

FIRECODE: A Computational Chemistry Workflow Driver and Modular Hub

Nicolò Tampellini ¹ ¶

¹ Massachusetts Institute of Technology, USA ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Computational chemistry, the branch of this science dedicated to the modeling of molecular systems for the extraction of properties of interest, is experiencing a recent renaissance thanks to impressive advances in theoretical models enabling ever-faster simulations of complex processes via density functional theory (DFT), semiempirical, and machine-learned interatomic potentials (MLIPs). However, the reduced computational cost will only translate into faster and more robust modeling with the help of automation frameworks, where trivial human intervention does not become the new bottleneck of computational pipelines.

Statement of need

FIRECODE is a modular workflow driver for computational chemistry written in Python. The code interfaces with a series of fast modern calculators (xTB ([Bannwarth et al., 2019](#)), tblite ([GitHub - Tblite/Tblite](#)), AIMNET2 ([Anstine, 2024](#)), UMA ([Wood et al., 2025](#)), usually via ASE ([Ase / Ase · GitLab — Gitlab.com](#))) and exposes a series of workflows and utilities for geometry optimization, frequency analysis, one- and two-ended transition state search, and conformational search. When possible, these routines are implemented in a calculator-independent way, and almost all of these can be arbitrarily chained to one another to define a complex workflow.

The software was born out of necessity over the last five years of research in asymmetric catalysis: FIRECODE was designed to streamline the necessary use of multiple software and routines in the computational modeling of small molecules, in a single centralized hub. The program can be called via command line with a minimal plain text input file defining the workflow, or used as a Python library in jupyter notebooks. The modular nature of the software enables rapid addition of new features, facilitating external contributions, while also retaining a small installation footprint for the core library. The concise input syntax also offers a streamlined definition of workflows and software orchestration with no coding experience required.

State of the field

While a manifold of computational chemistry software exists, workflow utilities are less common, particularly in the free and open source community. Recently, version 6.0 of ORCA ([Neese et al., 2020; Neese, 2022](#)) (free for academic use) started offering compound job functionality, which enables the chained execution of some core ORCA modules. The code is however not open source and functionality is limited to what is included in the ORCA program (*i.e.* no MLIPs, no similarity pruning, no ensemble processing...). While ORCA workflows are more extensively implemented in WEASEL ([Faccts](#)), this is commercial, closed-source software.

39 The excellent AQME software ([Alegre-Requena et al., 2023](#)) features the most similar design
40 philosophy to the one of this work. Key differences with FIRECODE, besides code architecture,
41 lie in: 1) the main interaction platform of AQME happening via jupyter notebooks, while the
42 software of this work favors the use of plain text input files, keeping the inputs as minimal
43 as possible. 2) The present software having a reactivity-centered set of features, chiefly via
44 fast semiempirical and ML calculators, over AQME's focus on DFT and on the generation of
45 ensemble properties like spectra and molecular descriptors.

46 Other examples of related software, targeting the exposure of high-level primitives over
47 structured workflows, include Cuby ([GitHub - Berquist/Cuby](#); [Řezáč, 2016](#)) (MD and high-level
48 DFT) written in Ruby, and the popular ASE ([Ase / Ase · GitLab — Gitlab.com](#)), which is
49 extensively used as a lower level interface for interacting with calculators in FIRECODE.

50 The development of this software from scratch was motivated by multiple reasons. First, a
51 modern and flexible hub for running computational chemistry workflows, particularly tailored
52 to modeling chemical reactivity and MLIPs, was lacking. While the benefit of developing
53 standalone applications for specific tasks remains, their integration in a single hub permits
54 automation and lowers the barrier of entry for adoption of both individual tools and workflows
55 alike. Second, modern machine-learned interatomic potentials (MLIPs) are more conveniently
56 ran via Python libraries like torch ([GitHub - Pytorch/Pytorch](#)), benefitting greatly from a
57 Python-based workflow manager. Additional features like implicit delta solvation for gas phase-
58 trained MLIPs ¹ and numerical quasi-RRHO thermochemistry ([Grimme, 2012](#)) are additional
59 core library functionalities that were not available in other actively maintained modules at the
60 time of development.

61 Software design

62 The software is developed with the philosophy of maintaining the greatest amount of
63 functionality and module interoperability with the minimal list of dependencies, which should
64 ideally be as modern and lean as possible. All core dependencies are distributed via pypi,
65 keeping the installation of the minimal version of the program down to a single command (`uv`
66 `pip install firecode`).

67 Non-essential dependencies can be installed based on the desired calculator and interfaces to
68 be used, all via conda/mamba.

69 The software implements various calculators (xTB ([Bannwarth et al., 2019](#)), tblite ([GitHub - Tblite/Tblite](#)),
70 AIMNET2 ([Anstine, 2024](#)), UMA ([Wood et al., 2025](#))) via ASE ([Ase / Ase · GitLab — Gitlab.com](#)), which can be used interchangeably in various ensemble routines.
71 Other core libraries leveraged throughout the codebase are numpy ([GitHub - Numpy/Numpy](#))
72 for algebraic manipulations, networkx ([GitHub - Networkx/Networkx](#)) for graph utilities and
73 prism_pruner ([GitHub - Ntampellini/Prism_pruner](#)) for the removal of duplicates. Sella
74 ([GitHub - Zadorlab/Sella](#)) provides a saddle point optimizer, and rdkit ([GitHub - Rdkit/Rdkit](#))
75 is used for ETKDG ([Riniker & Landrum, 2015](#)) conformational searches and SMARTS substructure
76 matching. The ML-FSM library ([GitHub - Thegomeslab/ML-FSM](#); [Marks & Gomes, 2025](#))
77 provides a two-ended TS search utility complementing ASE's implementation of the NEB method
78 ([Åsgeirsson et al., 2021](#)).

80 Other standalone conformational search software is instead interfaced via subprocess calls
81 and file-based I/O, like CREST ([Pracht et al., 2020, 2024](#)), ORCA's GOAT ([Souza, 2025](#)) and
82 racerts ([GitHub - Digital-Chemistry-Laboratory/Racerts](#); [Schmid et al., 2026](#)).

83 The modular nature of the calculators and interfaces allows the definition of complex workflows
84 in plain text files (input.txt):

¹That is, a calculator providing the energy and gradient differences between an atomic structure in an implicit solvent and the same structure in vacuum.

```
firecode input.txt -n test_run
```

85 A second standalone executable exposes the core optimizer routines acting directly on structure
86 files:

```
firecode_opt conformers.xyz --opt --freq --solvent toluene [...]
```

87 Contributions by way of adding new routines (or interfaces) to the codebase is made
88 straightforward by design: individual “operators” (routines) are simply functions reading a
89 molecular .xyz file and returning the filename of the processed ensemble. Run information
90 can be accessed from the Embedder class or via environment variables.

```
def center_operator(filename: str, embedder: Embedder) -> str:
    """Example operator centering the molecule."""

    # the Embedder class stores global information
    embedder.avail_gpus # 1
    embedder.options.solvent # "ch2cl2"
    embedder.options.T # 298.15

    # get a copy of the molecule of interest
    mol = embedder.mols[filename]

    # center coordinates
    mol.coords[0] -= np.mean(mol.coords[0], axis=0)

    # save any data you might need later
    embedder.options.centered = True
    embedder.options.last_operator = "center"

    # write to global log
    embedder.log(f"--> Center operator: centered molecule {mol.basename}")

    # write outfile and return its name
    outfile = f"{mol.basename}_centered.xyz"
    mol.to_xyz(outfile)

    return outfile
```

91 Research impact statement

92 Since its earliest versions in 2021 (formerly [TSCoDe](#)), this software proved essential in
93 orchestrating asymmetric catalysis workflows with ensembles up to thousands of structures
94 ([Huth et al., 2024](#); [Marvich et al., 2026](#); [Rozema et al., 2025](#); [Tampellini et al., 2024, 2025](#);
95 [Tampellini, Choi, et al., 2025](#)). Adoption is slowly starting to grow with pypi reporting
96 ~150 downloads per month at the present time. Part of the core code was exported to
97 the prism_pruner library ([GitHub - Ntampellini/Prism_pruner](#)) to facilitate its adoption by
98 the computational chemistry startup Rowan ([Rowan | ML-Powered Molecular Design and](#)
99 [Simulation — Rowansci.com](#)) (not affiliated with the author).

100 AI usage disclosure

101 Generative AI tools (Claude, Gemini) were used in some cases to generate first drafts of Python
102 functions that were then manually curated line-by-line and interfaced with the rest of the
103 codebase by hand. No generative AI tools were used in the writing of this manuscript.

Acknowledgements

We acknowledge years of research support from both Alma Mater Studiorum - University of Bologna (Prof. Giorgio Bencivenni and Prof. Paolo Righi) and Yale University (Prof. Scott J. Miller). These environments provided both the computational resources to run and develop the software as well as the primary problems it was designed to address and was stress-tested against.

References

- Alegre-Requena, J. V., Sowndarya S. V, S., Pérez-Soto, R., Alturaifi, T. M., & Paton, R. S. (2023). AQME: Automated quantum mechanical environments for researchers and educators [Journal Article]. *WIREs Computational Molecular Science*, 13(5), e1663. <https://doi.org/10.1002/wcms.1663>
- Anstine, R. I., Dylan; Zubatyuk. (2024). AIMNet2: A neural network potential to meet your neutral, charged, organic, and elemental-organic needs. This content is a preprint and has not been peer-reviewed. [Journal Article]. *ChemRXiv*. <https://doi.org/10.26434/chemrxiv-2023-296ch-v2>
- Ase / ase · GitLab — [gitlab.com](https://gitlab.com/ase/ase). <https://gitlab.com/ase/ase>.
- Ásgeirsson, V., Birgisson, B. O., Bjornsson, R., Becker, U., Neese, F., Riplinger, C., & Jónsson, H. (2021). Nudged elastic band method for molecular reactions using energy-weighted springs combined with eigenvector following [Journal Article]. *Journal of Chemical Theory and Computation*, 17(8), 4929–4945. <https://doi.org/10.1021/acs.jctc.1c00462>
- Bannwarth, C., Ehlert, S., & Grimme, S. (2019). GFN2-xTB—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions [Journal Article]. *Journal of Chemical Theory and Computation*, 15(3), 1652–1671. <https://doi.org/10.1021/acs.jctc.8b01176>
- Faccts. Weasel - FACCTs — [faccts.de](https://www.faccts.de/weasel/). <https://www.faccts.de/weasel/>.
- GitHub - berquist/cuby: A copy of cuby, a computational chemistry framework written in Ruby. — [github.com](https://github.com/berquist/cuby). <https://github.com/berquist/cuby>.
- GitHub - digital-chemistry-laboratory/racerts: A python package for efficient conformer sampling for transition states — [github.com](https://github.com/digital-chemistry-laboratory/racerts). <https://github.com/digital-chemistry-laboratory/racerts>.
- GitHub - networkx/networkx: Network Analysis in Python — [github.com](https://github.com/networkx/networkx). <https://github.com/networkx/networkx>.
- GitHub - ntampellini/prism_pruner: Similarity pruning engine for conformational ensembles — [github.com](https://github.com/ntampellini/prism_pruner). https://github.com/ntampellini/prism_pruner.
- GitHub - numpy/numpy: The fundamental package for scientific computing with Python. — [github.com](https://github.com/numpy/numpy). <https://github.com/numpy/numpy>.
- GitHub - pytorch/pytorch: Tensors and Dynamic neural networks in Python with strong GPU acceleration — [github.com](https://github.com/pytorch/pytorch). <https://github.com/pytorch/pytorch>.
- GitHub - rdkit/rdkit: The official sources for the RDKit library — [github.com](https://github.com/rdkit/rdkit). <https://github.com/rdkit/rdkit>.
- GitHub - tblite/tblite: Light-weight tight-binding framework — [github.com](https://github.com/tblite/tblite). <https://github.com/tblite/tblite>.
- GitHub - thegomeslab/ML-FSM: Public FSM implementation with support for ML-based

- potential energy surfaces — github.com. <https://github.com/thegomeslab/%7BM%7D%7BL%7D-%7BF%7D%7BS%7D%7BM%7D>.
- GitHub - zadorlab/sella: A Python software package for saddle point optimization and minimization of atomic systems. — github.com. <https://github.com/zadorlab/sella>.
- Grimme, S. (2012). Supramolecular binding thermodynamics by dispersion-corrected density functional theory [Journal Article]. *Chemistry – A European Journal*, 18(32), 9955–9964. <https://doi.org/10.1002/chem.201200497>
- Huth, S. E., Tampellini, N., Guerrero, M. D., & Miller, S. J. (2024). Catalytic enantioselective sulfoxidation of functionalized thioethers mediated by aspartic acid-containing peptides [Journal Article]. *Organic Letters*, 26(32), 6872–6877. <https://doi.org/10.1021/acs.orglett.4c02452>
- Marks, J., & Gomes, J. (2025). Incorporation of internal coordinates interpolation into the freezing string method [Journal Article]. *Journal of Chemical Theory and Computation*, 21(23), 12110–12120. <https://doi.org/10.1021/acs.jctc.5c01492>
- Marvich, H., Langner, O. C., Tampellini, N., & Miller, S. J. (2026). A bifunctional aminoxyl–bipyridine peptide catalyst for the atroposelective copper-catalyzed aerobic oxidation of biaryl diols [Journal Article]. *Submitted*.
- Neese, F. (2022). Software update: The ORCA program system—version 5.0 [Journal Article]. *WIREs Computational Molecular Science*, 12(5), e1606. <https://doi.org/10.1002/wcms.1606>
- Neese, F., Wennmohs, F., Becker, U., & Riplinger, C. (2020). The ORCA quantum chemistry program package [Journal Article]. *The Journal of Chemical Physics*, 152(22), 224108. <https://doi.org/10.1063/5.0004608>
- Pracht, P., Bohle, F., & Grimme, S. (2020). Automated exploration of the low-energy chemical space with fast quantum chemical methods [Journal Article]. *Physical Chemistry Chemical Physics*, 22(14), 7169–7192. <https://doi.org/10.1039/C9CP06869D>
- Pracht, P., Grimme, S., Bannwarth, C., Bohle, F., Ehlert, S., Feldmann, G., Gorges, J., Müller, M., Neudecker, T., Plett, C., Spicher, S., Steinbach, P., Wesołowski, P. A., & Zeller, F. (2024). CREST—a program for the exploration of low-energy molecular chemical space [Journal Article]. *The Journal of Chemical Physics*, 160(11), 114110. <https://doi.org/10.1063/5.0197592>
- Řezáč, J. (2016). Cuby: An integrative framework for computational chemistry [Journal Article]. *Journal of Computational Chemistry*, 37(13), 1230–1237. <https://doi.org/10.1002/jcc.24312>
- Riniker, S., & Landrum, G. A. (2015). Better informed distance geometry: Using what we know to improve conformation generation [Journal Article]. *Journal of Chemical Information and Modeling*, 55(12), 2562–2574. <https://doi.org/10.1021/acs.jcim.5b00654>
- Rowan | ML-Powered Molecular Design and Simulation — rowansci.com. <https://rowansci.com/>.
- Rozema, S. D., Tampellini, N., Rein, J., Sigman, M. S., Lin, S., & Miller, S. J. (2025). Experimental lineage and computational analysis of a general aminoxyl-based oxidation catalyst: Generality from substrate-specific interactions [Journal Article]. *ACS Catalysis*, 15(20), 17548–17557. <https://doi.org/10.1021/acscatal.5c05893>
- Schmid, S. P., Seng, H., Kläy, T., & Jorner, K. (2026). Rapid generation of transition-state conformer ensembles via constrained distance geometry [Journal Article]. *Journal of Chemical Information and Modeling*, 66(5), 2777–2790. <https://doi.org/10.1021/acs.jcim.5c02794>

- 195 Souza, B. de. (2025). GOAT: A global optimization algorithm for molecules and atomic
196 clusters [Journal Article]. *Angewandte Chemie International Edition*, 64(18), e202500393.
197 <https://doi.org/10.1002/anie.202500393>
- 198 Tampellini, N., Choi, E. S., & Miller, S. J. (2025). Peptide-catalyzed asymmetric amination of
199 sulfenamides enabled by DFT-guided catalyst optimization [Journal Article]. *Journal of the*
200 *American Chemical Society*, 147(44), 41122–41129. <https://doi.org/10.1021/jacs.5c15618>
- 201 Tampellini, N., Mercado, B. Q., & Miller, S. J. (2024). Scaffold-oriented asymmetric catalysis:
202 Conformational modulation of transition state multivalency during a catalyst-controlled
203 assembly of a pharmaceutically relevant atropisomer [Journal Article]. *Chemistry – A*
204 *European Journal*, 30(30), e202401109. <https://doi.org/10.1002/chem.202401109>
- 205 Tampellini, N., Mercado, B. Q., & Miller, S. J. (2025). Enantiocontrolled cyclization
206 to form chiral 7- and 8-membered rings unified by the same catalyst operating with
207 different mechanisms [Journal Article]. *Journal of the American Chemical Society*, 147(5),
208 4624–4630. <https://doi.org/10.1021/jacs.4c17080>
- 209 Wood, B. M., Dzamba, M., Fu, X., Gao, M., Shuaibi, M., Barroso-Luque, L., Abdelmaqsoud,
210 K., Gharakhanyan, V., Kitchin, J. R., & Levine, D. S. (2025). UMA: A family of universal
211 models for atoms [Journal Article]. *arXiv Preprint*. <https://arxiv.org/abs/2506.23971>

DRAFT