The Journal of Open Source Software

fastatomstruct: A High-Performance Library for Structural and Dynamical Analysis of Atomic Systems

Nils Holle [1], **Sebastian Walfort** [1], **Riccardo Mazzarello** [2], **and Martin Salinga** [1]

1 University of Münster, Institute of Materials Physics, Germany **2** Sapienza Università di Roma

DOI: 10.21105/joss.08106

Software

- Review
- Repository
- Archive

Editor: Christoph Junghans

Reviewers:

- @tukss
- @markcmiller86

Submitted: 24 February 2025
Published: 30 June 2025

Summary

`fastatomstruct` is a Python library, implemented primarily in Rust, designed to efficiently calculate a wide range of atomic structural and dynamical parameters. Using thread-based parallelization via Rust's `rayon` crate, the package provides fast and scalable performance for analyzing atomic configurations. In particular, many functions in this library are targeted at the structural and dynamical analysis of large liquid, supercooled-liquid, and amorphous systems. Its primary audience includes researchers in computational materials physics, materials science, and chemistry, though it is also broadly applicable to any field requiring detailed structural analysis of atomic systems.

The library provides a comprehensive suite of tools for analyzing atomic configurations, supporting the calculation of radial distribution functions, static structure factors, bond-orientational parameters, and vibrational density of states, among others. By leveraging MPI-based parallelization (`mpi4py`) alongside Rust's `rayon` crate for thread-based parallelization, `fastatomstruct` achieves high efficiency, making it particularly useful for large-scale simulations.

Statement of need

Understanding atomic structure and dynamics is fundamental in many scientific disciplines, including the study of phase transitions, the behavior of supercooled liquids, and the exploration of molecular interactions. Answering many questions in these fields requires simulations of large systems. Today, this is possible with quantum mechanical accuracy using machine-learned potentials (Friederich et al., 2021). However, efficiently calculating structural quantities from large atomic simulations can be computationally expensive. Existing tools often lack support for high-performance parallelization or the ability to integrate seamlessly with Python-based analysis workflows, such as those built around the widely used Atomic Simulation Environment (ASE) (Hjorth Larsen et al., 2017).

`fastatomstruct` addresses this gap by offering a Rust-backed implementation with efficient parallelization techniques that exploit multicore processors and thread-based parallelization. This allows researchers to investigate a wide range of atomic-scale phenomena. In materials science, the library facilitates the study of phase transitions, e.g., crystallization or melting, by tracking bond-orientational parameters and radial distribution functions, providing insights into how atomic structures evolve under changing conditions. In condensed matter physics and engineering, it enables the characterization of atomic defects, local structural distortions, and vibrational properties, aiding in the design of materials with specific desired properties. The library is particularly useful in the study of soft matter and supercooled liquids, where non-Gaussian transport behavior, mean squared displacement, and velocity autocorrelation functions help uncover the fundamental mechanisms of glassy dynamics. Furthermore, `fastatomstruct` provides tools for analyzing molecular structure through three-body correlations and other

structural parameters that go beyond the traditional two-body quantities like pair radial distribution functions.

The library's scalability makes it particularly suited for analyzing large datasets. For instance, computing the coordination numbers of a bulk structure containing half a million atoms can be completed in approximately 0.3 seconds on 96 cores (see Figure 2), demonstrating its ability to handle high-throughput simulations efficiently. Figure 1 shows a comparison of a radial distribution function calculation using `fastatomstruct` and the widely used `MDAnalysis` Python library. The calculations yield equivalent results, but `fastatomstruct` is approximately 20 times faster due to its parallelized implementation. In comparison to other commonly used packages such as `MDAnalysis` (Gowers et al., 2016; Michaud-Agrawal et al., 2011), `pymatgen` (Ong et al., 2013), or functions built into ASE, `fastatomstruct` distinguishes itself through its high-performance design and scalability. Its Rust-based core leverages advanced thread and MPI parallelization strategies to achieve substantially faster processing of large datasets, while ensuring memory and thread safety. Easy installation and direct integration with Python workflows and established tools like ASE further streamline its usage, avoiding the complexity often encountered with other codes.

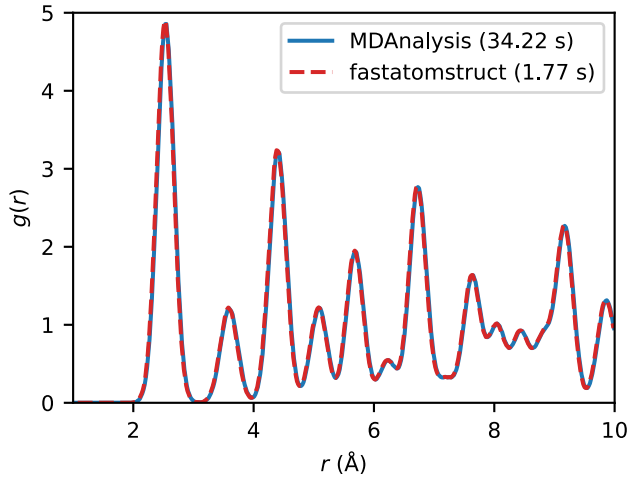

Figure 1: Comparison of radial distribution function calculation between `fastatomstruct` and `MDAnalysis` shows a $20\times$ speedup with the parallelized implementation in `fastatomstruct`. The RDF was calculated for a copper crystal with $108,000$ atoms. Calculations were performed on two Intel Xeon Gold 6140 processors with a total of 36 cores and threads. The speedup is mainly due to the parallelization of distance calculations. Note that we do not observe an ideal $36\times$ speedup as a) the simple distance calculations required for the computation of the radial distribution function do not scale as well as the calculation of more complex quantities like bond order correlation parameters (see Figure 3), b) some calculation time is spent on the non-parallelized creation of the histogram, and c) there is overhead due to the dual-processor setup.

Features and background

The `fastatomstruct` library calculates a variety of structural quantities that provide insights into atomic arrangements. Distances between atoms are calculated using a linear scaling method (see Figure 2), which also allows for fast calculations of quantities like coordination numbers. Spatial hashing enables efficient neighbor finding and distance calculations with $\mathcal{O}(N)$ scaling instead of the $\mathcal{O}(N^2)$ complexity of naive all-pairs approaches. This algorithm divides the simulation box into a regular grid of cells, where each cell has dimensions larger than the maximum interaction cutoff radius. Each atom's coordinates are converted into a cell index using a hash function based on discretized spatial coordinates—hence the term "spatial hashing". Atoms are assigned to cells based on these hashed coordinates, and neighbor searches only consider atoms within the same cell and its neighboring cells. For large systems, this dramatically reduces the number of pairwise distance calculations required.

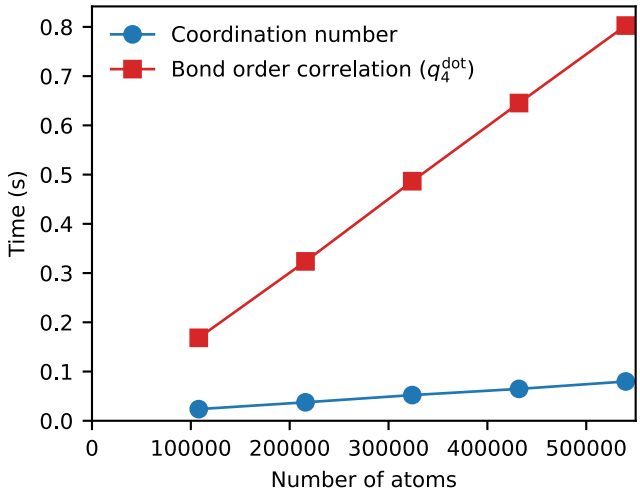

Figure 2: Linear scaling of distance calculations in `fastatomstruct` allows for efficient computation of structural quantities. The distance between atoms is calculated using spatial hashing, which divides the simulation box into cells and only considers atoms within neighboring cells. The calculations have been performed on an AMD EPYC Genoa 9654 processor with 96 cores and threads.

The implemented structural quantities range from basic coordination analysis to more sophisticated higher-order correlation functions that capture complex geometric motifs. Bond-orientational parameters (Steinhardt et al., 1983) and bond-order correlation parameters (Ronneberger et al., 2016) use spherical harmonics with smooth cutoff functions to distinguish between liquid and crystalline phases, while three-body correlations and their angular-limited variants (Bichara et al., 1993) are particularly effective for identifying Peierls-like distortions in disordered systems.

For dynamical analysis, the library provides tools to study both diffusive and vibrational motion. Mean squared displacement calculations leverage the efficient `tidynamics` library (Buyl, 2018), while non-Gaussian parameters quantify deviations from normal diffusion that indicate dynamical heterogeneities. Intermediate scattering functions (both coherent and incoherent) probe density fluctuations and single-particle motion, respectively. The overlap parameter and four-point susceptibility measure spatial correlations in dynamics (Lačević et al., 2003) and help identify dynamically correlated regions. Vibrational properties are accessible through velocity autocorrelation functions and vibrational density of states, while transport properties like viscosity can be calculated via Green-Kubo relations from stress tensor autocorrelations.

Table 1 provides a complete overview of all supported quantities, their mathematical definitions, and relevant literature references.

Table 1: Summary of structural and dynamical quantities supported by `fastatomstruct`. Structural quantities include basic measures like coordination numbers and radial distribution functions, as well as sophisticated tools for phase identification such as bond-orientational parameters (q_l) computed via spherical harmonics, and bond order correlation parameters (q_l^{dot}) that distinguish crystalline from amorphous environments. Three-body correlations (TBC) and angular-limited variants (ALTBC) capture geometric motifs and Peierls-like distortions, while tetrahedral order parameters quantify local tetrahedral coordination. Dynamical quantities encompass diffusive properties (mean squared displacement, non-Gaussian parameters), density fluctuation probes (intermediate scattering functions), dynamical correlation measures (overlap parameters, four-point susceptibility), and vibrational properties (velocity autocorrelation, vibrational density of states). Transport coefficients like viscosity are accessible through Green-Kubo relations. All calculations employ efficient algorithms with spatial hashing for neighbor finding and support both thread-based and MPI parallelization.

Quantity	Formula	Literature reference(s)
Structural Quantities		
Coordination Number	$\text{CN} = \sum_j \Theta(r_{\text{cut}} - \|\mathbf{r}_i - \mathbf{r}_j\|)$	Standard structural analysis
Radial Distribution Function	$g(r) = \frac{dn_r}{4\pi\rho r^2 dr}$	Standard statistical mechanics
Static Structure Factor	$S(q) = 1 + 4\pi\rho \int_0^\infty r^2[g(r) - 1]\frac{\sin(qr)}{qr} dr$	Standard scattering theory
Bond Angle Distribution	$\cos\theta_{ijk} = \frac{\mathbf{r}_{ij} \cdot \mathbf{r}_{ik}}{r_{ij} r_{ik}}$	Standard geometric analysis
Three-Body Correlation	$g^{(3)}(r_1, r_2) = \frac{V}{\rho(N-1)(N-2)} \sum_{i_1,i_2,i_3} \langle \delta(r_1 - r_{i_1 i_2}) \delta(r_2 - r_{i_2 i_3}) \rangle$	Ronneberger et al. (2016)
Angular-Limited TBC	$g^{(3)}_{\text{AL}}(r_1, r_2) = \frac{V}{\rho(N-1)(N-2)} \sum_{i_1,i_2,i_3} \langle \delta(r_1 - r_{i_1 i_2}) \delta(r_2 - r_{i_2 i_3}) \Theta(\beta - \delta) \rangle$	Bichara et al. (1993)
Bond Length Ratio	$\text{ALBLR} = \frac{\sum \frac{\max(r_{12}, r_{23})}{\min(r_{12}, r_{23})} \Theta(\beta-\delta) \bar{\Theta}(r_{12}) \bar{\Theta}(r_{23})}{\sum \Theta(\beta-\delta) \bar{\Theta}(r_{12}) \bar{\Theta}(r_{23})}$	Holle et al. (2025)
Bond-Orientational Parameters	$q_l = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \|q_{lm}\|^2}$	Steinhardt et al. (1983)
Bond Order Correlation	$q_l^{\text{dot}}(i) = \frac{1}{N_i^{\text{eff}}} \sum_j f(r_{ij}) C_{ij}$	Ronneberger et al. (2016), ten Wolde et al. (1995)
Tetrahedral Order Parameter	$q_{\text{tetrahedral}} = 1 - \frac{3}{8} \sum_{i>k} \left(\frac{1}{3} + \theta_{ijk}\right)^2$	Lee et al. (1993), Duboué-Dijon & Laage (2015)

Quantity	Formula	Literature reference(s)
Dynamical Quantities		
Mean Squared Displacement	$\mathrm{MSD}(t) = \frac{1}{N}\sum_{i=1}^{N}\langle\|\mathbf{r}_i(t) - \mathbf{r}_i(0)\|^2\rangle$	Standard diffusion analysis, Buyl (2018)
Non-Gaussian Parameter	$\alpha_2(t) = \frac{\frac{1}{N}\sum_{i=1}^{N}\langle\|\mathbf{r}_i(t)-\mathbf{r}_i(0)\|^4\rangle}{\left(\frac{1}{N}\sum_{i=1}^{N}\langle\|\mathbf{r}_i(t)-\mathbf{r}_i(0)\|^2\rangle\right)^2} - 1$	Standard diffusion analysis
Intermediate Scattering Function	$F(\boldsymbol{q},t) = \frac{1}{N}\sum_{i,j}^{N}\langle\exp[i\boldsymbol{q}\cdot(\boldsymbol{r}_i(t)-\boldsymbol{r}_j(0))]\rangle$	Standard scattering theory
Incoherent ISF	$F_{\mathrm{incoh}}(\boldsymbol{q},t) = \frac{1}{N}\sum_{i}^{N}\langle\exp[i\boldsymbol{q}\cdot(\boldsymbol{r}_i(t)-\boldsymbol{r}_i(0))]\rangle$	Standard scattering theory
Overlap Parameter	$Q(t) = \sum_{i,j=1}^{N} w(\|\mathbf{r}_i(0) - \mathbf{r}_j(t)\|)$	Lačević et al. (2003)
Four-Point Susceptibility	$\chi_4(t) = \frac{\beta V}{N^2}\langle Q^2(t)\rangle - \langle Q(t)\rangle^2$	Lačević et al. (2003)
Velocity Autocorrelation	$C_v(t) = \frac{1}{N}\sum_{i=1}^{N}\langle\mathbf{v}_i(0)\cdot\mathbf{v}_i(t)\rangle$	Standard vibrational analysis
Vibrational Density of States	Fourier transform of VACF or direct FFT of velocities	Schmerler (2025)
Viscosity	$\eta = \frac{V}{k_\mathrm{B}T}\lim_{T\to\infty}\int_0^T\langle\sigma_{xy}(t')\sigma_{xy}(0)\rangle dt'$	Cui et al. (1996)

Finally, we would like to note that parallelization is a core strength of the library. Thread-based parallelization using Rust's `rayon` library ensures efficient use of multicore processors, while image-based parallelization with MPI enables scalability to datasets with many different snapshots. Figure 3 shows the scaling behavior of different functions implemented in `fastatomstruct`. The library's compatibility with ASE and other Python-based tools further simplifies integration and usability for researchers. The Rust implementation has several advantages over more traditional approaches like combining Python with C or Fortran (Perkel, 2020). By design, Rust guarantees both memory and thread safety, which excludes entire classes of bugs that are common in other languages. In particular, Rust's ownership concept prevents race conditions, which makes it straightforward to write parallel code that is both fast and correct. The Rust compiler provides extensive static analysis, which can catch many bugs at compile time. As a result, `fastatomstruct` is a very robust and reliable library for structural and dynamical analysis of atomic systems. We hope that our approach using a rather non-traditional language for scientific computing can inspire other works to adopt Rust for high-performance computing tasks. Besides the advantages over languages like C/C++ or Fortran mentioned above, Rust also offers modern tooling and easy integration with Python through the `Py03` crate, and it can be interfaced with existing C/C++ or Fortran code. Increased usage of Rust should also improve the overall quality of scientific software by finding and eliminating common bugs very early in the development process.

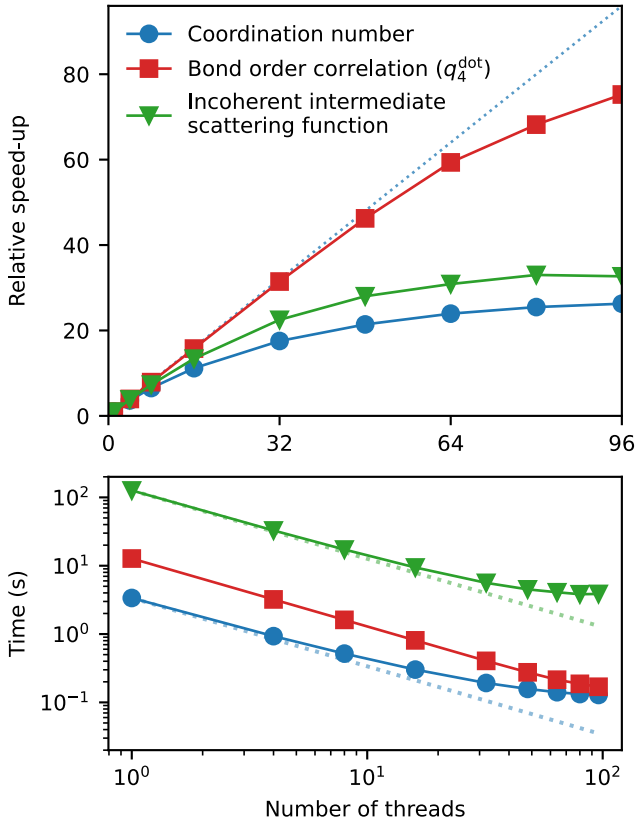

Figure 3: Scaling behavior of exemplary calculations with `fastatomstruct`. Relative speed-ups compared to a single thread (top) and total calculation times (bottom) are shown for calculations of coordination numbers, bond order correlation, and the incoherent intermediate scattering function. Tests have been performed on an AMD EPYC Genoa 9654 processor with 96 cores and threads. A system with $32,000$ atoms was used for the calculation of the bond order correlation parameter q_4^{dot}. $256,000$ atoms were used for coordination number calculations. The incoherent intermediate scattering function was calculated for 600 snapshots from a molecular dynamics simulation with $32,000$ atoms. Scalability for large thread counts is limited because the calculations are memory bound, which particularly holds for the intermediate scattering function calculations. The repository contains an example (`examples/perf_intermediate_scattering`) to demonstrate this.

Applications in Research

The `fastatomstruct` package enables the study of a wide range of structural and dynamical phenomena, while integrating well into many existing workflows based on the Atomic Simulation Environment (Hjorth Larsen et al., 2017). For instance, researchers can analyze phase transitions in amorphous materials by calculating bond-orientational parameters, which provide insights into changes in local atomic order. The library is particularly suited for studying the dynamics of supercooled liquids and glasses. This has been exploited in recent studies of liquid, supercooled-liquid and glassy antimony (Holle et al., 2025).

Acknowledgments

We acknowledge support from the German Research Foundation (DFG) through the collaborative research centers Nanoswitches (SFB 917) and Intelligent Matter (SFB 1459) as well as the European Research Council (ERC-Grant No. 640003). RM acknowledges funding from the PRIN 2020 project "Neuromorphic devices based on chalcogenide heterostructures" funded by the Italian Ministry for University and Research (MUR). Calculations for this publication were performed on the HPC cluster PALMA II of the University of Münster, subsidized by the DFG (INST 211/667-1).

References

Bichara, C., Pellegatti, A., & Gaspard, J.-P. (1993). Properties of liquid group-V elements: A numerical tight-binding simulation. *Physical Review B*, *47*(9), 5002–5007. https://doi.org/10.1103/PhysRevB.47.5002

Buyl, P. de. (2018). Tidynamics: A tiny package to compute the dynamics of stochastic and molecular simulations. *Journal of Open Source Software*, *3*(28), 877. https://doi.org/10.21105/joss.00877

Cui, S. T., Cummings, P. T., & Cochran, H. D. (1996). The calculation of the viscosity from the autocorrelation function using molecular and atomic stress tensors. *Molecular Physics*, *88*(6), 1657–1664. https://doi.org/10.1080/00268979609484542

Duboué-Dijon, E., & Laage, D. (2015). Characterization of the Local Structure in Liquid Water by Various Order Parameters. *The Journal of Physical Chemistry B*, *119*(26), 8406–8418. https://doi.org/10.1021/acs.jpcb.5b02936

Friederich, P., Häse, F., Proppe, J., & Aspuru-Guzik, A. (2021). Machine-learned potentials for next-generation matter simulations. *Nature Materials*, *20*(6), 750–761. https://doi.org/10.1038/s41563-020-0777-6

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. *Scipy*. https://doi.org/10.25080/Majora-629e541a-00e

Hjorth Larsen, A., Jørgen Mortensen, J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Bjerre Jensen, P., Kermode, J., Kitchin, J. R., Leonhard Kolsbjerg, E., Kubal, J., Kaasbjerg, K., Lysgaard, S., … Jacobsen, K. W. (2017). The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter*, *29*(27), 273002. https://doi.org/10.1088/1361-648X/aa680e

Holle, N., Walfort, S., Ballmaier, J., Mazzarello, R., & Salinga, M. (2025). Importance of Density for Phase-Change Materials Demonstrated by Ab Initio Simulations of Amorphous

Antimony. *Physical Review Letters*, *134*(4), 046101. https://doi.org/10.1103/PhysRevLett.134.046101

Lačević, N., Starr, F. W., Schrøder, T. B., & Glotzer, S. C. (2003). Spatially heterogeneous dynamics investigated via a time-dependent four-point density correlation function. *The Journal of Chemical Physics*, *119*(14), 7372–7387. https://doi.org/10.1063/1.1605094

Lee, C., Vanderbilt, D., Laasonen, K., Car, R., & Parrinello, M. (1993). Ab initio studies on the structural and dynamical properties of ice. *Physical Review B*, *47*(9), 4863–4872. https://doi.org/10.1103/PhysRevB.47.4863

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

Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., & Ceder, G. (2013). Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, *68*, 314–319. https://doi.org/10.1016/j.commatsci.2012.10.028

Perkel, J. M. (2020). Why scientists are turning to Rust. *Nature*, *588*(7836), 185–186. https://doi.org/10.1038/d41586-020-03382-2

Ronneberger, I., Mazzarello, R., & Wuttig, M. (2016). *Computational Study of Crystallization Kinetics of Phase Change Materials* [PhD thesis, RWTH Aachen University]. https://doi.org/10.18154/RWTH-2017-00376

Schmerler, S. (2025). *Elcorto/pwtools*. https://doi.org/10.5281/zenodo.13128303

Steinhardt, P. J., Nelson, D. R., & Ronchetti, M. (1983). Bond-orientational order in liquids and glasses. *Physical Review B*, *28*(2), 784–805. https://doi.org/10.1103/PhysRevB.28.784

ten Wolde, P. R., Ruiz-Montero, M. J., & Frenkel, D. (1995). Numerical Evidence for bcc Ordering at the Surface of a Critical fcc Nucleus. *Physical Review Letters*, *75*(14), 2714–2717. https://doi.org/10.1103/PhysRevLett.75.2714