The Journal of Open Source Software

pymatgen-analysis-defects: A Python package for analyzing point defects in crystalline materials

Jimmy-Xuan Shen [1] and **Joel Varley** [1]

1 Lawrence Livermore National Laboratory, Livermore, California 94550, United States

DOI: 10.21105/joss.05941

Software

- Review
- Repository
- Archive

Editor: Rachel Kurchin

Reviewers:

- @ireaml
- @atimmins7

Submitted: 02 October 2023
Published: 19 January 2024

Statement of need

Point defects can often determine the properties of semiconductor and optoelectronic materials. Due to the large simulation cell and the higher-cost density functionals required for defect simulations, the computational cost of defect calculations is often orders of magnitude higher than that of bulk calculations. As such, managing and curating the results of the defect calculations generated by a single user has the potential to save a significant amount of computational resources. Moreover, eventually building a high-quality, persistent defects database will significantly reduce the computational cost of defect calculations for the entire community.

Simulation of point defects is one of the most complex workflows in computational materials science, involving extensive pre- and post-processing of the structural and electronic structure data (Freysoldt et al., 2014). Multiple software packages exist to automate the simulation of point defects including work from Broberg et al. (2018), Kumagai et al. (2021), Huang et al. (2022), Arrigoni & Madsen (2021), Goyal et al. (2017), and Kavanagh et al. (2023); however, no available code focuses on:

1. Integration of but not insistence on standardized high-throughput workflow frameworks
2. Building large, persistent databases of point defects that are extensible to new calculations over time

Summary

Since the combinatorics of point defects in crystalline materials can be daunting, it is important to have a software package that can be easily integrated into high-throughput workflows to manage these complex calculations. However, most users of defect analysis packages will not need to run thousands of calculations, so it is important to have code focused purely on the defect analysis and relegate the high-throughput workflow aspect to a separate package. A focus of the present package is also to provide a base library for the analysis of point defects without invoking any high-throughput workflow frameworks. Even though this package was designed with high-throughput in mind and developed alongside a high-throughput workflow framework, it is not dependent on any particular workflow framework and can be used as a standalone analysis package.

Additionally, a well-known problem in the simulation of point defects is the fact that current structure optimization techniques can miss the ground state structure based on the initial guess in a sizable minority of cases, so the ability to easily re-visit and re-optimize structures is crucial to building a reliable database of point defects. Towards that end, we have developed a Python package, `pymatgen-analysis-defects`, and integrated it with the popular `atomate2` workflow framework to provide a complete set of tools for simulating, analyzing, and managing the results of point defect calculations.

Since the ability to revisit calculations is crucial to building a reliable database, but user tagging of calculations is inconsistent, especially in a high-throughput context, we have codified a structure-only definition of point defects that can be used to aggregate the results of multiple calculations of the same defect. This allows for the creation of a database of point defects that can be easily extended to new calculations over time. In addition to the focus on database building, we have also implemented several tools for analyzing carrier recombination in defects, these include:

1. Obtaining the chemical potential contribution to the defect formation energy without explicit calculations of the competing phases
2. Obtaining the Freysoldt finite-size correction without user intervention
3. Calculation of the optical transition between states under the independent-particle approximation
4. Calculation of the non-radiative recombination using the nonrad code (Turiansky et al., 2021)

Details of the implementation and tutorials for using the different parts of the package are provided at https://materialsproject.github.io/pymatgen-analysis-defects/intro.html.

Defect Definition

A core feature of pymatgen-analysis-defects is the ability to define point defects automatically. While symmetry analysis on the atomic structure alone is usually enough to define the distinct substitutional and vacancy defects, we found that the electronic charge density was the most effective at placing the interstitial defect at symmetry-inequivalent positions. A basic example of creating a full list of defects is shown below:

```python
from pymatgen.analysis.defects.generators import generate_all_native_defects
from pymatgen.ext.matproj import MPRester

with MPRester() as mpr:
    chgcar = mpr.get_charge_density_from_material_id("mp-804")

defects = []
for defect in generate_all_native_defects(chgcar):
    print(defect)
    defects.append(defect)
```

```
Ga Vacancy defect at site #0
N Vacancy defect at site #2
N subsitituted on the Ga site at at site #0
Ga subsitituted on the N site at at site #2
Ga intersitial site at [0.00,0.00,0.20]
Ga intersitial site at [0.35,0.65,0.69]
N intersitial site at [0.00,0.00,0.20]
N intersitial site at [0.35,0.65,0.69]
```

In the code above, we query the Materials Project database for the charge density object, which contains information about the bulk structure, as well as the electronic charge density. Using the generate_all_native_defects function, we can generate a list of all of the native point defects for this structure.

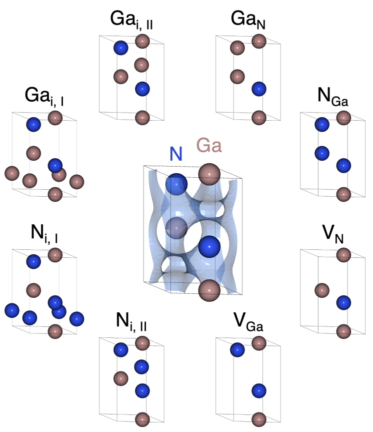

Figure 1: Defect generation.

Defect Simulation Workflow

A basic example of integration with the atomate2 workflow framework is provided below:

```
from atomate2.vasp.flows.defect import FormationEnergyMaker
from jobflow import Flow
from pymatgen.analysis.defects.generators import generate_all_native_defects
from pymatgen.ext.matproj import MPRester

with MPRester() as mpr:
    chgcar = mpr.get_charge_density_from_material_id("mp-804")


maker = FormationEnergyMaker()
jobs = []
for defect in generate_all_native_defects(chgcar):
    jobs.append(maker.make(defect))
flow = Flow(jobs)
```

The code above will generate a `Flow` object that contains all of the instructions to dynamically create all of the required defect calculations, which can be sent to the job manager on an HPC system.

Acknowledgements

The following parts of the present package contain contributions from other open-source projects: the finite size correction is based on the original implementation in pyCDT by Dr. Danny Broberg, the non-radiative recombination code is based on the original implementation in nonrad by Dr. Mark E. Turiansky. This work was performed under the auspices of the U.S. DOE by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344, and partially supported by LLNL LDRD funding under project number 22-SI-003.

Arrigoni, M., & Madsen, G. K. H. (2021). Spinney: Post-processing of first-principles calculations of point defects in semiconductors with Python. *Comput. Phys. Commun.*, *264*, 107946. https://doi.org/10.1016/j.cpc.2021.107946

Broberg, D., Medasani, B., Zimmermann, N. E. R., Yu, G., Canning, A., Haranczyk, M., Asta, M., & Hautier, G. (2018). PyCDT: A python toolkit for modeling point defects in semiconductors and insulators. *Comput. Phys. Commun.*, *226*, 165–179. https://doi.org/10.1016/j.cpc.2018.01.004

Freysoldt, C., Grabowski, B., Hickel, T., Neugebauer, J., Kresse, G., Janotti, A., & Walle, C. G. de. (2014). First-principles calculations for point defects in solids. *Rev. Mod. Phys.*, *86*, 253. https://doi.org/10.1103/RevModPhys.86.253

Goyal, A., Gorai, P., Peng, H., Lany, S., & Stevanović, V. (2017). A computational framework for automation of point defect calculations. *Comput. Mater. Sci.*, *130*, 1–9. https://doi.org/10.1016/j.commatsci.2016.12.040

Huang, M., Zheng, Z., Dai, Z., Guo, X., Wang, S., Jiang, L., Wei, J., & Chen, S. (2022). DASP: Defect and dopant ab-initio simulation package. *J. Semicond.*, *43*, 42101. https://doi.org/10.1088/1674-4926/43/4/042101

Kavanagh, S. R., Mosquera-Lois, I., Walsh, A., & Scanlon, D. O. (2023). doped. In *GitHub*. https://github.com/SMTG-Bham/doped

Kumagai, Y., Tsunoda, N., Takahashi, A., & Oba, F. (2021). Insights into oxygen vacancies from high-throughput first-principles calculations. *Phys. Rev. Mater.*, *5*, 123803. https://doi.org/10.1103/PhysRevMaterials.5.123803

Turiansky, M. E., Alkauskas, A., Engel, M., Kresse, G., Wickramaratne, D., Shen, J.-X., Dreyer, C. E., & Van de Walle, C. G. (2021). Nonrad: Computing nonradiative capture coefficients from first principles. *Comput. Phys. Commun.*, *267*, 108056. https://doi.org/10.1016/j.cpc.2021.108056