




# VibIR-Parallel-Compute: Enhancing Vibration and Infrared Analysis in High-Performance Computing Environments

Kurt Irvin M. Rojas <sup>1</sup>¶, Yoshitada Morikawa <sup>1,2</sup>, and Ikutaro Hamada <sup>1</sup>

<sup>1</sup> Department of Precision Engineering, Graduate School of Engineering, Osaka University, 2-1, Yamadaoka, Suita, Osaka, 565-0871, Japan <sup>2</sup> Research Center for Precision Engineering, Graduate School of Engineering, Osaka University, 2-1, Yamadaoka, Suita, Osaka, 565-0871, Japan ¶ Corresponding author

DOI: [10.21105/joss.07855](https://doi.org/10.21105/joss.07855)

## Software

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

Editor: [Lucy Whalley](#) 

## Reviewers:

- [@elindgren](#)
- [@mbagheri20](#)
- [@adamrhyshell](#)

Submitted: 02 November 2024

Published: 08 April 2025

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

In this work, we introduce VibIR-Parallel-Compute, a Python package designed to enhance both the usability and efficiency of vibrational and infrared (IR) analysis, particularly for high-performance computing environments. To address the computational bottleneck encountered in the vibrational spectroscopy modules in Atomic Simulation Environment (ASE) ([Hjorth Larsen et al., 2017](#)), we provide a *drop-in* replacement that parallelizes these bottleneck calculations in the current implementation. Additionally, we offer a compatibility layer for preparing and pre-processing the calculator data required for vibrational mode and IR intensity calculations, enabling the use of previously incompatible calculators. Furthermore, our package includes streamlined and efficient analysis modules that allow for quick visualization of vibrational modes, along with the plotting of vibrational density of states (VDOS) and IR spectra, all without the need to explore the underlying source code or manage excessive boilerplate.

Our primary objective is to develop an ASE extension or plugin that integrates seamlessly with existing workflows, enhancing computational capabilities while complementing current practices.

## Statement of need

Vibrational spectroscopy has become a valuable technique for analyzing structures of molecules and materials ([Kraka et al., 2020](#)). IR spectroscopy, in particular, is widely utilized to examine the inherent strength of chemical bonds and the characteristic bonding patterns present in materials. However, assigning vibrational modes to experimental spectra is challenging and typically requires comparison with model spectra predicted through electronic structure calculations, such as density functional theory (DFT).

A commonly used and efficient method for determining vibrational characteristics, such as modes and frequencies, involves solving for the eigenvector and eigenvalues of a Hessian matrix ([Giustino, 2014](#); [Wilson et al., 1980](#)). This matrix represents the strength of chemical bonds under the harmonic approximation. Each element in the Hessian matrix is defined as:

$$H_{ij} = -\frac{\partial F_j}{\partial u_i} \quad (1)$$

where the Hessian element  $H_{ij}$  reflects the negative gradient of the force  $F_j$  with respect to atomic position  $u_i$ . The  $i$  and  $j$  span the three spatial coordinates across  $N$  atoms.

The force gradient is determined using the finite-difference method, which involves applying small forward and backward displacements to atoms. This requires calculating the forces twice – once for the forward displacement and once for the backward displacement – both of which are independent computations. This independence is crucial as it enables parallel force evaluations for different displaced structure.

The construction and diagonalization of the Hessian matrix are implemented in the widely-used Python package, ASE. ASE offers various methods and workflows that interface with several *ab initio* calculators, such as VASP (Kresse & Furthmüller, 1996a, 1996b; Kresse & Hafner, 1993), Quantum Espresso (Paolo Giannozzi et al., 2009; P. Giannozzi et al., 2017) and GPAW (Mortensen et al., 2024). Spectroscopy-related modules are specifically available through the `ase.vibrations` module, providing tools for analyzing vibrational modes and frequencies. Alternative to ASE, several frameworks like phonopy (Togo et al., 2023; Togo, 2023) also enable phonon and lattice vibration calculations, which can be extended to simulated spectroscopy through dedicated tools such as Phonopy-Spectroscopy (Skelton et al., 2017) and THeSeuSS (Boziki et al., 2024). Additionally, high-throughput phonon and spectroscopic calculations can be streamlined using specialized workflow recipes provided by packages such as `aiida-vibroscopy` (Bastonero & Marzari, 2024), Atomic Simulation Recipes (Gjerding et al., 2021), and Atomate (Mathew et al., 2017). These tools are compatible with a wide range of DFT engines, offering flexibility for complex computational workflows.

In ASE's implementation, the total number force evaluations  $N_{\text{calc}}$  required by a system with  $N_{\text{atom}}$  atom is given by:

$$N_{\text{calc}} = 6 \cdot N_{\text{atom}} + 1 \quad (2)$$

In this equation, the factor of 6 accounts for the three spatial directions, considering both forward and backward displacements. The additional +1 term is for the equilibrium structure. This total number of calculations ensures compatibility with all three finite-difference methods: central, forward, and backward.

Although parallelizing the calculations for the displaced systems is straightforward, the current implementation runs in a serial loop with a time complexity of  $\mathcal{O}(N_{\text{calc}})$ . This limits the ability to take full advantage of the large computing resources typically available in high-performance computing (HPC) environments. While parts of the existing code could be adapted (along with new compatibility code) to implement parallelization, doing so would increase complexity and reduce the clarity of the script-level codes. Therefore, a more user-friendly, parallelized version of the `ase.vibrations` toolkit is desirable.

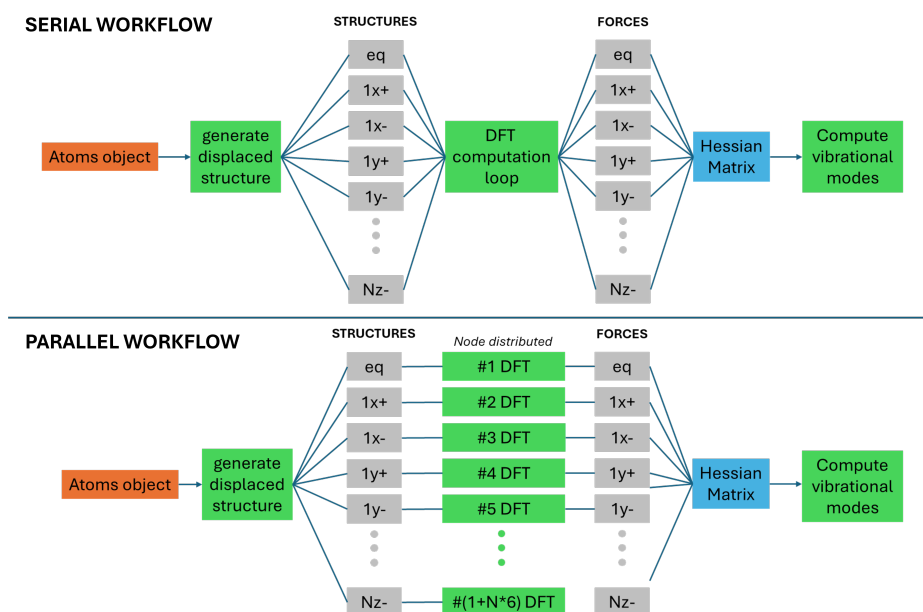
Another issue arises from the need for an additional calculation input and output processing layer. When generating IR spectra, we require the dipole moment changes of the system. In a three-dimensional system, such as an isolated molecule, this involves calculating a three-dimensional dipole moment vector. Unfortunately, some calculators – such as Quantum Espresso – only computes and outputs the dipole moment as a scalar along the chosen electric field direction (typically along the x, y, or z axis). This means that three separate calculations are necessary to obtain the full dipole moment vector. Currently, ASE does not support adding a processing layer capable of handling this very specialized but necessary routine.

## Features and Implementation

### Parallel ready code

In this work, we focused on enhancing the existing code to support parallelization by distributing each independent atomically displaced atomic system across separate computing resources. This approach, in theory, allows all systems to run simultaneously. We leveraged resource

workload managers commonly found in high-performance computing environments (e.g. PBS, SLURM, SGE) to efficiently assign each system to a distinct resource pool.



**Figure 1:** Workflow comparison. The serial and parallel workflow are compared, with the key difference being how resources are allocated for each displaced structure. In the structure/forces section, “eq” refers to the equilibrium structure, while “1x+” denotes atom 1 forward displaced along x-axis.

Figure 1 compares the workflows of the serial and parallel implementations. In the serial version, all structures are assigned to a single resource pool, while in the parallel version, each structure is distributed to a separate pool, enabling simultaneous calculations. The serial loop relies on the `run()` method, but we modified the code to support parallelization by adopting a module approach. We introduced a `run_tag(i)` method to run individual systems based on their index which can be executed using job arrays in resource workload managers. The following is a pseudo code for the difference between the serial and parallelized version:

```
# serial
vib = Vibrations(atoms)
vib.run()

# parallel - in serial loop
pvib = ParallelVibrations(atoms, calc_obj=EMT)
for i in range(1, vib.map_length + 1):
    pvib.run_tag(i)

# parallel - parallelized
pvib = ParallelVibrations(atoms, calc_obj=EMT)
pvib.run_tag(int(os.environ['PBS_ARRAY_INDEX']))
```

### Timings

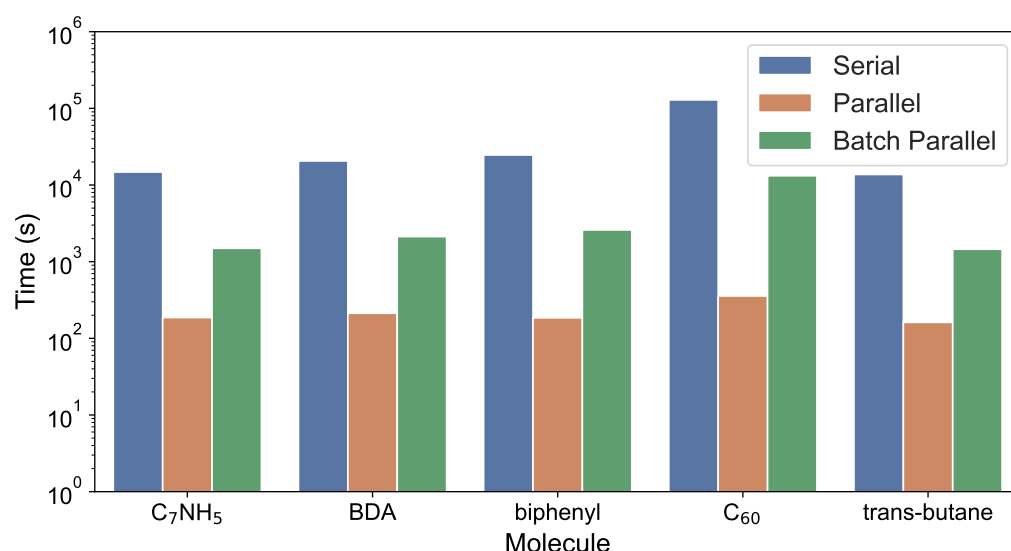
To study the impact of the parallelization scheme on the overall computation time, we estimated the computation time for a few molecules built from ASE's molecule module based on their energy and force evaluation time ( $T_{\text{eval}}$ ). We considered three implementation schemes, namely, serial, parallel, and batch parallel. The computation time for serial  $T_{\text{serial}}$ , parallel  $T_{\text{parallel}}$ , and batch parallel  $T_{\text{batch}}$  are given by:

$$T_{\text{serial}} = N_{\text{calc}} \cdot T_{\text{eval}} \quad (3)$$

$$T_{\text{parallel}} = T_{\text{eval}} \quad (4)$$

$$T_{\text{batch}} = \lceil N_{\text{calc}}/10 \rceil \cdot T_{\text{eval}} \quad (5)$$

The timing results in Figure 2 illustrate the distinct complexities associated with each implementation scheme. In the serial implementation, the time complexity is  $\mathcal{O}(N_{\text{calc}})$ . By contrast, in a fully parallelized version – where all needed resources are available – it reduces to  $\mathcal{O}(1)$ . However, in practice, this ideal scenario is rarely achieved due to HPC fair-use policies that limit the number of resources a user can access simultaneously. This limited-resource strategy can be referred to as *batch parallel*, where jobs are grouped into resource-constrained batches to meet fair-use requirements. The batch parallel scheme with a fair-use limit of 10 simultaneous jobs, leads to an effective time complexity of  $\mathcal{O}(\lceil N_{\text{calc}}/10 \rceil)$ .



**Figure 2:** Timing comparison between three implementation schemes. Herein, the total computation time for serial, parallel, and batch parallel are presented to demonstrate the effect of parallelization in reducing overall real-time spent. These timings are based on the energy and force evaluation time using the Quantum Espresso code with 32 cores (Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz)

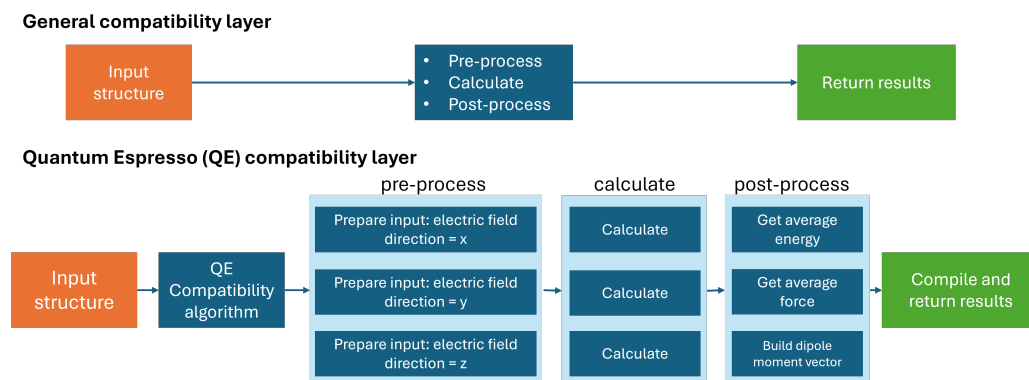
### Temporary files issue

One downside of this parallelized approach is the generation of numerous temporary files from the calculations, which can be quite large (e.g. charge density, wavefunction data). On larger systems, this could risk exhausting disk space. To address this, we implemented a file management system that allows deletion of temporary files after the calculations are complete. Additionally, since multiple calculations run in parallel, we created a system that ensures each calculation operates in a separate working directory, preventing any accidental sharing or overwriting of output and temporary files.

### Processing layer

In this additional processing layer, we designed the calculation to be flexible, allowing for more complex routines. Our focus was on the applicability of Quantum Espresso for infrared spectra calculations. As mentioned earlier, Quantum Espresso only calculates the dipole moment

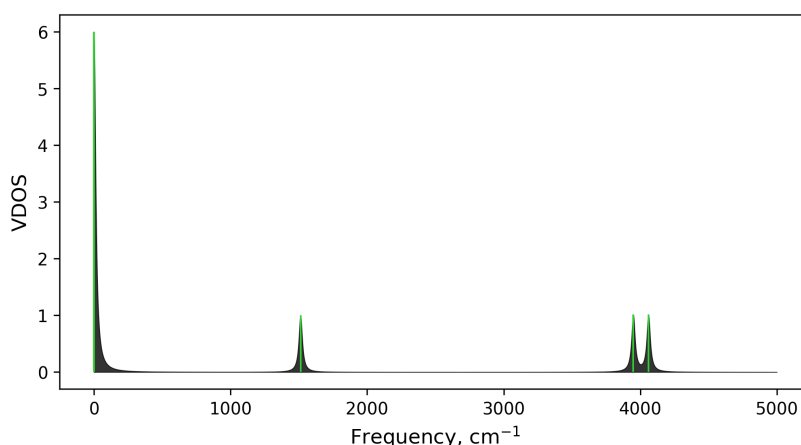
in a single direction. However, for molecular, ribbon, and sheet systems, we require three-dimensional, two-dimension, and one-dimensional dipole moment vectors, respectively. To handle this, we implemented a three-step calculation for each dipole direction and constructed the full dipole moment vector from these results. Additionally, for ribbon and sheet systems, we skip calculating the dipole moment along periodic directions to save time and resources. A schematic of the processing layer is shown in Figure 3.



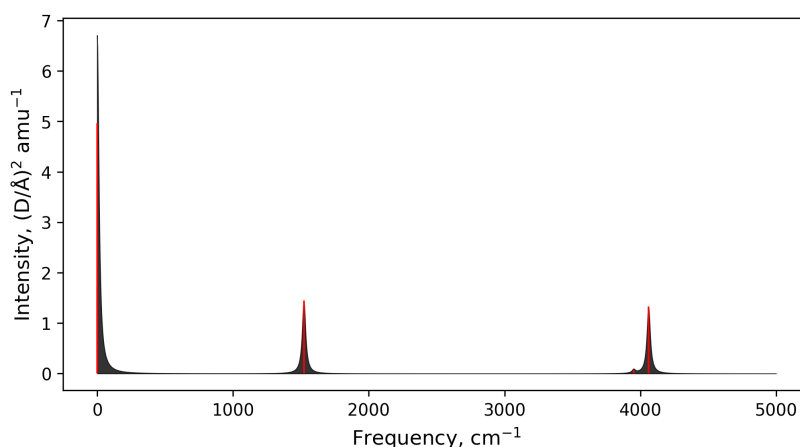
**Figure 3:** Compatibility layer. We present the diagram of the general compatibility layer and a specific example of the Quantum Espresso compatibility layer.

## Fast analysis

While ASE provides tools for analyzing vibrational modes and frequencies, we enhanced its user-friendliness. We introduced a quick-plot feature for vibrational modes and frequencies in both Vibrations (Figure 4) and Infrared (Figure 5) analyses. In this feature, we implemented a simplified export method that directly generates the VDOS and IR spectra plots. As shown in Figure 4 and Figure 5, the plots display two datasets: vertical colored lines indicate the precise, discrete frequency positions of each mode, and the black spectra represent the simulated spectra computed using a linear combination of Gaussian or Lorentzian functions from each mode. This streamlined method minimizes the amount of boilerplate code needed for visualization.

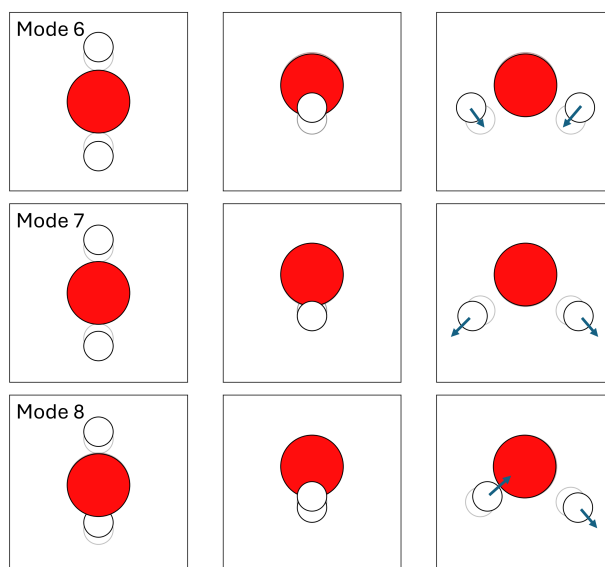


**Figure 4:** Vibrational density of states of water. The energy and force evaluation was performed by the Quantum Espresso code.



**Figure 5:** Simulated infrared spectra of water. The energy, force, and dipole moment evaluation was performed by the Quantum Espresso code. This infrared analyses also made use of the compatibility layer that allowed the use of the Quantum Espresso code for dipole moment calculations.

Additionally, we implemented a more detailed GIF animation of the vibrational modes to offer a clearer, more intuitive understanding. To improve efficiency, we applied CPU-based parallelization for GIF generation, significantly speeding up the process.



**Figure 6:** Static representation of mode GIFs. The image includes mode 6, 7 and 8 of water as an example. The energy and force evaluation was performed using the Quantum Espresso code. For better clarity in paper, arrows were added to indicate the mode direction.

## Usage

The VibIR-Parallel-Compute package is available on Gitlab <https://gitlab.com/morikawa-lab-osakau/vibir-parallel-compute> and documentations are provided in <https://morikawa-lab-osakau.gitlab.io/vibir-parallel-compute>.

## References

- Bastonero, L., & Marzari, N. (2024). Automated all-functionals infrared and Raman spectra. *Npj Computational Materials*, 10(1), 1–12. <https://doi.org/10.1038/s41524-024-01236-3>
- Boziki, A., Mebenga, F. N., Fernandes, P., & Tkatchenko, A. (2024). *A Journey with TheSeuSS: Automated Python Tool for Modeling IR and Raman Vibrational Spectra of Molecules and Solids* (No. arXiv:2409.06597). arXiv. <https://doi.org/10.48550/arXiv.2409.06597>
- 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., De Gironcoli, S., 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>
- Giannozzi, Paolo, Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G. L., Cococcioni, M., Dabo, I., Dal Corso, A., De Gironcoli, S., Fabris, S., Fratesi, G., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., ... Wentzcovitch, R. M. (2009). QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21(39), 395502. <https://doi.org/10.1088/0953-8984/21/39/395502>
- Giustino, F. (2014). *Materials modelling using density functional theory: Properties and predictions* (1st ed). Oxford University Press. ISBN: 978-0-19-966244-9
- Gjerding, M., Skovhus, T., Rasmussen, A., Bertoldo, F., Larsen, A. H., Mortensen, J. J., & Thygesen, K. S. (2021). Atomic Simulation Recipes: A Python framework and library for automated workflows. *Computational Materials Science*, 199, 110731. <https://doi.org/10.1016/j.commatsci.2021.110731>
- Hjorth Larsen, A., Jørgen Mortensen, J., Blomqvist, J., Castelli, I. E., Christensen, R., Duřak, 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>
- Kraka, E., Zou, W., & Tao, Y. (2020). Decoding chemical information from vibrational spectroscopy data: Local vibrational mode theory. *WIREs Computational Molecular Science*, 10(5), e1480. <https://doi.org/10.1002/wcms.1480>
- Kresse, G., & Furthmüller, J. (1996a). 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](https://doi.org/10.1016/0927-0256(96)00008-0)
- Kresse, G., & Furthmüller, J. (1996b). Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16), 11169–11186. <https://doi.org/10.1103/PhysRevB.54.11169>
- Kresse, G., & Hafner, J. (1993). *Ab Initio* molecular dynamics for liquid metals. *Physical Review B*, 47(1), 558–561. <https://doi.org/10.1103/PhysRevB.47.558>
- Mathew, K., Montoya, J. H., Faghaninia, A., Dwarakanath, S., Aykol, M., Tang, H., Chu, I., Smidt, T., Bocklund, B., Horton, M., Dagdelen, J., Wood, B., Liu, Z.-K., Neaton, J., Ong, S. P., Persson, K., & Jain, A. (2017). Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Computational Materials Science*, 139, 140–152. <https://doi.org/10.1016/j.commatsci.2017.07.030>

- 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), 092503. <https://doi.org/10.1063/5.0182685>
- Skelton, J. M., Burton, L. A., Jackson, A. J., Oba, F., Parker, S. C., & Walsh, A. (2017). Lattice dynamics of the tin sulphides SnS<sub>2</sub>, SnS and Sn<sub>2</sub>S<sub>3</sub>: Vibrational spectra and thermal transport. *Physical Chemistry Chemical Physics*, 19(19), 12452–12465. <https://doi.org/10.1039/C7CP01680H>
- Togo, A. (2023). First-principles Phonon Calculations with Phonopy and Phono3py. *Journal of the Physical Society of Japan*, 92(1), 012001. <https://doi.org/10.7566/JPSJ.92.012001>
- Togo, A., Chaput, L., Tadano, T., & Tanaka, I. (2023). Implementation strategies in phonopy and phono3py. *J. Phys. Condens. Matter*, 35(35), 353001. <https://doi.org/10.1088/1361-648X/acd831>
- Wilson, E. B., Decius, J. C., & Cross, P. C. (1980). *Molecular vibrations: The theory of infrared and raman vibrational spectra* (First published in 1980, is an unabridged and corrected republication of the work originally published in 1955 by McGraw-Hill Book Company, Inc). Dover Publications Inc. ISBN: 978-0-486-63941-3