The Journal of Open Source Software

MD-SAPT: Python Based Toolkit for Running Symmetry Adapted Perturbation Theory Calculations on Molecular Dynamics Trajectories

Alia E. Lescoulie[1], **Astrid M. Yu**[2], **and Ashley Ringer McDonald**[1¶]

1 Department of Chemistry and Biochemistry, College of Science and Mathematics, California Polytechnic State University, San Luis Obispo, CA, 93407 **2** Department of Computer Science and Software Engineering, College of Engineering, California Polytechnic State University, San Luis Obispo, CA, 93407 ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @wiederm
- @CarvFS

Submitted: 08 February 2023
Published: unpublished

Summary

Molecular dynamics (MD) simulations can identify important interactions in biomolecular systems and examine how these interactions change over the course of the simulation. Biomolecular interactions modeled with MD can be also analyzed in more detail using quantum chemical methods. To accomplish this, we developed MD-SAPT, an open-source Python package, to perform quantum calculations as a post-processing analysis of MD data to quantify and decompose non-covalent interaction energy. MD-SAPT has two modes of analysis: trajectory, which analyzes selected interaction pairs over the frames of an MD simulation, and docking, to examine the binding interactions between a protein and different ligands or compare different docking poses for one particular ligand to determine the relative magnitude of the interactions.

Statement of Need

Non-covalent interactions govern many important biochemical processes, and are most accurately modeled by quantum chemical methods (Sherrill, 2009; Wilson & Peterson, 2007). Quantum methods can also offer more detailed information about an interaction; for example, Symmetry Adapted Perturbation Theory (SAPT), decomposes the interaction energy of the fragments into its component interaction types: electrostatic, dispersion, induction and exchange (Patkowski, 2020).

The molecular mechanics methods used in MD simulations do not characterize noncovalent interactions at the same level of detail as quantum mechanical methods, but allow simulations of large biomolecular systems (Sherrill, 2009). In this work, we present a workflow tool to utilize ab initio methods as a post-processing technique on MD data, offering detailed insight into the strength and type of noncovalent interactions interactions. MD-SAPT automates this process in a reproducible way. It is released as an open source Python library, and distributed using Anaconda on the psi4 conda channel. It supports macOS and Linux, and can be used as a CLI tool, in an interactive Jupyter Notebook, or as an importable Python library.

MD-SAPT automates the process of applying SAPT calculations to structures generated from molecular dynamics simulations. It combines multiple workflow tasks, including selecting the specific residues for the QM calculation, adding protons where the residues were bound to the polypeptide chain, determining the formal charge, passing the charge and coordinates into psi4 (the QM calculation program), and saving the results.

Currently, a number of python based analysis tools exist for MD data. Some popular packages include MDtraj, pyLOOS, and MDAnalysis (need citations here? (Gowers et al., 2016;

McGibbon et al., 2015; Michaud-Agrawal et al., 2011; Romo et al., 2014)). MDAnalysis is a core dependency of MD-SAPT and utilizing existing tools as the starting point for an analysis allows for the development of more complex user-specific methodologies needs without requiring the reimplementation of the basic file processing needed to access MD data.

Specifically, we designed MD-SAPT as an MDA-kit, a plugin in the MDAnalysis ecosystem, by extending its existing analysis interface (Alibay et al., 2022). This is similar to the existing pyLOOS ecosystem, where smaller packages are built using the existing interface for analysis, while the library processes the MD data into a standardized object (Romo et al., 2014). Furthermore, this model has the additional advantage that users are only required to install the tools as well as their associated dependencies specific to their work.

Demonstration

MD-SAPT has two analysis modes. `TrajectorySAPT` calculate the SAPT interaction energy between two (is it just two or could it be two groups?) specified residues for snapshots selected from an MD trajectory. `DockingSAPT` analyzes interactions between similar topologies, such as different molecules in the same active site or a molecule in various poses in an active site. The workflow for each mode is very similar, but each mode uses different classes for their analyses, and have slight differences in the required user-specified parameters.

In MD-SAPT user-specified parameters are provided in a yaml input file. A template to generate blank input files is provided and can be generated using the command `MDSAPT generate [filename]`. This provides users with a preformatted file, such that they only need to fill in their parameters without any additional formatting. The input files for docking and trajectory are slightly different, since the docking mode reqiures several topologies and no trajectory, and the trajectory mode requires a topology and one or more trajectories. An example of each is given in figure Figure 1.

```yaml
psi4:
  method: 'sapt'
  basis: 'jun-cc-pvdz'
  save_output: true
  settings:
    reference: 'rhf'
simulation:
  ph: 7.0
  charge_guesser: 'standard'
system_limits:
  ncpus:
  memory:
analysis:
  type: 'trajectory'
  topology: 'mekfiles/1IR3.psf'
  trajectories:
    - 'mekfiles/step5_5.nc'
  pairs:
    - [369, 144]
    - [34, 146]
  frames:
    start: 0
    stop: 125
    step: 1
  output: 'output.csv'
```

```yaml
psi4:
  method: 'sapt0'
  basis: 'jun-cc-pvdz'
  save_output: true
  settings:
    reference: 'rhf'
simulation:
  ph: 7.0
  charge_guesser: 'standard'
system_limits:
  ncpus: 16
  memory: '64GB'
analysis:
  type: 'docking'
  combined_topologies: 'home/alia/mek_docking'
  pairs:
    - [11, 119]
  output: 'dock_out.csv'
```

Figure 1: Example input files for docking (right) and trajectory (left).

With an input file, running the analysis only requires a few lines of code.

```python
import mdsapt

config = mdsapt.load_from_yaml_file('runinput.yaml')
sapt_run = mdsapt.TrajectorySAPT(config)
sapt_run.run(config.start, config.stop, config.step)
sapt_run.results.to_csv('results.csv')
```

The line beginning with `config = mdsapt.load_from_yaml_file('runinput.yaml')` creates a `Config` object. The `Config` object holds the information needed for an analysis, and when initialized, runs several tests. These test ensure that any errors in the input file, like incorrect file paths or large out-of-bounds parameters are caught early, and highlighted to the user. When loading a `Config` from an input file, it is impossible to get an invalid `Config`, eliminating errors for emerging later in the program. The next line `sapt_run = mdsapt.TrajectorySAPT(config)` initializes the Analysis object, in this case, a `TrajectoryAnalysis`. Encapsulating analyses within a class is a design pattern drawn from MDAnalysis to simplify our code. The next line, `sapt_run.run(config.start, config.stop, config.step)`, runs the analysis frame by frame, creating Psi4 inputs as it iterates through the user's chosen residues and frames. In MDAnalysis user defined analyses, such as the sapt_run method here, are based on the `AnalysisBase` abstract class. `AnalysisBase` contains an abstract methods for preforming the analysis that are called as its `run` method handles iterating over the MD frames. Finally, `sapt_run.results.to_csv('results.csv')` saves the results, which are stored in a `DataFrame`.

The previous example examined the trajectory analysis method. There are a few key differences between it and the docking analysis method. On the user end, docking is preformed using the

class `DockingSAPT` and has slightly differnt input parameters. `DockingSAPT` was implemented using modified code from the `Ensemble` framework developed for MDPOW (Lescoulie, Alia, 2021).

More detailed documentation is available at our readthedocs page.

Using data provided in the MDAnalysis tests we generated the following results from a trajectory analysis over the protein 4AKE. This result was generated as a demonstration of the kinds of data MD-SAPT can generate. The code used to generate this data is available in the binder demo linked on the homepage of our documentation, note that it was for an older version of the software.

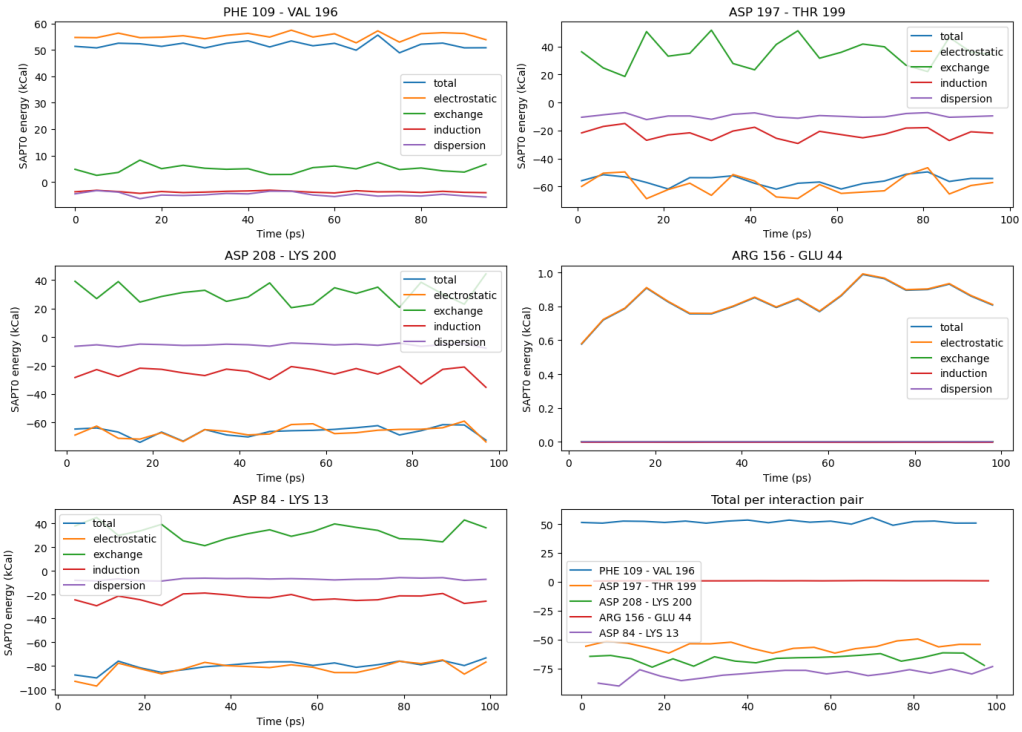

Figure 2: Example MD-SAPT data for various residue-residue interactions in 4AKE

Ongoing Applications in Research

The idea for MD-SAPT's mixed methods approach emerged from our work analyzing active site interactions within MD data. Previously our group preformed MD simulations the protein MEK1, a dual-factor human kinase responsible for cell cycle regulation, and potential targe for chemotherapeutics (Sabsay et al., 2019). Currently, we are applying MD-SAPT to further analyze active site interactions between MEK1 and ATP, working towards a better understanding the activation mechanism.

Acknowledgements

This work was supported by the Bill and Linda Frost Fund at Cal Poly San Luis Obispo. We acknowledge the support of NSF award CHE-2018427. This award provided the computational resources to support this project through the MERCURY Consortium Skylight cluster on Palmetto. # References

Alibay, I., Barnoud, J., Beckstein, O., Gowers, R. J., Naughton, F., & Wang, L. (2022). *MDAKits: Supporting and promoting the development of community packages leveraging the MDAnalysis library*. https://doi.org/10.6084/m9.figshare.20520726.v1

Gowers, R. J., Linke, M., Barnoud, J., Reddy, T. J. E., Melo, M. N., Seyler, S. L., Domański, J., Dotson, D. L., Buchoux, S., Kenney, I. M., & Beckstein, O. (2016). MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In S. Benthall & S. Rostrup (Eds.), *Proceedings of the 15th Python in Science Conference* (pp. 98–105). https://doi.org/10.25080/Majora-629e541a-00e

Lescoulie, Alia. (2021). *Technical Report: SPIDAL Summer REU 2021 Upgrading MDPOW and Adding Analysis Functionality*. 2829450 Bytes. https://doi.org/10.6084/M9.FIGSHARE.17156018.V1

McGibbon, R. T., Beauchamp, K. A., Harrigan, M. P., Klein, C., Swails, J. M., Hernández, C. X., Schwantes, C. R., Wang, L.-P., Lane, T. J., & Pande, V. S. (2015). MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical Journal*, *109*(8), 1528–1532. https://doi.org/10.1016/j.bpj.2015.08.015

Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*, *32*(10), 2319–2327. https://doi.org/10.1002/jcc.21787

Patkowski, K. (2020). Recent developments in symmetry-adapted perturbation theory. *WIREs Computational Molecular Science*, *10*(3). https://doi.org/10.1002/wcms.1452

Romo, T. D., Leioatts, N., & Grossfield, A. (2014). Lightweight object oriented structure analysis: Tools for building tools to analyze molecular dynamics simulations. *Journal of Computational Chemistry*, *35*(32), 2305–2318. https://doi.org/10.1002/jcc.23753

Sabsay, K. R., Lee, R. T., Ravatt, L. M., Oza, J. P., & McDonald, A. R. (2019). Computational Models for Activated Human MEK1: Identification of Key Active Site Residues and Interactions. *Journal of Chemical Information and Modeling*, *59*(5), 2383–2393. https://doi.org/10.1021/acs.jcim.8b00989

Sherrill, C. D. (2009). Computations of Noncovalent π Interactions. In K. B. Lipkowitz & T. R. Cundari (Eds.), *Reviews in Computational Chemistry* (pp. 1–38). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470399545.ch1

Wilson, A. K., & Peterson, K. A. (Eds.). (2007). *Electron Correlation Methodology* (Vol. 958). American Chemical Society. https://doi.org/10.1021/bk-2007-0958