The Journal of Open Source Software

msibi: Multistate Iterative Boltzmann Inversion

Chris D. Jones [1¶], **Mazin Almarashi** [1], **Marjan Albooyeh** [2], **Eric Jankowski** [2], **and Clare McCabe** [1]

1 School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, Scotland, United Kingdom **2** Micron School of Material Science and Engineering, Boise State University, Boise, Idaho, United States ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Jed Brown

Reviewers:

- @Pablolbannez
- @valsson

Submitted: 30 September 2025
Published: unpublished

Summary

Iterative Boltzmann inversion (IBI) is a well-established, and widely used, method for deriving coarse-grained (CG) force fields that recreate the structural distributions of an underlying atomistic model. Multiple state IBI (MS-IBI) as introduced by Moore et al. (Moore et al., 2014), addresses state-point transferability limitations of IBI by including distributions from multiple state points to inform the derived CG force field. Here, we introduce `msibi`, a pure python package that implements the MS-IBI method for creating CG force fields for both intramolecular and intermolecular interactions. The package offers a user-friendly, Python-native API, eliminating the need for bash scripting and manual editing of input files. `msibi` is ultimately simulation engine agnostic, but uses the HOOMD-Blue simulation engine (Anderson et al., 2020) under-the-hood to perform iterative CG model simulations. This allows `msibi` to utilize graphical processing unit (GPU) acceleration without requiring users to manually compile GPU compatible code.

Statement of need

Molecular dynamics (MD) simulations are computationally intensive and scale poorly with the number of particles in the system, which limits accessible time and length scales. As a result, atomistic MD simulations of complex systems such as polymers and biomolecules become prohibitively expensive, especially as their relevant length and time scales often surpass micrometers and microseconds. Coarse-graining (CG) is a commonly adopted solution to this challenge, as it reduces computational cost by grouping—or mapping—atoms into a single, larger bead (Joshi & Deshmukh, 2021). Beyond being a technical tool to improve efficiency, coarse-graining also reflects modelling choices about which molecular features are essential to retain, and which can be simplified, which depends in large part on the materials being studied, the properties measured, and the researcher's objectives (Noid, 2013). As a result, this approach introduces two challenges: first, the potential energy surface for a given chemistry and CG mapping is not known *a priori*, and second, as the mapping used is arbitrary, with multiple valid options, developing a single CG force field that is transferable across various mapping choices is not possible. Also, a CG force field derived using IBI is state-point dependent with limited transferability to other state-points (Carbone et al., 2008; Moore et al., 2014). Consequently, developing a CG force field is required each time a new under-lying chemistry, mapping or target state-point is used. IBI and MS-IBI are popular choices for deriving CG forces for polymers and biomolecules (Carbone et al., 2008; Fritz et al., 2009; Jones et al., 2025; Moore et al., 2016; Tang et al., 2023). While these methods are frequently used, open-source software tools that provide an accessible and reproducible, end-to-end workflow for IBI and MS-IBI remain limited, especially for arbitrary mappings and multi-state systems.

The MARTINI force field is a widely adopted and successful CG model that employs a top-down CG approach, where interactions are optimized to reproduce experimental properties rather than structural distributions from atomistic simulations, as done in IBI and MSIBI (Marrink et al., 2007). This design provides excellent transferability and robustness across diverse systems, but its predefined bead types and fixed mapping rules limit flexibility for arbitrary resolutions or novel chemistries (Joshi & Deshmukh, 2021). The Versatile Object-oriented Toolkit for Coarse-graining Applications (VOTCA) offers a robust implementation of IBI—among several other features—and is also widely used in the community (Baumeier et al., 2024). However, its workflow relies on manual management of multiple input files and bash operations, which can introduce operational complexity that reduces reproducibility and usability (Jankowski et al., 2020; M. W. Thompson et al., 2020). Additionally, VOTCA's implementation of IBI does not natively support inclusion and weighting of multiple state points, as implemented in the MS-IBI method.

Here, `msibi` adds support for multiple state points, and is designed to easily manage successive CG force optimizations in series, where the learned force from the previous optimization is included and held fixed while the next force is optimized. This approach follows best practices for deriving CG force fields via IBI and MS-IBI (Reith et al., 2003). Additionally, we emphasize that `msibi` is ultimately engine agnostic: any simulation engine can be used to generate the fine-grained target structural distributions, and the CG force field produced by `msibi` is compatible with any simulation engine that supports tabulated forces. This includes LAMMPS (A. P. Thompson et al., 2022), Gromacs (Van Der Spoel et al., 2005), DL_Poly (Smith et al., 2002), and HOOMD-Blue (Anderson et al., 2020), among others. It is required that the target trajectories are converted to the gsd file format, which is the native file format for HOOMD-Blue. `msibi` includes a utility function that converts trajectory files to the gsd file format. This converter utility relies on the MDAnalysis package (Naughton et al., 2022) as a back-end to streamline file conversions. It works on all toplogy and trajectory formats that are supported in MDAnalysis, which includes LAMMPS, Gromacs, and CHARMM.

Using msibi

`msibi` contains three primary classes:

1. msibi.state.State:

This class encapsulates state-point information such as target trajectories, temperature, weighting factor and sampling parameters. Multiple instances of this class can be created, and each is used in deriving the final CG force field.

2. msibi.force.Force:

The base class from which all force types in `msibi` inherit from:

- **msibi.force.Bond:** Optimizes bond-stretching forces.
- **msibi.force.Angle:** Optimizes bond-bending forces.
- **msibi.force.Pair:** Optimizes non-bonded pair forces.
- **msibi.force.Dihedral:** Optimizes bond-torsion forces.

Users can include any number and combination of forces for MS-IBI simulations, though only one *type* of force can be optimized at a time.

Setting Force Parameters

There are multiple methods for defining the parameters of a `msibi.force.Force` instance:

- **Force.set_from_file:** Creates a tabulated force from a `.csv` file. This is useful for setting a previously optimized CG force while learning another.

- **Force.set_polynomial:** Creates a tabulated force from a polynomial function. This is helpful to setting initial guess forces, especially for distributions with multiple peaks.
- **Force.set_harmonic & Force.set_periodic** Creates a static, immutable force (not tabulated). This is useful for setting force parameters for distributions that are easily described by harmonic or periodic functions.

Example Workflow

The force-setter methods above enable users to combine learned and static forces and include them in series. For example:

1. Fit a bond-stretching force to a simple distribution and set the force using `Bond.set_harmonic()`.
2. With the bond-stretching force included and held static (step 1), run `msibi` to learn bond-angle forces, resulting in a tabulated force stored in a `.csv` file.
3. Set up and run a new `msibi` instance where `Bond.set_harmonic()` and `Angle.set_from_file()` create static intra-molecular forces and learn a non-bonded force.

3. msibi.optimize.MSIBI:

This class serves as the context manager for orchestrating optimization iterations. A single instance of this class is needed, and all instances of `msibi.state.State` and `msibi.force.Force` are attached to it before optimizations begin. This class also stores global simulation parameters such as timestep, neighbor list, exclusions, thermostat and trajectory write-out frequency.

Primary Methods

- **MSIBI.add_state** & **MSIBI.add_force:** Handle data management between states and forces.
- **MSIBI.run_optimization**: Runs iterative simulations and updates for all instances of `msibi.force.Force` being optimized.

The `run_optimization` method is designed for flexibility as it can be called multiple times, resuming from the last iteration. This enables use in:

- while loops: Run single iterations until a convergence criterion is met.
- for loops: Perform operations between batches of iterations. For example:
  - Smoothing the force
  - Adjusting state-point weighting
  - Modifying simulation criteria (e.g., extending optimization simulations as the force stabilizes).

The repository and documentation contain more detailed examples.

Availability

`msibi` is open-source and freely available under the MIT License on GitHub. We encourage users to ask questions, file issues and make contributions as applicable on the repository. `msibi` is available on the conda-forge ecosystem. For installation instructions and Python API documentation, visit the documentation.

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

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

Baumeier, B., Wehner, J., Renaud, N., Ruiz, F. Z., Halver, R., Madhikar, P., Gerritsen, R., Tirimbo, G., Sijen, J., Rosenberger, D., Brown, J. S., Sundaram, V., Krajniak, J., Bernhardt, M., & Junghans, C. (2024). VOTCA: Multiscale frameworks for quantum and classical simulations in soft matter. *Journal of Open Source Software*, *9*(99), 6864. https://doi.org/10.21105/joss.06864

Carbone, P., Varzaneh, H. A. K., Chen, X., & Müller-Plathe, F. (2008). Transferability of coarse-grained force fields: The polymer case. *The Journal of Chemical Physics*, *128*(6). https://doi.org/10.1063/1.2829409

Fritz, D., Harmandaris, V. A., Kremer, K., & Van Der Vegt, N. F. (2009). Coarse-grained polymer melts based on isolated atomistic chains: Simulation of polystyrene of different tacticities. *Macromolecules*, *42*(19), 7579–7588. https://doi.org/10.1021/ma901242h

Jankowski, E., Ellyson, N., Fothergill, J. W., Henry, M. M., Leibowitz, M. H., Miller, E. D., Alberts, M., Chesser, S., Guevara, J. D., Jones, C. D., Klopfenstein, M., Noneman, K. K., Singleton, R., Uriarte-Mendoza, R. A., Thomas, S., Estridge, C. E., & Jones, M. L. (2020). Perspective on coarse-graining, cognitive load, and materials simulation. *Computational Materials Science*, *171*, 109129. https://doi.org/10.1016/j.commatsci.2019.109129

Jones, C. D., Fothergill, J. W., Barrett, R., Ghanbari, L. N., Enos, N. R., McNair, O., Wiggins, J., & Jankowski, E. (2025). Representing structural isomer effects in a coarse-grain model of poly (ether ketone ketone). *Polymers*, *17*(1), 117. https://doi.org/10.3390/polym17010117

Joshi, S. Y., & Deshmukh, S. A. (2021). A review of advancements in coarse-grained molecular dynamics simulations. *Molecular Simulation*, *47*(10-11), 786–803. https://doi.org/10.1080/08927022.2020.1828583

Marrink, S. J., Risselada, H. J., Yefimov, S., Tieleman, D. P., & De Vries, A. H. (2007). The MARTINI force field: Coarse grained model for biomolecular simulations. *The Journal of Physical Chemistry B*, *111*(27), 7812–7824. https://doi.org/10.1021/jp071097f

Moore, T. C., Iacovella, C. R., Hartkamp, R., Bunge, A. L., & McCabe, C. (2016). A coarse-grained model of stratum corneum lipids: Free fatty acids and ceramide NS. *The Journal of Physical Chemistry B*, *120*(37), 9944–9958. https://doi.org/10.1021/acs.jpcb.6b08046

Moore, T. C., Iacovella, C. R., & McCabe, C. (2014). Derivation of coarse-grained potentials via multistate iterative boltzmann inversion. *The Journal of Chemical Physics*, *140*(22). https://doi.org/10.1063/1.4880555

Naughton, F. B., Alibay, I., Barnoud, J., Barreto-Ojeda, E., Beckstein, O., Bouysset, C., Cohen, O., Gowers, R. J., MacDermott-Opeskin, H., Matta, M., & others. (2022). MDAnalysis 2.0 and beyond: Fast and interoperable, community driven simulation analysis. *Biophysical Journal*, *121*(3), 272a–273a. https://doi.org/10.1016/j.bpj.2021.11.1368

Noid, W. G. (2013). Perspective: Coarse-grained models for biomolecular systems. *The Journal of Chemical Physics*, *139*(9). https://doi.org/10.1063/1.4818908

Reith, D., Pütz, M., & Müller-Plathe, F. (2003). Deriving effective mesoscale potentials from atomistic simulations. *Journal of Computational Chemistry*, *24*(13), 1624–1636. https://doi.org/10.1002/jcc.10307

Smith, W., Yong, C., & Rodger, P. (2002). DL_POLY: Application to molecular simulation. *Molecular Simulation*, *28*(5), 385–471. https://doi.org/10.1080/08927020290018769

Tang, J., Kobayashi, T., Zhang, H., Fukuzawa, K., & Itoh, S. (2023). Enhancing pressure consistency and transferability of structure-based coarse-graining. *Physical Chemistry Chemical Physics*, *25*(3), 2256–2264. https://doi.org/10.1039/D2CP04849C

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., & others. (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

Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., & Berendsen, H. J. (2005). GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, *26*(16), 1701–1718. https://doi.org/10.1002/jcc.20291