




# FlowerMD: Flexible Library of Organic Workflows and Extensible Recipes for Molecular Dynamics

Marjan Albooyeh <sup>1\*</sup>, Chris Jones <sup>1\*</sup>, Rainier Barrett <sup>1</sup>, and Eric Jankowski <sup>1¶</sup>

<sup>1</sup> Boise State University, Boise, ID, United States of America ¶ Corresponding author \* These authors contributed equally.

DOI: [10.21105/joss.05989](https://doi.org/10.21105/joss.05989)

## Software

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

Editor: [Lucy Whalley](#) 

## Reviewers:

- [@csadorf](#)
- [@LIVazquezS](#)
- [@abhishektiware](#)

Submitted: 18 October 2023

Published: 06 December 2023

## License

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

## Summary

flowerMD is a package for reproducibly performing multi-stage HOOMD-blue ([Anderson et al., 2020](#)) simulation workflows. It enables the programmatic specification of tasks including definition of molecular structures, forcefield definition and application and chaining together simulation stages (e.g., shrinking, equilibration, simulating a sequence of ensembles, tensile testing, etc.) through an extensible set of Python classes. The modular design supports a library of workflows for organic macromolecular and polymer simulations. Tutorials are provided to demonstrate package features and flexibility.

## Statement of need

High-level programmatic specifications of molecular simulation workflows are needed for two reasons. First, they provide the information necessary for a simulation study to be reproduced, and second, they help lower the cognitive load associated with learning and performing simulations in general. Reproducible simulations benefit the research community by enabling studies to be validated and extended. Lowering the cognitive load of performing molecular simulations helps computational researchers of all levels of expertise reason about the logic of a simulation study. This is particularly important for researchers new to the discipline because developing the tools needed to perform experiments often involves: (a) gaining new software development skills and knowledge, and (b) repeating work that others have already performed.

Recent advances in open-source tools have made the programmatic specification of molecular simulation components easier than ever ([Anderson et al., 2020](#); [Eastman et al., 2017](#); [Grünwald et al., 2022](#); [Hedges et al., 2019](#); [Klein et al., 2016](#); [Martin et al., 2018](#); [Santana-Bonilla et al., 2023](#); [Swails et al., 2014](#); [A. P. Thompson et al., 2022](#); [M. Thompson et al., 2023](#)). Individually, each of these tools lower the cognitive load of one aspect of an overall workflow such as representing molecules, building initial structures, parameterizing and applying a forcefield, and running simulations. However, stitching these pieces together to create a complete workflow presents a need that we address in the present work.

The computational researcher who follows best practices for accurate, accessible and reproducible results may create a programmatic layer over these individual software packages (i.e. wrapper) that serves to consolidate and automate a complete workflow. However, these efforts often use a bespoke approach where the entire workflow design is tailored toward the specific question or project. Design choices might include the materials studied, the model used (e.g. atomistic or coarse-grained), the source of the forcefield in the model, and the simulation protocols followed. As a result, this wrapper is likely unusable for the next project where one of the aforementioned choices changes, and the process of designing a workflow

must begin again from scratch.

Software packages such as Radonpy ([Hayashi et al., 2022](#)) exist that provide an automated workflow for building molecules and bulk structures to calculating physical properties of polymers. However, these tools may not be suitable for modeling complex experimental processes that extend beyond measuring material properties, such as simulating fusion welding of polymer interfaces ([Aggarwal et al., 2020](#); [Bukowski et al., 2021](#)) and surface wetting ([Bamane et al., 2021](#); [Fan & Çağın, 1995](#)).

flowerMD is a Python package that consolidates and automates end-to-end workflows for modeling such engineering processes with a focus on organic molecules. We expand the capabilities of MoSDeF ([M. W. Thompson et al., 2020](#)) and HOOMD-blue ([Anderson et al., 2020](#)), following TRUE principles of software design (Transparent, Reproducible, Usable by others, and Extensible ([M. W. Thompson et al., 2020](#))) with modular components that facilitate building and running workflows for specific materials science research applications, while reducing the cognitive load and programming demands on the user's part.

## Building Blocks

flowerMD is extensible. Modular base classes serve as building blocks that lay the foundation for constructing workflow recipes designed for specific applications. The resultant recipes are agnostic to choices such as chemistry, model resolution (e.g. atomistic vs. coarse grained) and forcefield selection. This is accomplished via three base classes:

- The Molecule class utilizes the mBuild ([Klein et al., 2016](#)) and GMSO ([M. Thompson et al., 2023](#)) packages to initialize chemical structures from a variety of input formats. This class provides methods for building polymers and copolymer structures, and supports a straightforward coarse-graining process by leveraging SMARTS matching.
- The System class serves as an intermediary between molecular initialization and simulation setup. This class builds the initial configuration and applies a chosen forcefield that defines particle interactions.
- The Simulation class adds a layer on top of the HOOMD-blue simulation object, adding additional methods and features to simplify the process of starting and resuming a HOOMD-blue simulation.

Additionally, flowerMD offers a library pre-defined subclasses of these base classes including common polymers, forcefields, and bulk system initialization algorithms.

## Recipes

The modular design of flowerMD enables version-controlled workflows to be created and shared. flowerMD currently includes two complete workflows: a polymer fusion welding recipe and a surface wetting recipe. As an example, the polymer welding recipe uses the following subclasses: `flowerMD.modules.welding.SlabSimulation`, `flowerMD.modules.welding.Interface`, `flowerMD.modules.welding.WeldSimulation`, and `flowerMD.library.simulations.Tensile`.

Applying these routines in sequence defines a polymer welding recipe:

1. `SlabSimulation` creates one "slab" with two flat surfaces of e.g. polyethylene.
2. `Interface` duplicates the resultant slab and creates an "interface" system.
3. `WeldSimulation` simulates thermal welding at this interface.
4. `Tensile` simulates a tensile test of the resultant weld to create a stress/strain curve.

In this example, three different kinds of molecular dynamics simulations are combined in sequence to enable the mechanical properties of a welded joint to be computed. By combining

independent simulation steps and enabling any of them to be iterated over, a user can build more complex workflows. For example, we could specify a screening study by iterating over several weld temperatures, following each weld simulation with a sequence of tensile simulations to investigate how temperature and strain rate influence the debonding pressure of a polymer weld. For an in-depth understanding of the surface wetting workflow, interested readers can refer to the flowerMD tutorials.

Each of the steps in a recipe takes the molecular system and forcefield as arguments, enabling recipes that iterate over these concepts as well. For example, a user can create a recipe that generates welds of a set of polymer chemistries. Or, a user can create a recipe that measures the agreements in structural predictions for a set of forcefields.

flowerMD's recipe focus allows researchers to concisely and programmatically specify their desired simulation logic. By offloading the cognitive load of workflow specification, researchers can more easily ensure this logic is correct before embarking on computationally expensive studies. We encourage simulators of all levels of expertise to file issues requesting new features and to submit pull requests to extend flowerMD's utility.

## Availability

flowerMD is freely available under the GNU General Public License (version 3) on [GitHub](#). For installation instructions, and Python API documentation please visit the [documentation](#). For examples of how to use flowerMD, please visit the [tutorials](#).

## Acknowledgements

This research was partially supported by the National Aeronautics and Space Administration (NASA) under the University Leadership Initiative program; grant number 80NSSC20M0165. This material is based upon work supported by the National Science Foundation under Grant Numbers: 1653954, 1835593, and 2118217. No sponsor had any involvement in the development of flowerMD.

## Conflict of Interest Statement

The authors declare the absence of any conflicts of interest: No author has any financial, personal, professional, or other relationship that affect our objectivity toward this work.

## References

- Aggarwal, I., Paul, S., Sinha, N. K., & Basu, S. (2020). Molecular dynamics studies on the strength and ductility of symmetric thermally welded joints. *Modelling and Simulation in Materials Science and Engineering*. <https://doi.org/10.1088/1361-651X/ab6a44>
- Anderson, J. A., Glaser, J., & Glotzer, S. C. (2020). HOOMD-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations. *Computational Materials Science*, 173, 109363. <https://doi.org/10.1016/j.commatsci.2019.109363>
- Bamane, S. S., Gaikwad, P. S., Radue, M. S., Gowtham, S., & Odegard, G. M. (2021). Wetting simulations of high-performance polymer resins on carbon surfaces as a function of temperature using molecular dynamics. *Polymers*, 13(13), 2162. <https://doi.org/10.3390/polym13132162>
- Bukowski, C., Zhang, T., Riggleman, R. A., & Crosby, A. J. (2021). Load-bearing entanglements in polymer glasses. *Science Advances*, 7(38), eabg9763. <https://doi.org/10.1126/sciadv.abg9763>

- Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simonett, A. C., Harrigan, M. P., Stern, C. D., & others. (2017). OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, 13(7), e1005659.
- Fan, C. F., & Çağın, T. (1995). Wetting of crystalline polymer surfaces: A molecular dynamics simulation. *The Journal of Chemical Physics*, 103(20), 9053–9061. <https://doi.org/10.1063/1.470016>
- Grünwald, F., Alessandri, R., Kroon, P. C., Monticelli, L., Souza, P. C. T., & Marrink, S. J. (2022). Polyply; a python suite for facilitating simulations of macromolecules and nanomaterials. *Nature Communications*, 13(1), 68. <https://doi.org/10.1038/s41467-021-27627-4>
- Hayashi, Y., Shiomi, J., Morikawa, J., & Yoshida, R. (2022). RadonPy: Automated physical property calculation using all-atom classical molecular dynamics simulations for polymer informatics. *Npj Computational Materials*, 8(1), 222. <https://doi.org/10.1038/s41524-022-00906-4>
- Hedges, L., Mey, A., Laughton, C., Gervasio, F., Mulholland, A., Woods, C., & Michel, J. (2019). BioSimSpace: An interoperable python framework for biomolecular simulation. *Journal of Open Source Software*, 4(43), 1831. <https://doi.org/10.21105/joss.01831>
- Klein, C., Sallai, J., Jones, T. J., Iacovella, C. R., McCabe, C., & Cummings, P. T. (2016). A hierarchical, component based approach to screening properties of soft matter. In R. Q. Snurr, C. S. Adjiman, & D. A. Kofke (Eds.), *Foundations of molecular modeling and simulation: Select papers from FOMMS 2015* (pp. 79–92). Springer Singapore. [https://doi.org/10.1007/978-981-10-1128-3\\_5](https://doi.org/10.1007/978-981-10-1128-3_5)
- Martin, T. B., Gartner III, T. E., Jones, R. L., Snyder, C. R., & Jayaraman, A. (2018). pyPRISM: A computational tool for liquid-state theory calculations of macromolecular materials. *Macromolecules*, 51(8), 2906–2922. <https://doi.org/10.1021/acs.macromol.8b00011>
- Santana-Bonilla, A., López-Ríos De Castro, R., Sun, P., Ziolek, R. M., & Lorenz, C. D. (2023). Modular software for generating and modeling diverse polymer databases. *Journal of Chemical Information and Modeling*, 63(12), 3761–3771. <https://doi.org/10.1021/acs.jcim.3c00081>
- Swails, J., Hernandez, C., Mobley, D. L., Nguyen, H., Wang, L.-P., & Janowski, P. (2014). *ParmED: Cross-program parameter and topology file editor and molecular mechanical simulator engine* (Version 4.1.0). GitHub. <https://github.com/ParmEd/ParmEd>
- Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., In 'T Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., & Plimpton, S. J. (2022). LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271, 108171. <https://doi.org/10.1016/j.cpc.2021.108171>
- Thompson, M. W., Gilmer, J. B., Matsumoto, R. A., Quach, C. D., Shamaprasad, P., Yang, A. H., Iacovella, C. R., McCabe, C., & Cummings, P. T. (2020). Towards molecular simulations that are transparent, reproducible, usable by others, and extensible (TRUE). *Molecular Physics*, 118(9), e1742938. <https://doi.org/10.1080/00268976.2020.1742938>
- Thompson, M., Yang, A., Matsumoto, R., Timalisina, U., Quach, C., CalCraven, Shamaprasad, P., Gilmer, J., DeFever, R. S., Chris, J., Dice, B., Crawford, B., Iacovella, C., Albooyeh, M., Smith, R., Bansal, A., Marin-Rimoldi, E., & zijiewu3. (2023). *GMSO: General Molecular Simulation Object* (Version 0.11.2). <https://doi.org/10.5281/zenodo.8370982>