




asyncmd: A python library to orchestrate complex molecular dynamics simulation campaigns on high performance computing systems

Hendrik Jung ¹ and Gerhard Hummer ^{1,2}

¹ Max Planck Institute of Biophysics, Department of Theoretical Biophysics, Frankfurt am Main, Germany ² Institute of Biophysics, Goethe University Frankfurt, Frankfurt am Main, Germany

DOI: [10.21105/joss.08321](https://doi.org/10.21105/joss.08321)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Evan Spotte-Smith](#) 

Reviewers:

- [@braniiii](#)
- [@corettialessandro](#)

Submitted: 06 May 2025

Published: 04 August 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Molecular dynamics (MD) simulations have become an integral tool to study complex molecular rearrangements and molecular phenomena in many fields and of many different molecular compounds. They are especially useful to study the dynamics, interactions, and function of biomolecules such as proteins, DNA, RNA, lipids, and small drug-like molecules. In MD simulations, the time evolution of a molecular system is obtained by solving Newton's equations of motion in small timesteps starting from a given initial configuration. Due to the inherent sequential nature of MD and the high dimensionality and complexity of the studied (bio)molecular systems, it can be challenging to reach the timescale of the (biological) events under investigation within a single MD trajectory. This challenge can be addressed by running many MD simulations simultaneously on a high performance computing (HPC) system - possibly in an adaptive manner and combined with enhanced sampling techniques - and then making full use of the resulting ensemble of trajectories. It is therefore paramount to enable MD users to efficiently setup and orchestrate a large number of simultaneous simulations with dynamic dependencies and flexible termination conditions. `asyncmd` enables users to define complex MD sampling workflows in python from simple building blocks and executes these computations directly via the queuing system of a HPC resource. The submission of smaller computation tasks as single jobs (with differing requirements) to the queuing system ensures optimal usage of the heterogeneous resources of modern HPC systems. In addition, this approach supports dynamically growing and shrinking of the total resources allocated to the computation by design, also depending on the demand of other users of the same HPC system. By providing their common building blocks, `asyncmd` also simplifies the development and implementation of advanced trajectory-based enhanced sampling algorithms, including the weighted ensemble method, the string method, or transition path sampling.

Statement of need

A challenge faced when performing MD simulations of complex (bio)molecular rearrangements is the disparity of timescales between the integration timestep, which needs to be small (usually on the order of femtoseconds) to ensure accurate integration of Newton's equations of motion, and the time needed to observe the (biological) process under investigation, which can be on the order of seconds to minutes. Due to the sequential nature inherent to solving Newton's equations of motion in small timesteps, MD simulations can only be parallelized to a certain degree and much of the parallelization possible relies on dividing the system into regions that are sufficiently far apart to not interact directly with each other. Taken together with the trend that, in the recent past, computing resources have mostly become wider (i.e. more parallel) but not faster (i.e. not increased in clock speed), the result is that the simulated systems have

become larger while the simulated time (per trajectory) did not increase by much anymore. A common strategy to accumulate the required simulation time to observe the process under investigation while also making the most efficient use of modern highly parallel HPC resources, is to run many MD simulations of the same (biological) system with different initial conditions simultaneously. This strategy results in a much higher total accumulated simulated time than running one simulation on the same computing resources due to the inherent sequential nature of MD. While, depending on the number of simultaneous MD simulations, it can be tedious but still possible to manually setup a large number of MD simulations, continually monitoring their progress quickly becomes unfeasible. However, in many cases it is much more computationally efficient to, instead of performing MD for a fixed number of integration steps, terminate the simulations (and potentially use the freed resources for another MD simulation) once a certain condition is met or event occurred, e.g., the transition between two functional states of a biomolecule. Conversely, if the goal is to explore a functional state previously unreachable by MD, it can be much more efficient to (re)start a large number of MD simulations from the first configuration that reaches the state in any of a number of simulations started from known states.

`asyncmd` is a python library facilitating flexible, programmatic and parallel setup, control, and analysis of an arbitrary number of MD simulations on HPC resources using the python `async/await` syntax. The library currently supports the SLURM queuing system (Jette & Wickberg, 2023) and the GROMACS MD engine (Páll et al., 2020), but can easily be extended to other queuing systems and MD engines. To enable the handling of many MD engines simultaneously, all MD engines return a lightweight Trajectory object that only contains references to the underlying files and some useful metadata such as the length of the trajectory or the integration timestep used. All trajectory reading and writing operations performed make use of the MDAnalysis library (Gowers et al., 2016; Michaud-Agrawal et al., 2011), which means that `asyncmd` can profit from the extensive variety of trajectory formats and MD engines supported in MDAnalysis. Notable features of the `asyncmd` library include the propagation of MD until any or all user-supplied conditions are fulfilled on the trajectory, the parallelized application of user defined (python) functions on existing or generated trajectories (including the automatic caching of calculated values), and a dictionary-like interface to the MD parameters. By additionally making it easy to extract any molecular configuration to (re)start an arbitrary number of MD simulations from it, users are enabled to build complex sampling schemes with dynamical dependencies from simple building blocks in python.

All computationally costly operations are submitted via the queuing system, i.e., the process running `asyncmd` has a low computational footprint and can be run on the login node to control complex and long running simulation setups with dynamic dependencies. Additionally, the submission via the existing queuing system ensures that the available HPC resources are efficiently shared with other users independently of them using `asyncmd` or not. For each submittable computation `asyncmd` includes a locally executed equivalent using the same calling convention to facilitate quick prototyping and implementation of (new) algorithms using small test systems on local compute resources. By making it easy to move the execution from a workstation to a HPC cluster, the newly implemented algorithms can then be applied directly to larger, computationally more costly molecular systems.

`asyncmd` can therefore be used to manage large scale MD simulation campaigns on HPC resources. In addition, `asyncmd` provides an ideal building block to develop and implement trajectory based enhanced sampling methods as, e.g., variants of the string method (E et al., 2002), highly parallelized transition path or transition interface sampling methods (Dellago et al., 2002; van Erp et al., 2003), flux sampling methods (Chandler, 1978; Ruiz-Montero et al., 1997), or the weighted ensemble method (G. A. Huber & Kim, 1996). Showcasing its potential, `asyncmd` was already used to develop and implement the AI for molecular mechanism discovery (`aimmd`) algorithm (Jung, 2022; Jung et al., 2023), which adaptively steers a large number of simultaneous MD simulations using a combination of transition path sampling and machine learning.

In addition to its documentation, `asyncmd` includes a number of examples in the form of jupyter notebooks. These notebooks illustrate most of the common operations, such as performing MD simulations (possibly until a condition is fulfilled) or how to extract configurations from trajectories to restart MD simulations from it, but also include an example showcasing how to implement the weighted ensemble method.

State of the field

A number of other software packages are relevant in the context of submitting MD simulations on HPC resources or to control them from python, which will be discussed in the following.

Notably, it is possible to control and define MD simulation workflows for GROMACS in python by using its `gmxml` python interface (C. A. K. Irrgang M. Eric AND Davis, 2022; M. E. Irrgang et al., 2018). While `gmxml` allows for fine grained control of the MD simulation (including, e.g., custom stopping conditions and user plugin code within the force calculation), it is only possible to interact with MD simulations running within the same job allocation or on the same local machine.

The definition and submission of a general (non MD-specific) computational workflow spanning over multiple job allocations on HPC resources from python is possible by using AiiDA (S. P. Huber et al., 2020; Uhrin et al., 2021) or by using the combination of row (Anderson et al., 2024) and signac (Adorf et al., 2018). Both, AiiDa and signac/row, have an emphasis on automatically storing data provenance, while `asyncmd` makes no attempt to store any input/output relations and it is the users responsibility (and freedom) to choose an adequate solution for their use-case to track data provenance. AiiDA currently supports a number of different queuing systems, but it is not possible to request accelerator resources such as GPUs. row currently only supports the SLURM queuing system, but offers a finer control over the job resources including the requested memory and number of GPUs. `asyncmd` currently also only supports the SLURM queuing system, but offers support for any option of the SLURM “`sbatch`” command to control the requested resources for and execution of the jobs. MD simulations can be performed with AiiDA by using the `aiida-gromacs` plugin (Gebbie-Rayet & Kalayan, 2022) and `martignac` (Bereau et al., 2024) defines a number of coarse-grained Martini simulation workflows with signac. Another notable package enabling high-throughput MD simulations in the context of materials science is `atomate2` (Ganose et al., 2025), which is also capable of submitting the computations to remote (HPC) resources. However, to the best knowledge of the authors, no other package besides `asyncmd` exists that offers the submission and control of many MD simulation via a queuing system, while also focusing on versatile and dynamic stopping conditions for the simulations to provide simple building blocks for enhanced sampling algorithms.

Finally, what sets `asyncmd` apart from full-fledged implementations of path sampling and other trajectory based sampling methods, such as, e.g., `openpathsampling` (Swenson et al., 2019a, 2019b), is that it does not implement any specific algorithms to drive the sampling, but instead strives to only provide the common building blocks shared between many trajectory based enhanced sampling methods.

Acknowledgements

The authors thank all users of `asyncmd` for contributing feedback and suggesting new features, especially Matea Turalija and Vedran Miletic, for feedback on and contributions to the code. H.J. and G.H. thank the Max Planck Society for financial support and the Max Planck Computing and Data Facility (MPCDF) for computing support.

References

- Adorf, C. S., Dodd, P. M., Ramasubramani, V., & Glotzer, S. C. (2018). Simple data and workflow management with the signac framework. *Computational Materials Science*, 146(C), 220–229. <https://doi.org/10.1016/j.commatsci.2018.01.035>
- Anderson, J. A., Bradley, J., Burkhart, J., Jensen, K., Moore, T., Kerr, C., & Teague, T. (2024). Row. In *GitHub repository*. GitHub. <https://github.com/glotzerlab/row>
- Bereau, T., Walter, L. J., & Rudzinski, J. F. (2024). Martignac: Computational workflows for reproducible, traceable, and composable coarse-grained martini simulations. *Journal of Chemical Information and Modeling*, 64(24), 9413–9423. <https://doi.org/10.1021/acs.jcim.4c01754>
- Chandler, D. (1978). Statistical mechanics of isomerization dynamics in liquids and the transition state approximation. *The Journal of Chemical Physics*, 68(6), 2959–2970. <https://doi.org/10.1063/1.436049>
- Dellago, C., Bolhuis, P. G., & Geissler, P. L. (2002). Transition Path Sampling. In *Advances in Chemical Physics* (Vol. 123, pp. 1–78). John Wiley & Sons, Ltd. <https://doi.org/10.1002/0471231509.ch1>
- E, W., Ren, W., & Vanden-Eijnden, E. (2002). String method for the study of rare events. *Physical Review B*, 66(5), 052301. <https://doi.org/10.1103/PhysRevB.66.052301>
- Ganose, A. M., Sahasrabudde, H., Asta, M., Beck, K., Biswas, T., Bonkowski, A., Bustamante, J., Chen, X., Chiang, Y., Chrzan, D. C., Clary, J., Cohen, O. A., Ertural, C., Gallant, M. C., George, J., Gerits, S., Goodall, R. E. A., Guha, R. D., Hautier, G., ... Jain, A. (2025). Atomate2: Modular workflows for materials science. *Digital Discovery*. <https://doi.org/10.1039/D5DD00019J>
- Gebbie-Rayet, J., & Kalayan, J. (2022). Aiida-gromacs. In *GitHub repository*. GitHub. <https://github.com/PSDI-UK/aiida-gromacs>
- Gowers, Richard J., Linke, Max, Barnoud, Jonathan, Reddy, Tyler J. E., Melo, Manuel N., Seyler, Sean L., Domański, Jan, Dotson, David L., Buchoux, Sébastien, Kenney, Ian M., & Beckstein, Oliver. (2016). MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In Sebastian Benthall & Scott Rostrup (Eds.), *Proceedings of the 15th Python in Science Conference* (pp. 98–105). <https://doi.org/10.25080/Majora-629e541a-00e>
- Huber, G. A., & Kim, S. (1996). Weighted-ensemble Brownian dynamics simulations for protein association reactions. *Biophysical Journal*, 70(1), 97–110. [https://doi.org/10.1016/S0006-3495\(96\)79552-8](https://doi.org/10.1016/S0006-3495(96)79552-8)
- Huber, S. P., Zoupanos, S., Uhrin, M., Talirz, L., Kahle, L., Häuselmann, R., Gresch, D., Müller, T., Yakutovich, A. V., Andersen, C. W., Ramirez, F. F., Adorf, C. S., Gargiulo, F., Kumbhar, S., Passaro, E., Johnston, C., Merkys, A., Cepellotti, A., Mounet, N., ... Pizzi, G. (2020). AiIDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Scientific Data*, 7(1), 300. <https://doi.org/10.1038/s41597-020-00638-4>
- Irrgang, C. A. K., M. Eric AND Davis. (2022). Gmxapi: A GROMACS-native python interface for molecular dynamics with ensemble and plugin support. *PLOS Computational Biology*, 18(2), 1–12. <https://doi.org/10.1371/journal.pcbi.1009835>
- Irrgang, M. E., Hays, J. M., & Kasson, P. M. (2018). Gmxapi: A high-level interface for advanced control and extension of molecular dynamics simulations. *Bioinformatics*, 34(22), 3945–3947. <https://doi.org/10.1093/bioinformatics/bty484>
- Jette, M. A., & Wickberg, T. (2023). Architecture of the slurm workload manager. In D. Klusáček, J. Corbalán, & G. P. Rodrigo (Eds.), *Job scheduling strategies for par-*

- allele processing* (pp. 3–23). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43943-8_1
- Jung, H. (2022). Aimmd: AI for molecular mechanism discovery. In *GitHub repository*. GitHub. <https://github.com/bio-phys/aimmd>
- Jung, H., Covino, R., Arjun, A., Leitold, C., Dellago, C., Bolhuis, P. G., & Hummer, G. (2023). Machine-guided path sampling to discover mechanisms of molecular self-organization. *Nature Computational Science*, 3(4), 334–345. <https://doi.org/10.1038/s43588-023-00428-z>
- Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*, 32(10), 2319–2327. <https://doi.org/10.1002/jcc.21787>
- Páll, S., Zhmurov, A., Bauer, P., Abraham, M., Lundborg, M., Gray, A., Hess, B., & Lindahl, E. (2020). Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. *The Journal of Chemical Physics*, 153(13), 134110. <https://doi.org/10.1063/5.0018516>
- Ruiz-Montero, M. J., Frenkel, D., & Brey, J. J. (1997). Efficient schemes to compute diffusive barrier crossing rates. *Molecular Physics*, 90(6), 925–942. <https://doi.org/10.1080/002689797171922>
- Swenson, D. W. H., Prinz, J.-H., Noe, F., Chodera, J. D., & Bolhuis, P. G. (2019a). OpenPath-Sampling: A Python framework for path sampling simulations. 1. Basics. *Journal of Chemical Theory and Computation*, 15(2), 813–836. <https://doi.org/10.1021/acs.jctc.8b00626>
- Swenson, D. W. H., Prinz, J.-H., Noe, F., Chodera, J. D., & Bolhuis, P. G. (2019b). OpenPath-Sampling: A Python framework for path sampling simulations. 2. Building and customizing path ensembles and sample schemes. *Journal of Chemical Theory and Computation*, 15(2), 837–856. <https://doi.org/10.1021/acs.jctc.8b00627>
- Uhrin, M., Huber, S. P., Yu, J., Marzari, N., & Pizzi, G. (2021). Workflows in AiiDA: Engineering a high-throughput, event-based engine for robust and modular computational workflows. *Computational Materials Science*, 187, 110086. <https://doi.org/10.1016/j.commatsci.2020.110086>
- van Erp, T. S., Moroni, D., & Bolhuis, P. G. (2003). A novel path sampling method for the calculation of rate constants. *The Journal of Chemical Physics*, 118(17), 7762–7774. <https://doi.org/10.1063/1.1562614>