The Journal of Open Source Software

SPARC-X-API: Versatile Python Interface for Real-space Density Functional Theory Calculations

Tian Tian [1,3¶], **Lucas R Timmerman** [1], **Shashikant Kumar** [1], **Ben Comer** [1], **Andrew J Medford** [1¶], **and Phanish Suryanarayana** [1,2¶]

1 College of Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA **2** College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA **3** Department of Chemical and Materials Engineering, University of Alberta, Edmonton AB, T6G 2R3, Canada ¶ Corresponding author

DOI: 10.21105/joss.07747

Software

- Review
- Repository
- Archive

Editor: Bonan Zhu

Reviewers:

- @jacksund
- @vijaymocherla
- @naik-aakash

Submitted: 23 November 2024
Published: 01 June 2025

Summary

Density Functional Theory (DFT) is the de facto workhorse for large-scale electronic structure calculations in chemistry and materials science. While plane-wave DFT implementations remain the most widely used, real-space DFT provides advantages in handling complex boundary conditions and scaling to very large systems by allowing for the efficient use of large-scale supercomputers and linear-scaling methods that circumvent the cubic scaling bottleneck. The SPARC-X project (https://github.com/SPARC-X) provides highly efficient and portable real-space DFT codes for a wide range of first principle applications, available in both Matlab (M-SPARC (Xu et al., 2020; Zhang et al., 2023)) and C/C++ (SPARC (Xu et al., 2021; Zhang et al., 2024)). The rapid growth of SPARC's feature set has created the need for a fully functional interface to drive SPARC in high-throughput calculations. Here we introduce SPARC-X-API, a Python package designed to bridge the SPARC-X project with broader computational frameworks. Built on the Atomic Simulation Environment (ASE (Hjorth Larsen et al., 2017)) standard, the SPARC-X-API allows users to handle SPARC file formats and run SPARC calculations through the same interface as with other ASE-compatible DFT packages. Beyond standard ASE capabilities, SPARC-X-API provides additional features including 1) support of SPARC-specific setups, including complex boundary conditions and unit conversion, 2) a JSON schema parsed from SPARC's documentation for parameter validation and compatibility checks, and 3) a comprehensive socket communication layer derived from the i-PI protocol (Ceriotti et al., 2014; Kapil et al., 2019) facilitating message passing between low-level C code and the Python interface. The goal of the SPARC-X-API is to provide an easy-to-use interface for users with diverse needs and levels of expertise, allowing for minimal effort in adapting SPARC to existing computational workflows, while also supporting developers of advanced real-space methods.

Statement of Need

DFT has unarguably become one of the cornerstones of electronic structure simulations in chemical and materials sciences due to its simplicity and wide range of applicability. Among the various numerical implementations of DFT, the plane-wave pseudopotential method has gained significant popularity, owing to both its robustness and the maturity of associated software packages. However, despite their widespread use, plane-wave methods face several long-standing challenges, mostly related with the reliance of Fourier transformation to switch between reciprocal and real-space representations, including 1) establishing efficient schemes on massively parallel computing environments, 2) overcoming the extensive global communication during Fourier transformation calculations on very large systems, 3) developing linear scaling routines (Bowler & Miyazaki, 2012) and 4) handle non-periodic boundary conditions for isolated

and semi-finite systems. A compelling alternative to overcome these limitations is to solve the Kohn-Sham equations using a finite-difference (FD) approach on real-space grids. The locality of the FD method makes real-space DFT methods inherently scalable and paves the way for the development of linearly-scaling solutions to the Kohn-Sham equations. Real-space DFT also naturally supports periodic and Dirichlet boundary conditions, and combinations thereof, allowing for the flexible treatment of systems in any dimensionality.

In the past few years, the SPARC-X project (https://github.com/SPARC-X) has led efforts to develop an open-source, real-space DFT code that is both user-friendly and competitive with state-of-the-art plane-wave codes. The philosophy of the SPARC-X project is to provide codes that are highly efficient and portable (i.e., straightforward to install and use across various computational environments). The codes also seek to be both user- and developer-friendly to facilitate the implementation of new algorithms. In line with this, SPARC-X offers real-space DFT algorithms through two implementations: 1) Matlab-based M-SPARC (Xu et al., 2020; Zhang et al., 2023) for algorithm prototyping and small-system simulations, with no external dependencies other than Matlab itself, and 2) C/C++ based SPARC (Xu et al., 2021; Zhang et al., 2024) for large-scale production calculations that can accommodate a wide range of system sizes and requires only MPI and MKL/BLAS for compilation. New features of SPARC include spin-orbit coupling, dispersion interactions, and advanced exchange-correlation (xc) functionals (Zhang et al., 2024), linear-scaling Spectral Quadrature (SQ) method (Suryanarayana et al., 2018), cyclic/helical symmetry (Sharma & Suryanarayana, 2021), real-space density functional perturbation theory (DFPT) (Sharma & Suryanarayana, 2023), orbital-free DFT (ODFT) (Ghosh & Suryanarayana, 2016), and on-the-fly machine-learning force fields (OTF-MLFF) (Kumar et al., 2023, 2024; Timmerman et al., 2024).

The rapid development of SPARC has naturally created a demand for a fully functional and user-friendly interface that integrates SPARC smoothly into high-throughput workflows, which also completes the SPARC-X toolkit and complements its philosophy of usability and portability. To address this, we introduce the SPARC-X-API, a Python interface designed to bridge the SPARC code with a wide range of scientific workflows. The SPARC-X-API builds upon the Python wrapper originally shipped with SPARC version 1.0 (Xu et al., 2021), offering an API compatible with the widely-used ASE (ASE (Hjorth Larsen et al., 2017)) standard and updated with the latest versions of SPARC. With ASE's support for various popular DFT methods, including both plane-wave (e.g. VASP (Kresse & Furthmüller, 1996), Quantum ESPRESSO (Giannozzi et al., 2017), and Abinit (Gonze et al., 2020)), and real-space (e.g. GPAW (Enkovaara et al., 2011; Mortensen et al., 2024) and Octopus (Tancogne-Dejean et al., 2020)) implementations, SPARC-X-API enables seamless integration of SPARC into existing workflows, allowing users to incorporate real-space DFT calculations with minimal adjustments. The modular design of SPARC-X-API makes it straightforward to be plugged into complex computational workflows, for example high-throughput dynamics simulations by i-PI (Litman et al., 2024) and PLUMED (Bonomi et al., 2019), as well as active machine learning frameworks including FineTuna (Musielewicz et al., 2022), powered by state-of-art neural network interatomic potentials such as FAIR-Chem (https://github.com/FAIR-Chem/fairchem) and MACE-MP (Batatia et al., 2024) model series. A summary of the role SPARC-X-API in the SPARC-X project is shown in Figure 1. In addition to the capabilities inherited from ASE, SPARC-X-API seeks to enhance the user experience in a few key aspects, including 1) supporting SPARC-specific features in an ASE-compatible API, 2) a parameter validation mechanism based on SPARC's LaTeX documentation, and 3) a versatile socket communication layer for efficient high-throughput calculations. Details will be discussed next.

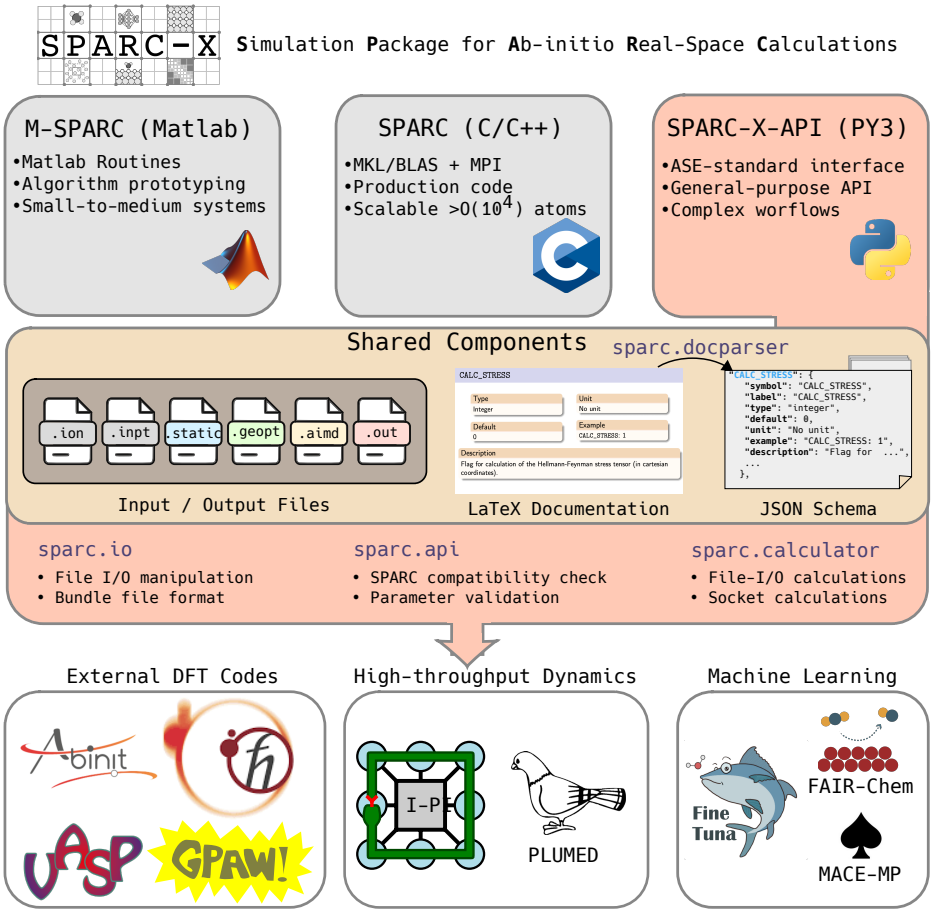

Figure 1: Overview of SPARC-X-API in the SPARC-X project system

Features and Functionalities

The SPARC-X-API is structured as a Python package, `sparc`. A summary of its key functionalities is provided below; for current detailed documentation, please refer to the official documentation.

`sparc.io`: File I/O Manipulation

In SPARC and M-SPARC calculations, input information is provided by two files: a `.inpt` (cell dimensions, boundary conditions, calculation flags), and a `.ion` file (atomic configurations and locations to pseudopotential). Depending on the type of calculation, various output files may be written, such as`.static`, `.geopt` or `.aimd`. The separation of information across multiple files means converting ASE `Atoms` objects to SPARC input files or retrieving energy and forces information from SPARC calculations requires handling more than just a single file, as is common in most ASE I/O formats. To manage this, the SPARC-X-API operates on the directory level, treating each calculation directory as a "SPARC bundle". The `sparc.io.SparcBundle` class facilitates reading from and writing to this bundle, ensuring that all necessary input and output files are properly handled. By default, the SPARC-X-API also copies relevant pseudopotential files into the calculation directory, making the SPARC bundle portable across different machines. From version 1.0.7 onwards, the SPARC-X-API leverages the new features introduced in ASE version 3.23 to register as an external I/O format, allowing reading and writing SPARC files directly using `ase.io` submodule:

```python
from ase.io import read, write
# 1. Read a SPARC bundle by specifying the `sparc` format
atoms = read("sparc_output_dir", format="sparc")
# 2. Write to a SPARC bundle from aboth object
write("sparc_input_dir", atoms, format="sparc")
```

The SPARC-X-API also supports parsing complex boundary conditions from the `.inpt` file. The periodic (P) and Dirichlet (D) boundary conditions are translated into `True` and `False` values, respectively, in the corresponding `pbc` direction of an `Atoms` object. Standard ASE objects do not natively support cyclic (C) or helical (H) boundary conditions that are available in SPARC, so the SPARC-X-API treats them similarly to Dirichlet boundaries and stores the original boundary condition information in the `info` attribute of the atomic object. This ensures that the correct boundary combinations are preserved when re-writing to SPARC input files.

`sparc.api`: Parameter Validation

In the ASE ecosystem, default calculator interfaces such as `FileIOCalculator` do not implement parameter validation, which can lead to issues such as incorrect parameter settings or incompatibility when running calculations through ASE. To address this, the SPARC-X-API introduces a robust parameter validation system using a JSON schema generated from SPARC's LaTeX documentation. A JSON schema contains the version of the SPARC software, a list of input parameters used in `.inpt` and `.ion` files, as well as supported data types and parameter categories. Validation is handled via the `sparc.api.SparcAPI` class, and includes:

- Verify that the schema is compatible with the version of SPARC binary.
- Convert `.inpt` fields into Python data types.
- Validate input parameters in both string and numerical formats.
- Output help information about specific parameter(s).

Each release of the SPARC-X-API contains a copy of a JSON schema linked with the latest SPARC release as the default validator, although the users can select different combination of SPARC versions and schemas depending on the version they are using. The separation between the SPARC-X-API and the core SPARC code not only prevents the need for hard-coding parameter lists into the API, but also facilitates easier maintenance: the "central truth" of parameters remains in the SPARC documentation, maintained by the SPARC core developers, while the SPARC-X-API focuses on providing a user-friendly interface without being tied to constant updates. This approach maximizes flexibility and avoids version conflicts between the API and the underlying code.

`sparc.calculator`: Socket-Communication Calculator Interface

The submodule `sparc.calculator` provides a class `SPARC` as the main entry point for driving SPARC calculations. This class provides two modes of operation: 1) a file I/O-based calculator extending the `ase.calculators.FileIOCalculator` class, and 2) a comprehensive socket communication layer that allows direct communication between the Python API and low-level C/C++ code.

In file I/O mode, the SPARC calculator object utilizes the `sparc.io.SparcBundle` for generating input files and `sparc.api.SparcAPI` for parameter validation, while the mode of calculation (single-point, relaxation or molecular dynamics) is controlled by the input flags. For users transitioning from other DFT packages and their ASE calculators, the SPARC-X-API is designed to minimize adaptation effort, but the API is designed to also enable advanced inputs from expert users. The `SPARC` calculator class achieves this by supporting two sets of input parameters: 1) lower-case special parameters that follow conventions from other ASE DFT calculators (e.g. real-space grid spacing `h` from GPAW, and exchange-correlation keyword `xc` from VASP) that use the ASE default Angstrom-eV system, and 2) case-insensitive raw SPARC input parameters in Bohr-Hartree units for fine-grained control. This dual approach is

designed so that users familiar with other DFT codes can adopt SPARC with minimal changes to their existing workflows, while expert users can exert full control. Basic DFT calculations can be covered by using standard ASE parameter sets in the SPARC-X-API, as shown by the side-by-side constructor with VASP and GPAW, using the same exchange-correlation functional and compatible convergence settings:

```
#1. Using VASP
from ase.calculators.vasp import Vasp
calc = Vasp(xc="pbe", kpts=(9, 9, 9), ecut=450, ediff=1.e-4)

#2. Using GPAW
from gpaw import GPAW
calc = GPAW(xc="pbe", kpts=(9, 9, 9), h=0.25, convergence={"energy": 1.e-4})

#3. Using SPARC
from sparc.calculator import SPARC
calc = SPARC(xc="pbe", kpts=(9, 9, 9), h=0.25, convergence={"energy": 1.e-4})
```

In high-throughput frameworks requiring thousands of single-point DFT evaluations, relying on file I/O mode can be inefficient, as calculations are restarted at each DFT call and the total number of files may exceed SPARC's default file count limit. The socket layer in the SPARC-X-API avoids these limitations by directly communicating with a long-running SPARC process for updating atomic positions, while keeping density and orbitals in memory and reducing self-consistent field (SCF) cycles. While alternative communication methods exist, such as C-binding approaches seen in GPAW (Mortensen et al., 2024) and Psi4 (Smith et al., 2020), these typically involve complex compilation and integration steps when installing the Python package. We chose a socket-based communication layer for its simplicity, which allows for a clear separation between the Python and SPARC codebases, minimal modifications to the existing C/C++ code, and ease of installation without requiring recompilation.

The communication protocol used in the SPARC-X-API socket, referred to as the SPARC protocol, is based on the i-PI protocol (Ceriotti et al., 2014; Kapil et al., 2019), which is also adopted by a wide range of ASE calculators. The SPARC protocol introduces additional header types and supports binary data transfers via Python's pickle format. While SPARC's C/C++ code maintains compatibility with the original i-PI standard, the SPARC-X-API leverages the extended protocol with pickle decoding. The two-tier design offers flexibility for socket calculations. At its core, the SPARC binary can communicate directly with any i-PI-compatible server, such as `ase.calculators.socketio.SocketIOCalculator` in ASE, using the basic protocol, though this requires careful setup by the user. However, the SPARC-X-API leverages the SPARC protocol, which allows the API to internally relay more advanced data types to the SPARC binary, handling object decoding and socket resets automatically. When running socket calculations on a single machine, users can activate socket mode by simply adding `use_socket=True` to the `SPARC` calculator constructor, enabling UNIX socket communication without additional setup. More importantly, the design of the SPARC protocol allows easy and seamless integration in distributed computational systems, offering the following features: 1) flexible client initialization / restart 2) efficient data transfer 3) heterogeneous computational setups. The design of the SPARC protocol allows insertion of bidirectional additional routines between two DFT calls, allowing further control over the low-level C/C++ code. Figure 2 summarizes the server-client setup across hybrid computing platforms.

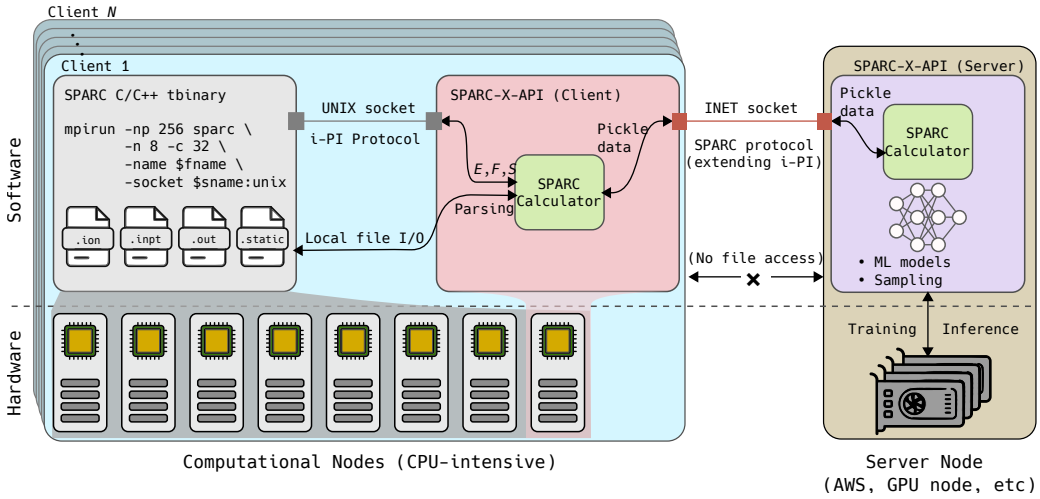

Figure 2: Example of socket communication across hybrid computing platforms using SPARC-X-API

Miscellaneous Helper Functionalities

The SPARC-X-API also provides several helper functions to facilitate user installation and testing, including:

- `sparc.quicktest`: a utility to verify the installation and environment setups for SPARC-X-API and SPARC.
- `sparc.docparser`: a submodule to convert existing LaTeX documentation included in SPARC source code into JSON schema.
- `sparc.download_data`: a tool to download the latest ONCV pseudopotentials distributed by SPARC.
- `sparc-ase`: an extension to the commandline ase tool, adding compatibility with SPARC file formats.

Code Release and Maintenance

The SPARC-X-API is released as source code in github repository https://github.com/SPARC-X/SPARC-X-API, and as a `conda-forge` package `sparc-x-api`. When installed using `conda-forge`, the package is bundled with the optimized SPMS pseudopotentials (Shojaei et al., 2023), and compatible with the `sparc` package that contains the compiled SPARC binary.

It also integrates continuous integration (CI) workflows for:

- Unit testing and code coverage
- Fetching the latest SPARC documentation for updating the JSON schema
- Validating all test examples from the SPARC repository

These workflows ensure that SPARC-X-API remains up-to-date with ongoing SPARC developments while separating parameter updates from the main SPARC maintainers' efforts.

Acknowledgements

The authors gratefully acknowledge the support of the U.S. Department of Energy, Office of Science, under Grant No. DE-SC0019410 and DE-SC0023445.

References

Batatia, I., Benner, P., Chiang, Y., Elena, A. M., Kovács, D. P., Riebesell, J., Advincula, X. R., Asta, M., Avaylon, M., Baldwin, W. J., Berger, F., Bernstein, N., Bhowmik, A., Blau, S. M., Cărare, V., Darby, J. P., De, S., Della Pia, F., Deringer, V. L., … Csányi, G. (2024). *A foundation model for atomistic materials chemistry*. arXiv. https://doi.org/10.48550/ARXIV.2401.00096

Bonomi, M., Bussi, G., Camilloni, C., Tribello, G. A., Banáš, P., Barducci, A., Bernetti, M., Bolhuis, P. G., Bottaro, S., Branduardi, D., Capelli, R., Carloni, P., Ceriotti, M., Cesari, A., Chen, H., Chen, W., Colizzi, F., De, S., De La Pierre, M., … The PLUMED consortium. (2019). Promoting transparency and reproducibility in enhanced molecular simulations. *Nature Methods*, *16*(8), 670–673. https://doi.org/10.1038/s41592-019-0506-8

Bowler, D. R., & Miyazaki, T. (2012). O(N) methods in electronic structure calculations. *Reports on Progress in Physics*, *75*(3), 036503. https://doi.org/10.1088/0034-4885/75/3/036503

Ceriotti, M., More, J., & Manolopoulos, D. E. (2014). I-PI: A python interface for ab initio path integral molecular dynamics simulations. *Computer Physics Communications*, *185*(3), 1019–1026. https://doi.org/10.1016/j.cpc.2013.10.027

Enkovaara, J., Romero, N. A., Shende, S., & Mortensen, J. J. (2011). GPAW - massively parallel electronic structure calculations with python-based software. *Procedia Computer Science*, *4*, 17–25. https://doi.org/10.1016/j.procs.2011.04.003

Ghosh, S., & Suryanarayana, P. (2016). Higher-order finite-difference formulation of periodic orbital-free density functional theory. *Journal of Computational Physics*, *307*, 634–652. https://doi.org/10.1016/j.jcp.2015.12.027

Giannozzi, P., Andreussi, O., Brumme, T., Bunau, O., Buongiorno Nardelli, M., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Cococcioni, M., Colonna, N., Carnimeo, I., Dal Corso, A., Gironcoli, S. de, Delugas, P., DiStasio, R. A., Ferretti, A., Floris, A., Fratesi, G., … Baroni, S. (2017). Advanced capabilities for materials modelling with Quantum ESPRESSO. *Journal of Physics: Condensed Matter*, *29*(46), 465901. https://doi.org/10.1088/1361-648x/aa8f79

Gonze, X., Amadon, B., Antonius, G., Arnardi, F., Baguet, L., Beuken, J.-M., Bieder, J., Bottin, F., Bouchet, J., Bousquet, E., Brouwer, N., Bruneval, F., Brunin, G., Cavignac, T., Charraud, J.-B., Chen, W., Côté, M., Cottenier, S., Denier, J., … Zwanziger, J. W. (2020). The Abinit project: Impact, environment and recent developments. *Computer Physics Communications*, *248*, 107042. https://doi.org/10.1016/j.cpc.2019.107042

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

Kapil, V., Rossi, M., Marsalek, O., Petraglia, R., Litman, Y., Spura, T., Cheng, B., Cuzzocrea, A., Meißner, R. H., Wilkins, D. M., Helfrecht, B. A., Juda, P., Bienvenue, S. P., Fang, W., Kessler, J., Poltavsky, I., Vandenbrande, S., Wieme, J., Corminboeuf, C., … Ceriotti, M. (2019). I-PI 2.0: A universal force engine for advanced molecular simulations. *Computer Physics Communications*, *236*, 214–223. https://doi.org/10.1016/j.cpc.2018.09.020

Kresse, G., & Furthmüller, J. (1996). Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, *6*(1), 15–50. https://doi.org/10.1016/0927-0256(96)00008-0

Kumar, S., Jing, X., Pask, J. E., Medford, A. J., & Suryanarayana, P. (2023). Kohn–Sham accuracy from orbital-free density functional theory via δ-machine learning. *The Journal of Chemical Physics*, *159*(24). https://doi.org/10.1063/5.0180541

Kumar, S., Pask, J. E., & Suryanarayana, P. (2024). Shock hugoniot calculations using on-the-fly machine learned force fields with ab initio accuracy. *Physics of Plasmas*, *31*(10). https://doi.org/10.1063/5.0230060

Litman, Y., Kapil, V., Feldman, Y. M. Y., Tisi, D., Begušić, T., Fidanyan, K., Fraux, G., Higer, J., Kellner, M., Li, T. E., Pós, E. S., Stocco, E., Trenins, G., Hirshberg, B., Rossi, M., & Ceriotti, M. (2024). I-PI 3.0: A flexible and efficient framework for advanced atomistic simulations. *The Journal of Chemical Physics*, *161*(6), 062504. https://doi.org/10.1063/5.0215869

Mortensen, J. J., Larsen, A. H., Kuisma, M., Ivanov, A. V., Taghizadeh, A., Peterson, A., Haldar, A., Dohn, A. O., Schäfer, C., Jónsson, E. Ö., Hermes, E. D., Nilsson, F. A., Kastlunger, G., Levi, G., Jónsson, H., Häkkinen, H., Fojt, J., Kangsabanik, J., Sødequist, J., … Thygesen, K. S. (2024). GPAW: An open python package for electronic structure calculations. *The Journal of Chemical Physics*, *160*(9). https://doi.org/10.1063/5.0182685

Musielewicz, J., Wang, X., Tian, T., & Ulissi, Z. (2022). FINETUNA: Fine-tuning accelerated molecular simulations. *Machine Learning: Science and Technology*, *3*(3), 03LT01. https://doi.org/10.1088/2632-2153/ac8fe0

Sharma, A., & Suryanarayana, P. (2021). Real-space density functional theory adapted to cyclic and helical symmetry: Application to torsional deformation of carbon nanotubes. *Physical Review B*, *103*(3). https://doi.org/10.1103/physrevb.103.035101

Sharma, A., & Suryanarayana, P. (2023). Calculation of phonons in real-space density functional theory. *Physical Review E*, *108*(4). https://doi.org/10.1103/physreve.108.045302

Shojaei, M. F., Pask, J. E., Medford, A. J., & Suryanarayana, P. (2023). Soft and transferable pseudopotentials from multi-objective optimization. *Computer Physics Communications*, *283*, 108594. https://doi.org/10.1016/j.cpc.2022.108594

Smith, D. G. A., Burns, L. A., Simmonett, A. C., Parrish, R. M., Schieber, M. C., Galvelis, R., Kraus, P., Kruse, H., Di Remigio, R., Alenaizan, A., James, A. M., Lehtola, S., Misiewicz, J. P., Scheurer, M., Shaw, R. A., Schriber, J. B., Xie, Y., Glick, Z. L., Sirianni, D. A., … Sherrill, C. D. (2020). PSI4 1.4: Open-source software for high-throughput quantum chemistry. *The Journal of Chemical Physics*, *152*(18). https://doi.org/10.1063/5.0006002

Suryanarayana, P., Pratapa, P. P., Sharma, A., & Pask, J. E. (2018). SQDFT: Spectral quadrature method for large-scale parallel O(N) Kohn–Sham calculations at high temperature. *Computer Physics Communications*, *224*, 288–298. https://doi.org/10.1016/j.cpc.2017.12.003

Tancogne-Dejean, N., Oliveira, M. J. T., Andrade, X., Appel, H., Borca, C. H., Le Breton, G., Buchholz, F., Castro, A., Corni, S., Correa, A. A., De Giovannini, U., Delgado, A., Eich, F. G., Flick, J., Gil, G., Gomez, A., Helbig, N., Hübener, H., Jestädt, R., … Rubio, A. (2020). Octopus, a computational framework for exploring light-driven phenomena and quantum dynamics in extended and finite systems. *The Journal of Chemical Physics*, *152*(12). https://doi.org/10.1063/1.5142502

Timmerman, L. R., Kumar, S., Suryanarayana, P., & Medford, A. J. (2024). Overcoming the chemical complexity bottleneck in on-the-fly machine learned molecular dynamics simulations. *Journal of Chemical Theory and Computation*, *20*(14), 5788–5795. https://doi.org/10.1021/acs.jctc.4c00474

Xu, Q., Sharma, A., Comer, B., Huang, H., Chow, E., Medford, A. J., Pask, J. E., & Suryanarayana, P. (2021). SPARC: Simulation package for ab-initio real-space calculations. *SoftwareX*, *15*, 100709. https://doi.org/10.1016/j.softx.2021.100709

Xu, Q., Sharma, A., & Suryanarayana, P. (2020). M-SPARC: Matlab-simulation package for ab-initio real-space calculations. *SoftwareX*, *11*, 100423. https://doi.org/10.1016/j.softx.2020.100423

Zhang, B., Jing, X., Kumar, S., & Suryanarayana, P. (2023). Version 2.0.0 - M-SPARC: Matlab-simulation package for ab-initio real-space calculations. *SoftwareX*, *21*, 101295. https://doi.org/10.1016/j.softx.2022.101295

Zhang, B., Jing, X., Xu, Q., Kumar, S., Sharma, A., Erlandson, L., Sahoo, S. J., Chow, E., Medford, A. J., Pask, J. E., & Suryanarayana, P. (2024). SPARC v2.0.0: Spin-orbit coupling, dispersion interactions, and advanced exchange–correlation functionals. *Software Impacts*, *20*, 100649. https://doi.org/10.1016/j.simpa.2024.100649