 2, and 3) starting from the shooting points (red circles) will relatively quickly end at λ₀ in the backward and forward time direction, yielding new acceptable paths. This situation is, for instance, typical for bond breaking, nucleation, and protein folding. b) The situation for which the additional λ₁ interface was introduced and is typical in, for example, permeation studies. In this case, the free energy does not provide a natural boundary at the left. Trajectories can in principle continue to move toward the left, allowing endlessly long trajectories. PyRETIS 2, however, allows the user to define a λ₁ interface, such that the shooting moves hitting the λ₁ interface are directly rejected. This is the case for trials 2 and 4 which hit the λ₁ interface in the backward and forward time direction, respectively. [Color figure can be viewed at wileyonlinelibrary.com]

have order parameter values <λ₀ except for the start and end points.

This assumes that the reactant well has a natural confinement at the left side of the barrier, either because the potential energy increases when the order parameter is decreased below the equilibrium position of the reactant state, or because of periodic boundaries. For instance, if the order parameter is (minus) the distance between two molecules that can react if they approach, the periodic boundaries will ensure that their separation is bounded. However, if the box dimensions are large or if another undesired stable state exists at the left side of the reactant well, it might be needed to terminate the path early, as a completed path starting and ending at λ₀ could be exceedingly long. This can be done in PyRETIS 2 by setting an extra boundary, a “minus one” interface λ₋₁ with λ₋₁ < λ₀, that ensures that the reactant state ensemble is restricted between λ₋₁ and λ₀. In the present implementation, a path is rejected when it hits this left boundary. Please note that the use of a left boundary on this ensemble results in an incorrect flux calculation. A more sophisticated approach will be implemented in a future release in which the flux calculation will account for the presence of the extra boundary. At the present, the usage of the left boundary requires a reevaluation of flux term by means of another simulation without the left boundary. An illustrative scheme of the λ₋₁ is reported in Figure 1.

**Interface with External Engines**

The PyRETIS 1 program provided interfaces with GROMACS and CP2K. In PyRETIS 2, we have extended this with LAMMPS and OpenMM. In addition, some fundamental changes related to the GROMACS interface has been established in PyRETIS 2. These new developments are shortly discussed below. Working examples for each external engine can be found in the website (section: Getting started). The external engines communicate with PyRETIS at a certain frequency, defined by the subcycles number in the PyRETIS input file in the engine section. At each subcycles number, the external engine provides the order parameter or the information for its computation. Its magnitude depends on the selected engine and on the system under investigation (trade between speed, accuracy, and storage requirements).

**GROMACS**

The GROMACS engine has been updated for PyRETIS 2. The strategy for the GROMACS engine in PyRETIS 1 relied on repeatedly starting and stopping the execution of GROMACS. That is, PyRETIS 1 executes GROMACS for a few MD steps, defined by the number of subcycles, obtains the order parameter and uses this to determine if the GROMACS run should be ended or continued. PyRETIS 2 still not only supports this, but also provides a potentially more efficient strategy. It will execute GROMACS and obtain the order parameter while GROMACS is running and will use this to determine when it is time to end the GROMACS run. This increases the efficiency of running GROMACS with PyRETIS as it reduces the number stop/restart calls to the GROMACS engine. To select the old approach, the engine class to select in the input file is gromacs, while the latter method is called by the gromacs2 keyword. The new approach exploits the functionality of the MDTraj\[241 library for efficiently reading GROMACS trajectory files.
PyRETISS 2 also includes an interface for LAMMPS.\footnote{22} For this engine, PyRETISS 2 will only provide information about the initial configuration and the stopping conditions (i.e., the maximum number of steps to perform and the location of the relevant interfaces). This enables the execution of LAMMPS to be fully handled by LAMMPS itself. This also requires that the order parameter is defined in a separate LAMMPS input file, which LAMMPS can use to calculate it. After the completion of a
LAMMPS MD run, PyRETISS 2 will read the calculated order parameter and energies, and store the generated trajectory. Currently, PyRETISS 2 only supports microcanonical (NVE) dynamics when using LAMMPS. Note that temperature effects can still be studied as the temperature is linked to the MC sampling. That is, paths describe dynamics at constant energy, but the energies of different paths will fluctuate due to the shooting move. The approach can be defined as a canonical (NVT) sampling of NVE paths, and it is a popular approach in path sampling since it removes the dynamics from unphysical modifications of the equations of motion by a thermostat, while still sampling the canonical distribution. It reflects a system that is weakly coupled to a heat bath such that its effect is not noticeable on the time scale of the path length.

OpenMM

PyRETISS 2 has added an interface with OpenMM.\footnote{21} In this version, OpenMM can only be used with PyRETISS as a library. That means that if an OpenMM Simulation class is initialized, this object can then be used to initiate the OpenMMEngine class inside PyRETISS. This OpenMMEngine class can then be used for all the PyRETISS internals. An automated setup from restructured text input, like the way PyRETISS handles the connection with the other engines, is not yet supported and will be added in a later version of PyRETISS. As OpenMM also supports running on GPUs, special care was taken to not create new OpenMM contexts, but instead update the coordinates and velocities by an in-place operation. This minimizes the communication and prevents unnecessary compilation time for running on GPUs. The current implementation is only suitable for simulation in the canonical ensemble. Support for the isothermal–isobaric (NPT) will be added to a later version of PyRETISS.

Library Structure

Ensemble structure

The paths generated by Path Sampling are grouped into ensembles. Each of them focuses on a different region of path space. Each one can rely on different MC rules to generate new paths. Dedicated setups, tailored to the region to explore, can thus improve the sampling by enabling the application of the most suitable techniques. These possible techniques are for instance the use of Stone Skipping Web Throwing moves (see “Advanced path generating MC moves” section) or the different ways to disturb the velocities when performing a shooting move. To use this feature, a user should simply declare the

Simulation
-----------
task = retis
steps = 10000
interfaces = [ -0.9, -0.6, -0.3, 0.0, 1.0]

 Ensemble
----------
interface = -0.9
shooting_move = sh

 Ensemble
----------
interface = -0.6
shooting_move = ss
n_jumps = 16

 Ensemble
----------
interface = -0.3
shooting_move = wt
n_jumps = 4
interface_source = -0.5

Figure 2. New input structures to insert specific input for an ensemble. Each ensemble section refers to an interface specified in within. In the example, for the three ensembles selected, different shooting moves, with different parameters, have been selected. Note that the default shooting move is “sh,” the first ensemble section is, therefore, not required.

Defining the order parameter and additional collective variables

The input file to PyRETISS has been updated to directly support several collective variables. This implies that a set of additional collective variables can be listed in the input file and those will be calculated along with the main order parameter. These collective variables do not affect the RETIS algorithm but can provide valuable information for the analysis of the path ensembles. The additional collective variables are hence descriptors to be used in postprocessing to elucidate mechanisms occurring in the investigated transition. There are, therefore, no constraints on the number or type of collective variables. The new input scheme to include additional collective variables is illustrated in Figure 3.

Paths storage and restart reproducibility

The storage of generated paths has been updated for PyRETISS 2. The trajectories can be saved with any frequency in a compressed format. The respective order parameter and energies (as a function of time) are saved in separate files with arbitrary frequencies too. This setup allows independent visual inspections of the generated trajectories and the restart of a new path
Order parameter
-------------
class = Position
dim = x
index = 0
periodic = False

Collective-variable
---------------------
class = Velocity
dim = y
index = 16

Collective-variable
---------------------
class = Angle
index = 1, 3, 7

Collective-variable
---------------------
class = Custom
module = orderp.py
index = 11, 13, 17, 22

Figure 3. New input structures to include multiple collective variables along with the main order parameter. The order of the collective variables will be maintained in the generated output by PyRETIS in the orderp.txt files. In this example, the order parameter is the x position of atom 0, the first collective variable is the y component of the atom 16 velocity. The second collective variable is the angle between the atoms with indexes 1, 3, and 7. While these descriptors are computed with internal functions, the latter order parameter, called Custom, is an example of an externally computed descriptor (located in the module orderp.py).

Random number generators
A new treatment of random number generators has been implemented in PyRETIS 2 to allow an exact reproducibility of simulations even if they have been performed in parallel. The random number generator is called at many places in the RETIS algorithm: In each cycle, a random number is drawn to select (1) the RETIS move (to select between the options swap/time-reversal/shooting or other types of path MC moves), (2) the relative shooting (if applicable), (3) the selection of the frame index, (4) the selection of the new velocities (velocities may be kept unchanged in the case of stochastic dynamics), and (5) the random forces whenever stochastic dynamics is applied.

In PyRETIS 2, each task has a dedicated and unique random number generator. The feature permits a more accessible reproduction of the simulation results and a precise continuation of simulations even in parallel jobs.

MDTraj
PyRETIS 2 includes MDTraj as an interpreter for external files. That is, external trajectories (from GROMACS, CP2K, LAMMPS) can be read with this Python library that was developed to deal efficiently with massive trajectories. The aim is to gain, in later releases of PyRETIS, increased independence from the external engine. As the minimal output that PyRETIS needs to receive from the external engine is only an ordered list of order parameters describing a trajectory, a universal interpreter to read external files in the various formats would simplify and uniform the interface for the various external engines. In PyRETIS 2, MDTraj is used in the load function to extract the desired frames. The package allows the use and development of arbitrary external functions to compute the order parameters and additional collective variables.

Input/Output Schemes
Load functionality and initialization
A new functionality, the load feature, has been added to PyRETIS 2 that simplifies one of the most user demanding tasks of the path sampling algorithm, the initialization of the simulation. The feature permits a direct initialization of path sampling simulations using configuration frames that could be generated by any type of fast simulation method and software. The new load function reads frames and trajectories supplied to PyRETIS 2 without the need for any further descriptor. PyRETIS 2 will compute the order-parameter, the additional collective variables and eventually the energy of the provided frames and trajectories. The input information will be automatically rearranged to satisfy the various ensemble definitions, when possible. PyRETIS 2 will then start the exploration of the path space from these initial frames indicating, with the “Id” flag in the output “pathensemble.txt” files, that the latest accepted path is a repetition of the initial path loaded. Once the initialization is completed this “Id” flag should no longer be present in the newly produced output lines in the “pathensemble.txt” files.

The strategy allows the inclusions of frames along the transition of interest that can be constructed by for instance constrained dynamics, nudge elastic band (25) or metadynamics (2) using any type of software. It should be underlined that the load function only simplifies the initialization procedure and will not influence the final converged result. It is, therefore, possible to provide to the load function a hypothetical path that has no physical meaning. The load initialization procedure also has a preprocessing feature to limit the overall simulation memory requirements. That is, for a massive trajectory, that explores the transition region only in a relatively short period, PyRETIS 2 automatically omits the frames that are not part of the ensemble of interest.
The load function can also be invoked after a restart. The frames from the restart file are then used as frames for the load method. The difference compared to a normal restart, which is a continuation, is that the load permits a broad change of the simulation parameters: from the order parameter selection to the interface positioning. It should be reminded here that, unlike a normal restart, the use of load with concomitant changes in the simulation parameters, should not be viewed as a continuation, and the previous sampling should be discarded in the final analysis.

Visualization

As specified by the user, different output files are created and also the frequency of writing the data can be set. Data sets can be created by optionally writing all-atom coordinates, velocities, and a list of descriptors (collective variables) for each trajectory at each time slice. Writing of all coordinates and velocities gives rise to huge data sets which asks for significant memory requirements for storage, and complicates the interpretation and data analysis. Therefore, we advise the user to reflect on what may be potentially important collective variables and then define a large set of descriptors prior to performing production runs. This simplification reduces the dimensionality of the data on which the interpretation of the reaction mechanisms shall be based. Still, even with this reduced dimensionality it can be nontrivial to filter out the essential information.

To facilitate this task, PyRETIS 2 includes a new visualization tool, named PyVisA. The software permits an almost immediate visualization of the initial, partial and final simulation results by allowing the automated generation of plots of various simulation collective variables. A user can prompt plot energies, the order parameter, collective variables, cycle number, and path lengths as a function of each other in different type of plots, for different ensembles, for selected cycle ranges, for accepted or rejected paths. A GUI has been constructed facilitate this visualization and to easily navigate through the PyRETIS 2 outputs.

The visualization of the various descriptors, for example, the collective variable density plots, in the GUI allows the user to interactively inspect different parameter combinations, potentially revealing additional information about the system dynamics. In the initialization stage, a user can better position the various interfaces, select the most appropriate MC move (e.g., standard shooting, Stone Skipping, Web Throwing, etc.), determine metastable states and even evaluate the efficacy of the selected order parameter in comparison with other collective variables. The descriptors include order parameters, collective variables, energy, number of simulation steps for path, RETIS cycles. The user can select the range of cycles to visualize, their type (accepted/rejected) and the plot type (e.g., 2D scatter, 3D scatter, local density). Figures 5 and 4 show two reports that can be obtained by the visualization tool’s GUI. To execute the visualization and analysis tool (named “PyVisA”), the flag -pyvisa shall be added to the pyretisanalyse command. Further details, instructions, and examples can be found in Ref. [27], where the tool structure and features are detailed.

The visualization tool, furthermore, provides a structured computational framework that will permit a general implementation of advanced analysis approaches. Predictor methods or machine learning methods to evaluate the quality of the selected order parameter in the description of the sampled event are the two most immediate examples.

User Support

Compatibility and installation

PyRETIS 2 supports Python 3.6 and 3.7 and is distributed via pip and conda (via the conda-forge channel) for the main releases. The visualization package, PyVisA, can be found in the development versions of PyRETIS (PyRETIS v2.develop and PyREITS 3. beta) and in the forthcoming releases beyond Version 2.4. The development versions can be installed by downloading the PyRETIS source code from gitlab via the command “python setup.py install” from the main directory. Further details on its installation and usage can be found in Ref. [27].

Test examples

Along with a long list of unit tests, the PyRETIS development is also tested versus a series of main test simulations that are automatically executed daily to assert the code and its
dependencies status. These test simulations are available in the development versions of PyRETIS (which can be installed via git as described in the section on code availability). The test simulations have also a pedagogical purpose: users can gain experience and familiarity with the PyRETIS input scheme. These test simulations are available in the development version are accessible at: https://gitlab.com/pyretis/pyretis. The source code, the visualization tool and working examples for each of the external engines supported.

Cases of study performed with PyRETIS are constantly uploaded on the PyRETIS web page under the “Main studies performed by PyRETIS” section, and the files used to initiate and control the simulations are shared to facilitate the reproducibility of our investigations, with respect to FAIR data policies.[30]

In particular, our recent investigation of water autoionization[31] and the Histone-like Nucleoid Structuring protein (H-NS) binding to DNA[26] are listed on the website. They constitute two successful applications of the RETIS algorithm, where the sampling software has been interfaced with CP2K[19] in the first work, and GROMACS[18] in the latter. The User Guide section contains a new set of entries to facilitate the usage of the code. The guide comprises (1) instructions for the installation of PyRETIS in a user and a developer mode, (2) information on how to use and set up external engines and order parameters, (3) help for common errors, and (4) instructions on how to report bugs.

Future Work

Despite the considerable efforts put in the code development, there are various expansions that would be desirable to increase usability, efficiency, and compatibility with other sampling software. We aim to automatize some of the parameter selections such as the interface positions, the number of jumps in the stone skipping and web throwing moves and the SOUR interface[23] in web throwing, the relative shooting weights, and the frequency of selection of the various MC moves.

An interface with VASP has been initiated and partially completed. It will be released after completion and after performing sufficient testing. In parallel, we are considering the implementation of a “translation” platform that might enable a single scheme to deal with the input and output of multiple engines. Besides the already implemented MDTraj,[24] Python packages such as MDAnalysis[32,33] and Atomic Simulation Environment (ASE)[34] can facilitate the realization of such a platform.

In the next PyRETIS release, the visualization tool will become an integral part of the code. It will provide a multidimensional and structured analysis framework. Advanced analysis approaches, such as the predictive power analysis method[28] and machine learning based methods to evaluate the quality of different collective variables,[29] will be readily implemented. We are therefore interested in direct support and collaboration with potential new developers that are interested to apply and expand PyRETIS.

Software Availability

PyRETIS 2 is free (released under a LGPLv2.1+ license) and can be obtained as described previously and at http://www.pyretis.org/user/install.html. The source code, the visualization tool and the development version are accessible at: https://gitlab.com/pyretis/pyretis.
Acknowledgments

The authors thank the Research Council of Norway for funding (project nos: 250875 and 267669), NOTUR for providing HPC facilities (project no.: NN9254K), the Olav Thon foundation for the support in the development of interactive visualization tools for teaching and the Peder Sather Center for the support in the development of the LAMMPS-PyRETIS interface. The authors also thank the Lorentz Center for supporting our workshop on PyRETIS and Path Sampling in Leiden, March 11–15, 2019. Sudi Jawahery, Anastasia Maslechko, Raffaela Cabriolu, An Ghysels, Jocelyne Vreede, Christopher Daub, and Mahmoud Moqadam are thanked for their feedbacks and useful discussions.

Keywords: PyRETIS · rare event · path sampling · python · kinetics

How to cite this article: E. Riccardi, A. Lervik, S. Roet, O. Aarøen, T. S. van Erp. J. Comput. Chem. 2020, 41, 370–377. DOI: 10.1002/jcc.26112

References

[1] F. G. Wang, D. P. Landau, Phys. Rev. Lett. 2001, 86, 2050.
[2] A. Laio, F. L. Gervasio, Rep. Prog. Phys. 2008, 71, 126601.
[3] D. Castelvecchi, Sci. News 2007, 171, 372.
[4] D. Adams, The Hitch Hiker’s Guide to the Galaxy: A Trilogie in Five Parts, Hitchhiker’s Guide to the Galaxy Series, Penguin Random House, 1995.
[5] C. Dellago, P. G. Bolhuis, F. S. Csajka, D. Chandler, J. Chem. Phys. 1998, 108, 1964.
[6] P. G. Bolhuis, C. Dellago, D. Chandler, Faraday Discuss. 1998, 110, 421.
[7] C. Dellago, P. G. Bolhuis, D. Chandler, J. Chem. Phys. 1999, 110, 6617.
[8] L. Verlet, Phys. Rev. 1967, 159, 98.
[9] O. Buneman, J. Comput. Phys. 1967, 1, 517.
[10] B. J. Leimkuhler, S. Reich, R. D. Skeel, Integration Methods for Molecular Dynamics. In Mathematical Approaches to Biomolecular Structure and Dynamics, New York: Springer, 1996, p. 161.
[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, J. Chem. Phys. 1953, 21, 1087.
[12] W. K. Hastings, Biometrika 1970, 57, 97.
[13] T. S. van Erp, D. Moroni, P. G. Bolhuis, J. Chem. Phys. 2003, 118, 7762.
[14] T. S. van Erp, Phys. Rev. Lett. 2007, 98, 268301.
[15] A. Lervik, E. Riccardi, T. S. van Erp, J. Comput. Chem. 2017, 38, 2439.
[16] D. W. H. Swenson, J.-H. Prinz, F. Noe, J. D. Chodera, P. G. Bolhuis, J. Chem. Theory Comput. 2015, 10, 1513.
[17] D. W. H. Swenson, J.-H. Prinz, F. Noe, J. D. Chodera, P. G. Bolhuis, J. Chem. Theory Comput. 2019, 15, 837.
[18] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, E. Lindahl, SoftwareX 2015, 1–2, 19.
[19] J. Hutter, M. Iannuzzi, F. Schiffmann, J. VandeVondele, WIREs Comput. Mol. Sci. 2014, 4, 15.
[20] 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. Yee, M. Houston, T. Stich, C. Klein, M. R. Shirts, V. S. Pande, J. Chem. Theory Comput. 2013, 9, 461.
[21] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, PLoS Comput. Biol. 2017, 13, 1.
[22] S. Plimpton, J. Comput. Phys. 1995, 110, 1.
[23] E. Riccardi, O. Dahlen, T. S. van Erp, J. Phys. Chem. Lett. 2017, 8, 4456.
[24] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, V. S. Pande, Biophys. J. 2015, 109, 1528.
[25] G. Henkelman, B. P. Uberuaga, H. Jónsson, J. Chem. Phys. 2000, 113, 9901.
[26] E. Riccardi, E. C. van Mastbergen, W. W. Navarre, J. Vreede, PLoS Comput. Biol. 2019, 15, e1006845.
[27] O. Aarøen and E. Riccardi. [Submitted in October 2019 to PeerJ Physical Trajectories].
[28] T. S. van Erp, M. Moqadam, E. Riccardi, A. Lervik, J. Chem. Theory Comput. 2016, 12, 5398.
[29] 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.
[30] E. Riccardi, S. Pantano, R. Potestio, Interface Focus 2019, 9, 20190005.
[31] M. Moqadam, E. Riccardi, T. T. Trinh, A. Lervik, T. S. van Erp, Phys. Chem. Chem. Phys. 2017, 19, 13361.
[32] N. Michaud-Agrawal, E. J. Denning, T. B. Wooff, and O. Beckstein. MD Analysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. J. Comput. Chem. 2011, 32, 2319–2327. https://doi.org/10.1002/jcc.21787.
[33] R. J. Gowers, M. Linke, J. Barnoud, T. J. Reddy, M. N. Melo, S. L. Seyler, J. Domański, D. L. Dotson, S. Buchoux, I. M. Kenney, O. Beckstein, MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In Proceedings of the 15th Python in Science Conference, Vol. 98, SciPy, Austin, TX, 2016.
[34] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dukak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiitz, O. Schütz, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K. W. Jacobsen, J. Phys. Condens. Matter 2017, 29, 273002.

Received: 8 July 2019
Revised: 25 October 2019
Accepted: 29 October 2019
Published online on 19 November 2019