matscipy: materials science at the atomic scale with Python

Petr Grigorev 1*, Lucas Frérot 2*, Fraser Birks 3, Adrien Gola 2,4, Jacek Golebiowski 4, Jan Grießer 2, Johannes L. Hörmann 2,8, Andreas Klemenz 5, Gianpietro Moras 5, Wolfram G. Nöhring 2, Jonas A. Oldenstaedt 2, Punit Patel 4, Thomas Reichenbach 5, Thomas Rocke 3, Lakshmi Shenoy 3, Michael Walter 8, Simon Wengert 7, Lei Zhang 9, James R. Kermode 3*, and Lars Pastewka 2,8*

1 Aix-Marseille Université, CNRS, CINaM UMR 7325, Campus de Luminy, 13288 Marseille, France
2 Department of Microsystems Engineering, University of Freiburg, 79110 Freiburg, Germany
3 Warwick Centre for Predictive Modelling, School of Engineering, University of Warwick, Coventry CV4 7AL, United Kingdom
4 Institute for Applied Materials, Karlsruhe Institute of Technology, Engelbert-Arnold-Straße 4, 76131 Karlsruhe, Germany
5 Fraunhofer IWM, MikroTribologie Centrum µTC, Wöhlerstraße 11, 79108 Freiburg, Germany
6 Department of Materials, Imperial College London, London SW7 2AZ, United Kingdom
7 Fritz Haber Institute of the Max Planck Society, Faradayweg 4-6, 14195 Berlin, Germany
8 Cluster of Excellence livMatS, Freiburg Center for Interactive Materials and Bioinspired Technologies, University of Freiburg, Georges-Köhler-Allee 105, 79110 Freiburg, Germany
9 Engineering and Technology Institute Groningen, Faculty of Science and Engineering, University of Groningen, Nijenborgh 4, 9747 AG Groningen, The Netherlands

* Corresponding author
¶ These authors contributed equally.

Summary

Behaviour of materials is governed by physical phenomena that occur at an extreme range of length and time scales. Computational modelling requires multiscale approaches. Simulation techniques operating on the atomic scale serve as a foundation for such approaches, providing necessary parameters for upper-scale models. The physical models employed for atomic simulations can vary from electronic structure calculations to empirical force fields. However, construction, manipulation and analysis of atomic systems are independent of the given physical model but dependent on the specific application. matscipy implements such tools for applications in materials science, including fracture, plasticity, tribology and electrochemistry.

Statement of need

The Python package matscipy contains a set of tools for researchers using atomic-scale models in materials science. In atomic-scale modelling, the primary numerical object is a discrete point in three-dimensional space that represents the position of an individual atom. Simulations are often dynamical, where configurations change over time and each atom carries a velocity. Complexity emerges from the interactions of many atoms, and numerical tools are required for generating initial atomic configurations and for analysing output of such dynamical simulations, most commonly to connect local geometric arrangements of atoms to physical processes. An example, described in more detail below, is the detection of the tip of a crack that moves through a solid body.

We never see individual atoms at macroscopic scales. To understand the behaviour of everyday objects, atomic-scale information needs to be transferred to the continuum scale. This is the
primary objective of multi-scale modelling. *matscipy* focuses on atomic representations of materials, but implements tools for connecting to continuum descriptions in mechanics and transport theory. Each of the application domains described in the following therefore relies on the computation of continuum fields, that is realised through analytic or numerical solutions.

There is no other package that we are aware of, which fills the particular niche of the application domains in the next section. The package addresses the boundary between atomic-scale and continuum modelling in materials with particular emphasis on plasticity, fracture and tribology. The *atomman* atomistic manipulation toolkit (Hale, 2022) and the *atomsk* package (Hirel, 2015) have some overlapping objectives but are restricted to a narrower class of materials systems, principally point defects, stacking faults and dislocations. We target interoperability with the widely used Atomic Simulation Environment (ASE) (Larsen et al., 2017), which offers great flexibility by providing a Python interface to tens of simulation codes implementing different physical models. While ASE’s policy is to remain pure Python, we augment some of ASE’s functionality with more efficient implementations in C, such as the computation of the neighbour list. Large scale molecular dynamics (MD) simulations are most efficiently performed with optimised codes such as LAMMPS (Thompson et al., 2022), with *matscipy*’s main contribution being to set up input structures and to post-process output trajectories.

The central class in ASE is the *Atoms* class, which is a container that stores atomic positions, velocities and other properties. Calculators describe relationships between atoms, and are used for example to compute energies and forces, and can be attached to *Atoms* objects. All other *matscipy* functionality is implemented as functions acting on *Atoms* objects. This is comparable to the design of *numpy* (Harris et al., 2020) or *scipy* (Virtanen et al., 2020), that are collections of mathematical functions operating on core array container objects. In our experience, separating code into functions and containers lowers the barrier to entry for new users and eases testability of the underlying code base.

*matscipy* is a toolbox that enables multi-scale coupling, but it is not a toolbox for actually carrying out two-way coupled calculations. Its target is the construction of atomic domains from continuum information and the extraction of continuum fields from atomic structures. Other packages exist that take care of the actual, two-way coupling. In contrast to *matscipy*, those have a primary focus on handling discretised continuum fields, typically in the form of finite-element meshes, and interpolating nodal or element values between atomic-scale and continuum descriptions. *matscipy* itself has no provisions for handling discrete continuum data, but does implement analytic expressions for continuum fields.

Example implementations of two-way coupling codes are the open-source code *libmultiscale* (Guillaume Anciaux, 2007; G. Anciaux et al., 2018) that explicitly targets atomistic-continuum coupling, or the generic coupling libraries *preCICE* (Chourdakis et al., 2022) or *MpCCI* (Dehning et al., 2015). Another two-way coupling code is *MultiBench* (Miller & Tadmor, 2009), that was specifically designed for benchmarking a wide range of two-way atomistic-continuum coupling schemes. Furthermore, there are specialised multiscale coupling code, such as Green’s function molecular dynamics (GFMD) (Campañá & Müser, 2006; Pastewka et al., 2012) which targets two-way coupling in contact mechanics and friction simulations. All of these packages have only limited capabilities for constructing atomistic domains. *matscipy* could be combined with these packages for two-way coupled simulation of plasticity, fracture or frictional processes.

**Application domains**

Within materials science, the package has different application domains:

- **Elasticity.** Solids respond to small external loads through a reversible elastic response. The strength of the response is characterised by the elastic moduli. *matscipy.elasticity* implements functions for computing elastic moduli from small deformation that consider potential symmetries of the underlying atomic system, in particular for crystals. The implementation also includes estimates of uncertainty on elastic moduli - either from...
a least-squares error, or from a Bayesian treatment if stress uncertainty is supplied. 
matscipy also implements analytic calculation of elastic moduli for some interatomic 
potentials, described in more detail below. The computation of elastic moduli is a 
prerequisite for multi-scale modelling of materials, as they are the most basic parameters 
of continuum material models. matscipy was used to study finite-pressure elastic 
constants and structural stability in crystals (Grießer, Frérot, et al., 2023) and glasses 
(Grießer, Moras, et al., 2023).

- **Plasticity.** For large loads, solids can respond with irreversible deformation. One form 
of irreversibility is plasticity, that is carried by extended defects, the dislocations, in 
crystals. The module matscipy.dislocation implements tools for studying structure 
and movement of dislocations. Construction and analysis of model atomic systems is 
implemented for compact and dissociated screw, as well as edge dislocations in cubic 
crystals. The implementation supports ideal straight as well as kinked dislocations. Some 
of the dislocation functionality requires the atomman and/or OVITO packages as optional 
dependencies (Hale, 2022; Stukowski, 2009). The toolkit can be applied to a wide range 
of single- and multi-component ordered systems, and could be used as an initial starting 
point for modelling dislocations in systems with chemical disorder. The module was 
employed in a study of interaction of impurities with screw dislocations in tungsten 
(Grigorev et al., 2020, 2023).

- **Fracture mechanics.** Cracking is the process of generating new surface area by splitting 
the material apart. The module matscipy.fracture_mechanics provides functionality 
for calculating continuum linear elastic displacement fields near crack tips, including 
support for anisotropy in the elastic response (Sih et al., 1965). The module also 
implements generation of atomic structures that are deformed according to this near-tip 
field. This functionality has been used to quantify lattice trapping, which is the pinning 
of cracks due to the discreteness of the atomic lattice, and to compare simulations with 
experimental measurements of crack speeds in silicon (Kermode et al., 2015). There is 
also support for flexible boundary conditions in fracture simulations using the formalism 
proposed by Sinclair (Sinclair, 1975), where the finite atomistic domain is coupled to an 
infinite elastic continuum. Finally, we provide an extension of this approach to give a 
flexible boundary scheme that uses numerical continuation to obtain full solution paths 
for cracks (Buze & Kermode, 2021).

- **Tribology.** Tribology is the study of two interfaces sliding relative to one another, 
as encountered in frictional sliding or adhesion. Molecular dynamics simulations of 
representative volume elements of tribological interfaces are routinely used to gain 
insights into the atomistic mechanisms underlying friction and wear. The module 
matscipy.pressurecoupling provides tools to perform such simulations under a constant 
normal load and sliding velocity. It includes an implementation of the pressure coupling 
algorithm described by Pastewka et al. (2010). By dynamically adjusting the distance 
between the two sliding surfaces according to the local pressure, the algorithm ensures 
mechanical boundary conditions that account for the inertia of the bulk material which 
is not explicitly included in the simulation. This algorithm was used to study friction 
(Seidl et al., 2021) and wear (G. Moras et al., 2011; G. Moras et al., 2018; Pastewka et 
al., 2011; Peguiron et al., 2016; Reichenbach et al., 2021).

- **Electrochemistry.** Electrochemistry describes the motion and spatial distribution of 
charged atoms and molecules (ions) within an external electric field. Classical treatment of 
charged systems leads to continuous fields that describe mean concentration distributions, 
while true atomic systems consist of discrete particles with fixed charges. Neither do 
continuum models account for structured layering of ions in a polar solvent like water 
nor do they describe finite size effects at high concentrations such as densely packed 
monolayers. Sampling discrete particle positions from smooth distributions may, however, 
yield good initial configurations that accelerate equilibration in atomistic calculations. The 
matscipy.electrochemistry module provides tools that statistically sample discrete
coordinate sets from continuum fields and apply steric corrections (Martinez et al., 2009) to avoid overlap of finite size species. To generate continuum concentration distributions, the package also contains a control-volume solver (Selberherr, 1984) for the one-dimensional Poisson–Nernst–Planck equations (Bazant et al., 2006), as well as an interface to the finite-element solver FEniCS (Logg et al., 2012).

**All-purpose atomic analysis tools**

As well as these domain-specific tools, matscipy contains general utility functionality which is widely applicable:

- **Neighbour list.** An efficient linear-scaling neighbour list implemented in C delivers orders-of-magnitude faster performance for large systems than the pure Python implementation in ASE (Larsen et al., 2017), see Figure 1. The neighbour list is stored in a data structure comparable to coordinate (COO) sparse matrix storage format (Saad, 1990), where two arrays contain the indices of the neighbouring atoms and further arrays store distance vectors, absolute distances, and other properties associated with an atomic pair. This allows compact code for evaluating properties that depend on pairs, such as pair-distribution function or interatomic potential energies and forces. Most of the tools described in the following rely on this neighbour list format. The neighbour list is becoming widely used for post-processing and structural analysis of the trajectories resulting from molecular dynamics simulations, and even to accelerate next-generation message passing neural networks such as MACE (Batatia, Kovacs, et al., 2022; Batatia, Batzner, et al., 2022).

![Figure 1: Execution time of the computation of the neighbour list in ASE and matscipy. These results were obtained on a single core of an Intel i7-1260P processor on the ASE master branch (git hash 52a8e783).](image)

- **Atomic strain.** Continuum mechanics is formulated in terms of strains, which characterises the fractional shape changes of small volumes. Strains are typically only well-defined if averaged over sufficiently large volumes, and extracting strain fields from atomicscale calculations is notoriously difficult. matscipy implements calculations of strain by observing changes in local atomic neighbourhoods across trajectories. It fits a per-atom displacement gradient that minimises the error in displacement between two configurations as described by M. L. Falk & Langer (1998). The error resulting from this fit quantifies the non-affine contribution of the overall displacement and is known as $D_\text{min}^2$. We used this analysis to quantify local strain in the deformation of crystals (Gola et al., 2019, 2020) and glasses (Jana & Pastewka, 2019).

• **Radial, spatial and angular correlation functions.** Topological order in atomic-scale systems is often characterised by statistical measures of the local atomic environment. The simplest one is the pair-distribution or radial-distribution function, that gives the probability \( g_2(r) \) of finding an atom at distance \( r \). For three atoms, we can define a probability of finding a specific angle, yielding the angular correlation functions. matscipy has utility function for computing these correlation functions to large distances, including the correlation of arbitrary additional per-atom properties such as per-atom strains.

• **Ring analysis.** Topological order in network glasses can be characterised by statistics of shortest-path rings (Franzblau, 1991). matscipy implements calculations of these rings using a backtracking algorithm in C. We regularly use matscipy to characterise shortest-path rings in amorphous carbon (Jana & Pastewka, 2019; Pastewka et al., 2008).

• **Topology building for non-reactive MD simulations.** Non-reactive force fields for MD simulations consist of non-bonded and bonded interaction terms (Jorgensen et al., 1996). The latter require an explicit specification of the interatomic bonding topology, i.e. which atoms are involved in bond, angle and dihedral interactions. matscipy provides efficient tools to generate this topology for an atomic structure based on matscipy’s neighbour list, and then assign the relevant force field parameters to each interaction term. Input and output routines for reading and writing the corresponding control files for LAMMPS (Thompson et al., 2022) are also available. We used this functionality in various studies on tribology, wetting and nanoscale rheology (K. Falk et al., 2020, 2022; Goeldel et al., 2021; Mayrhofer et al., 2016; Reichenbach et al., 2020).

### Interatomic potentials and other calculators

Besides generating and analysing atomic-scale configurations, matscipy implements specific interatomic potentials (Müser et al., 2023). The goal here is not to provide the most efficient implementation of computing interatomic forces. We rather aim to provide simple implementations for testing new functional forms, or testing new features such as the computation of derivatives of second order.

• **Interatomic potentials.** The module matscipy.calculators has implementations of classical pair-potentials, Coulomb interactions, the embedded-atom method (EAM) (Daw & Baskes, 1984) and other many-body potentials (e.g. Stillinger & Weber, 1985; Tersoff, 1989).

• **Second-order derivatives.** The thermodynamic and elastic properties of solid materials are closely connected to the Hessian of the overall system, which contains the second derivatives of the total energy with respect to position and macroscopic strains. matscipy implements analytic second-order potential derivatives for pair-potentials (Lennard-Jones, 1931), EAM potentials (Daw & Baskes, 1984), bond-order potentials (Brenner, 1990; Kumagai et al., 2007; Tersoff, 1989), cluster potentials (Stillinger & Weber, 1985) and electrostatic interaction (Van Beest et al., 1990). This is achieved through a generic mathematical formulation of the manybody total energy (Grießer, Frérot, et al., 2023; Müser et al., 2023) in matscipy.calculators.manybody. The module matscipy.numerical additionally provides routines for the numerical (finite-differences) evaluation of these properties. These analytic second-order derivatives allow a fast and accurate computation of the aforementioned properties in crystals, polymers and amorphous solids, even for unstable configurations where numerical methods are not applicable.

• **Quantum mechanics/molecular mechanics.** The module matscipy.calculators.mcfm implements a generalised force-mixing potential (Bernstein et al., 2009) with support for multiple concurrent QM clusters, named MultiClusterForceMixing (MCFM). It has been

---

applied to model failure of graphene-nanotube composites (Gołębiowski et al., 2018, 2020).

- Committee models. The module matscipy.calculators.committee provides support for committees of interatomic potentials with the same functional form but differing parameters, in order to allow the effect of the uncertainty in parameters on model predictions to be estimated. This is typically used with machine learning interatomic potentials (MLIPs). The implementation follows the approach of (Musil et al., 2019) where the ensemble of models is generated by training models on different subsets of a common overall training database.

Acknowledgements

We thank Arnaud Allera, Manuel Aldegunde, Kristof Bal, James Brixey, Alexander Held, Jan Jansen, Till Junge, Henry Lambert and Zhilin Zheng for contributions and bug fixes. matscipy was partially funded by the Deutsche Forschungsgemeinschaft (projects 258153560, 390951807 and 461911253), the European Research Council (ERC StG 757343), the European Commission (NOMAD project grant agreement 951786 and ENTENTE project grant agreement 900018), the Engineering and Physical Sciences Research Council (grants EP/P002188/1, EP/R012474/1, EP/R043612/1 and EP/S022848/1) and the Leverhulme Trust (grant RPG-2017-191).

References


