The Journal of Open Source Software

# dcTMD: a python package for performing dissipation-corrected targeted molecular dynamics

**Miriam Jäger**[1], **Victor Tänzel**[1], **Daniel Nagel**[2], **and Steffen Wolf**[1,3]

**1** University of Freiburg, Institute of Physics, Freiburg, Germany **2** University of Heidelberg, Institute of Theoretical Physics, Heidelberg, Germany **3** Darmstadt University of Applied Science, Faculty of Chemical Engineering and Biotechnology, Darmstadt, Germany

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:**

**Submitted:** 28 November 2025
**Published:** unpublished



## Summary

dcTMD is a Python package designed to extract free-energy and non-equilibrium friction estimates from targeted molecular dynamics (TMD) simulations by Schlitter et al. (1994). The method implemented here is called *dissipation-corrected targeted molecular dynamics* (dcTMD) by Wolf & Stock (2018). Given a set of non-equilibrium simulations where, for example, a ligand is pulled from a protein binding site via a velocity constraint, this tool performs automated post-processing of the bias-force time traces to estimate the underlying free-energy landscape and the friction (dissipation) along the unbinding coordinate.

The method is based on a second-order cumulant expansion of the *Jarzynski equality* (Jarzynski (1997)), which connects non-equilibrium work distributions to equilibrium free-energy differences. Combined with a Markovian Langevin Equation, dcTMD further allows the extraction of *position- and velocity-dependent friction coefficients* from the same non-equilibrium data. This approach has been succesfully applied in multiple studies (Wolf et al. (2020), Jäger et al. (2022), Post et al. (2022), Tänzel et al. (2024), and Jäger & Wolf (2025) Milster et al. (2025)). The resulting free-energy and friction profiles can subsequently be used to estimate *binding and unbinding rate constants* following Wolf et al. (2020).

The software is intended for molecular dynamics practitioners interested in ligand–protein unbinding, mechanistic interpretation of binding kinetics, and quantitative modeling of non-equilibrium effects in soft condensed matter and biomolecular systems.

## Statement of need

Ligand unbinding from proteins is of fundamental interest in computational biophysics (Schuetz et al. (2017)). In many cases, the unbinding event is rare and requires enhanced-sampling or biased-simulation strategies to observe within computationally feasible timescales. dcTMD-based workflows have been shown to yield accurate free energy and non-equilibrium friction coefficients from velocity-constrained pulling simulations. The dcTMD package builds on this work by offering a unified, documented, and extensible implementation that is currently not available, thereby lowering the barrier for applying dcTMD to new biomolecular systems and for reproducing published dcTMD studies. By providing a dedicated Python framework with a `scikit-learn`-style API, `dcTMD` enables users to integrate dissipation-corrected analysis into existing workflows, ensuring reproducibility and broad accessibility.

The Journal of Open Source Software

## State of the field

Enhanced sampling methods are readily integrated in simulation software packages, e.g., umbrella sampling, or steered MD in GROMACS (Abraham et al. (2015)). Furthermore, dedicated frameworks such as PLUMED (The PLUMED consortium (2019)), PySAGES (Zubieta Rico et al. (2024)), and Colvars (Fiorin et al. (2013)) provide infrastructure for collective variable definition, enhanced sampling and biasing during simulation across multiple molecular dynamics engines.

The present software targets the analysis of an ensemble of constant-velocity pulling trajectories required for dcTMD. This approach relies on constraint pulling simulations created with GROMACS in which the reaction coordinate is enforced via holonomic constraints. The corresponding constraint forces are recorded for subsequent analysis. PLUMED and Colvars employ steered molecular dynamics through moving harmonic restraints rather than constraint-based TMD pulling, while PySAGES does not provide support for GROMACS constraint pulling workflows.

To address this gap, we developed a standalone analysis package that handles ensembles of GROMACS constraint pulling simulations and provides a dedicated workflow for dcTMD post-processing.

## Software design

dcTMD is implemented in Python, since it allows rapid development and community contributions. Potential performance trade-offs associated with an interpreted language are mitigated through the usage of optimized numerical backend libraries such as NumPy and SciPy.

The package inherits the core base classes `BaseEstimator` and `TransformerMixin` from *scikit-learn* (Pedregosa et al. (2011)). Thus, the design follows the familiar `fit/transform` conventions and ensures interoperability with scikit-learn's ecosystem. This simplifies integration into analysis pipelines and promotes a familiar workflow for users. The software architecture has a modular design with five top-level submodules: `dcTMD`, `storing`, `io`, `utils`, and `featureset`.

Because trajectory parsing constitutes the dominant runtime cost, we separate data ingestion from model evaluation. Saving of trajectory data into numpy arrays and the evaluation routines are separated into two core submodules: `storing` and `dcTMD`. The `storing` submodule defines the dataset objects `WorkSet` and `ForceSet`. These are data handlers that standardize the trajectory files into aligned NumPy arrays and allow the processed datasets to be saved to disk and reloaded without reprocessing the original trajectory files. The estimators reside in the `dcTMD` submodule and operate directly on these dataset objects. This separation reduces redundant I/O during iterative analyses.

The `io` and `utils` modules isolate file handling, bootstrapping, smoothing, and plotting utilities. Finally, the `featureset` enables pathway-level analysis by loading trajectory-specific feature files into a consistent array representation, enabling downstream similarity/clustering workflows.

## Research impact statement

As mentioned above, dcTMD has been already shown to efficiently coarse-grain system dynamics for such diverse systems as protein-ligand complexes (Wolf et al. (2020)), ion channels (Jäger et al. (2022)) or to gain insight into shear-dependent viscosities in solvents and lubricants (Post et al. (2022), Milster et al. (2025)). The method thus has proven its capabilities to the scientific community. Its distribution so far was hampered by the necessity to manually implement dcTMD in the form of self-written scripts. The presented package represents a comprehensive and easy-to-use tool to allow other groups who are interested

in applying dcTMD to their problems, that has been optimized for fast and efficient data evaluation with minimal runtime and memory requirements. We allow not only for dissipation correction or friction coefficient calculation via a work-based analysis, but to gain insight into bath degrees of freedom via a force-correlation analysis. Furthermore, the standardised outputs allow for data evaluation quality control across research groups and allow for the establishment of best practices in the usage of dcTMD. The software has already been successfully applied in the following studies: Tänzel et al. (2024), Lalis et al. (2025), Jäger & Wolf (2025), and Milster et al. (2025).

## Implementation

The code is written in Python (versions 3.9–3.14) and is available under the MIT license.
**Repository:** https://github.com/moldyn/dcTMD

Key features include:

- Input support for *GROMACS* pulling trajectories.
- Core functionality for computing free energy and non-equilibrium friction profiles along the biasing coordinate.
- Support for analysis of multiple unbinding pathways.
- Force correlation analysis for non-equilibrium friction analysis.
- Continuous integration and testing via GitHub Actions; documentation hosted at https://moldyn.github.io/dcTMD.

## Use case

A typical workflow begins with the user performing at least 100 independent velocity-constraint pulling simulations. `dcTMD` provides two analysis routes, both following the same workflow pattern:

1. Work-based analysis using a `WorkSet` and `WorkEstimator`
2. Force-correlation analysis using a `ForceSet` and `ForceEstimator`

Both methods yield free-energy and friction profiles but differ in how these properties are estimated.

1. Load trajectories into a `WorkSet` or `ForceSet` The user loads all pulling trajectories into an appropriate container:

- `WorkSet` for the work-based route, which is computationally cheaper, as the resolution of the trajectories can be reduced after integration.
- `ForceSet` for the force-correlation route.

Each trajectory contains the constraint force $f(t)$ from which the work along the pulling coordinate is computed as $W(x) = \int_{x_0}^{x} \mathrm{d}x' f(x')$.

2. Perform dcTMD analysis via an estimator

- **Work-based estimator (`WorkEstimator`)** The free-energy profile is estimated as $\Delta G(x) = \langle W(x) \rangle - \frac{\beta}{2}\langle \delta W(x)^2 \rangle$, with $\delta W = W - \langle W \rangle$, $\beta = (k_\mathrm{B}T)^{-1}$, and $\langle . \rangle$ denoting a trajectory ensemble mean. The dissipated work is $W_\mathrm{diss}(x) = \frac{\beta}{2}\langle \delta W(x)^2 \rangle$. The non-equilibrium position-dependent friction is obtained from its derivative as $\Gamma(x) = \frac{1}{v}\frac{\mathrm{d}}{\mathrm{d}x} W_\mathrm{diss}(x)$.
- **Force-correlation-based estimator (`ForceEstimator`)** In this approach, $\Delta G$ and $\Gamma$ are computed directy from the force data as $\Delta G(x) = \int_{x_0}^{x} \mathrm{d}x' \langle f(x') \rangle - v \int_{x_0}^{x} \mathrm{d}x' \Gamma(x')$

and $\Gamma(x) = \beta \int_0^{t(x)} \mathrm{d}\tau \langle \delta f(t(x)) \delta f(\tau) \rangle$. The two-time force autocorrelation function $C_t(\tau) = \langle \delta f(t(x)) \delta f(\tau) \rangle$ can be plotted to gain insight into timescales within degrees of freedom orthogonal to $x$.

3. Visualize dcTMD provides plotting tools for work distribution analysis, free-energy $\Delta G(x)$ and friction profiles $\Gamma(x)$.

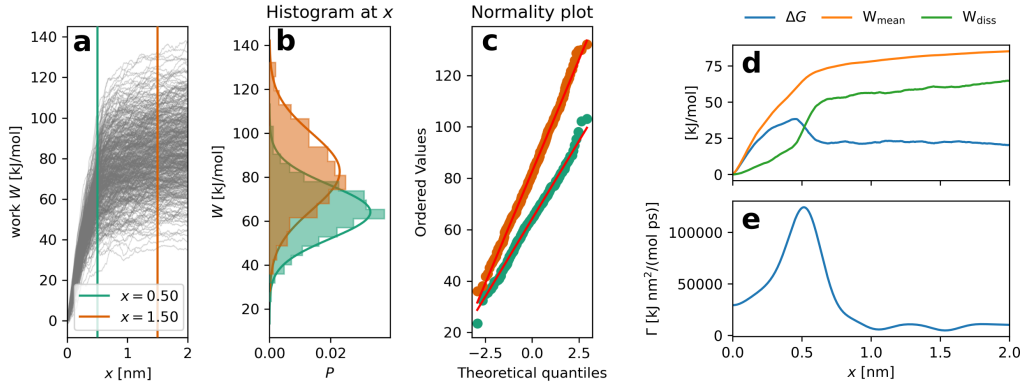


**Figure 1:** Figures crated using data taken from Wolf et al. (2020) of trypsin-benzamidine unbinding. a)-c) work distribution analysis. d) Decomposition of mean work $W_{\mathrm{mean}} = \langle W(x) \rangle$ into free energy $\Delta G(x)$ and dissipation work $W_{\mathrm{diss}}(x)$. e) non-equilibrium friction coefficient $\Gamma(x)$ along the pulling coordinate $x$.

4. Example Figure 1 displays a common analysis of a set of unbinding trajectories from TMD simulations of the trypsin-benzamidine complex (Wolf et al., 2020). The analysis of the work distribution displays good agreement with a normal distribution at two different evaluated positions of the pulling coordinate $x$. The mean work $W_{\mathrm{mean}} = \langle W(x) \rangle$, which shows no features on its own, yields a free energy profile $\Delta G(x)$, which displays a clearly defined transition state at $x \approx 0.45$ nm as well as a bound state in form of a free energy minimum at $x \approx 0.0$ nm and an unbound continuum for $x > 0.6$ nm. The maximum in friction $\Gamma$ around $x = 0.5$ nm is indicative of changes in the hydration of both ligand and binding site.

## AI usage disclosure

Generative AI (ChatGPT with models GPT-4, GPT-4o and GPT-5) was used in a limited capacity during development. It assisted by drafting portions of the documentation, drafting code for selected tests, utility functions, and parts of the `featureset` module, and preparing an initial draft of the manuscript text. All scientific concepts, core design decisions, and implementation of the core modules (`dcTMD`, `io`, and `storing`) were developed without generative AI. The authors guarantee that they reviewed, edited, and validated all AI-assisted outputs.

## Acknowledgements

The implementation of dcTMD builds on the Python scientific stack, relying on **NumPy** (Harris et al. (2020)) for numerical operations, **Matplotlib** (Hunter (2007)) for visualization, and **Click** (The Pallets Projects (2023)) for the command-line interface. We thank Gerhard Stock, Matthias Post and Georg Diez for valuable discussions, and Fabian Rohrbach and Leo Küchler

for testing the software. This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) via grants WO2410/2-1 and WO2410/2-2 within the framework of the Research Unit FOR 5099 *"Reducing complexity of nonequilibrium systems"* (project No. 431945604). The authors acknowledge support by the High Performance and Cloud Computing Group at the Zentrum für Datenverarbeitung of the University of Tübingen and the Rechenzentrum of the University of Freiburg, as well as the state of Baden-Württemberg through bwHPC and the DFG through Grants Nos. INST 37/935-1 FUGG and INST 39/963-1 FUGG.

## References

Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, *1–2*, 19–25. https://doi.org/10.1016/j.softx.2015.06.001

Fiorin, G., Klein, M. L., & Hénin, J. (2013). Using collective variables to drive molecular dynamics simulations. *Molecular Physics*, *111*(22-23), 3345–3362.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55

Jäger, M., Koslowski, T., & Wolf, S. (2022). Predicting ion channel conductance via dissipation-corrected targeted molecular dynamics and langevin equation simulations. *J. Chem. Theory Comput.*, *18*(1), 494–502. https://doi.org/10.1021/acs.jctc.1c00426

Jäger, M., & Wolf, S. (2025). More sophisticated is not always better: A comparison of similarity measures for unsupervised learning of pathways in biomolecular simulations. *J. Phys. Chem. B*, *129*(42), 10956–10966. https://doi.org/10.1021/acs.jpcb.5c04586

Jarzynski, C. (1997). Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, *78*(14), 2690–2693. https://doi.org/10.1103/PhysRevLett.78.2690

Lalis, M., Moitrier, L., Jäger, M., Meinert, C., Brulé, M., Belloir, C., Jones, N. C., Hoffmann, S. V., Fiorucci, S., Wolf, S., Briand, L., & Topin, J. (2025). How allosteric mutations control ligand binding in lipocalin protein: Odorant binding protein as a test case. *Cellular and Molecular Life Sciences*, *82*(1), 250. https://doi.org/10.1007/s00018-025-05777-8

Milster, S., Dzubiella, J., Stock, G., & Wolf, S. (2025). Nonequilibrium friction and free energy estimates for kinetic coarse-graining—driven particles in responsive media. *J. Chem. Phys.*, *162*(15), 154113. https://doi.org/10.1063/5.0261459

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://doi.org/10.5555/1953048.2078195

Post, M., Wolf, S., & Stock, G. (2022). Molecular origin of driving-dependent friction in fluids. *J. Chem. Theory Comput.*, *18*(5), 2816–2825. https://doi.org/10.1021/acs.jctc.2c00190

Schlitter, J., Engels, M., & Krüger, P. (1994). Targeted molecular dynamics: A new approach for searching pathways of conformational transitions. *J. Mol. Graph.*, *12*(2), 84–89. https://doi.org/10.1016/0263-7855(94)80072-3

Schuetz, D. A., Witte, W. E. A. de, Wong, Y. C., Knasmueller, B., Richter, L., Kokh, D. B., Sadiq, S. K., Bosma, R., Nederpelt, I., Heitman, L. H., Segala, E., Amaral, M., Guo, D., Andres, D., Georgi, V., Stoddart, L. A., Hill, S., Cooke, R. M., Graaf, C. de, … Ecker, G. F. (2017). Kinetics for drug discovery: An industry-driven effort to target drug residence time. *Drug Discov. Today*, *22*(6), 896–911. https://doi.org/10.1016/j.drudis.2017.02.002

Tänzel, V., Jäger, M., & Wolf, S. (2024). Learning protein–ligand unbinding pathways via single-parameter community detection. *J. Chem. Theory Comput.*, *20*(12), 5058–5067. https://doi.org/10.1021/acs.jctc.4c00250

The Pallets Projects. (2023). *Click: Python composable command line interface toolkit* (Version 8.1.*). https://palletsprojects.com/p/click

The PLUMED consortium. (2019). Promoting transparency and reproducibility in enhanced molecular simulations. *Nature Methods*, *16*(8), 670–673. https://doi.org/10.1038/s41592-019-0506-8

Wolf, S., Lickert, B., Bray, S., & Stock, G. (2020). Multisecond ligand dissociation dynamics from atomistic simulations. *Nat. Commun.*, *11*(1), 2918. https://doi.org/10.1038/s41467-020-16655-1

Wolf, S., & Stock, G. (2018). Targeted molecular dynamics calculations of free energy profiles using a nonequilibrium friction correction. *J. Chem. Theory Comput.*, *14*(12), 6175–6182. https://doi.org/10.1021/acs.jctc.8b00835

Zubieta Rico, P. F., Schneider, L., Pérez-Lemus, G. R., Alessandri, R., Dasetty, S., Nguyen, T. D., Menéndez, C. A., Wu, Y., Jin, Y., Xu, Y., & others. (2024). PySAGES: Flexible, advanced sampling methods accelerated with GPUs. *Npj Computational Materials*, *10*(1), 35.