pyscal

Release 3.0.0.dev.6

Sarath Menon

Jun 16, 2023

CONTENTS

1 Installation **3**

2 Why version 3? **5**
- 2.1 Version 3 is much faster . . . 5
- 2.2 Version 3 uses less memory . . . 5
- 2.3 What are reasons for these benefits? . . . 5
- 2.4 What are the other feature updates? . . . 6

3 Examples **7**

4 Support, contributing and extending **9**
- 4.1 Reporting and fixing bugs . . . 9
- 4.2 New features . . . 9

5 Help and support **11**

6 Citing the code **13**

7 Acknowledgements **15**
- 7.1 Developer . . . 15
- 7.2 Contributers . . . 15
- 7.3 Acknowledgements . . . 15

8 License **17**
- 8.1 pyscal License . . . 17
- 8.2 Voro++ license . . . 17
- 8.3 pybind 11 license . . . 18
- 8.4 sphinx theme license . . . 18

pyscal is a python module for the calculation of local atomic structural environments including Steinhardt's bond orientational order parameters during post-processing of atomistic simulation data. The core functionality of pyscal is written in C++ with python wrappers using pybind11 which allows for fast calculations with possibilities for easy expansion in python.

Steinhardt's order parameters are widely used for identification of crystal structures. They are also used to identify if an atom is solid or liquid. pyscal is inspired by BondOrderAnalysis code, but has since incorporated many additions and modifications.

CHAPTER

ONE

INSTALLATION

pyscal can be installed through conda:

```
conda install -c conda-forge/label/pyscal_dev -c conda-forge pyscal
```

CHAPTER

TWO

WHY VERSION 3?

pyscal v3 is a new version with mostly updated codebase and breaking changes. Anybody who has working pyscal code will need to update it to get it working with this new version. Therefore, it is necessary to discuss why this new version was needed and the benefits of updating.

2.1 Version 3 is much faster

In the plot below, the time needed to calculate neighbors with the 'cutoff" method for systems with varying number of atoms with versions 2.10.15 and 3.0 is shown.

v3 is faster for all system sizes. At a system size of about 50,000 atoms, v3 is about 4x faster.

2.2 Version 3 uses less memory

A major issue with pyscal v2.x series was that it not useful for large system sizes due to the large amount of memory needed. In the plot below, the memory usage of both versions for the same calculation above is shown.

v3 uses less memory, for a system size of 50,000 atoms, v3 uses 14x less memory. A more interesting feature is the slope of the data, or how much the memory scales with the system size. For v3 it is only 0.008, while for v2 it is .12! For a system of 1 million atoms, v2 would use 117 GB of memory while v3 would need only 8 GB, making larger calculations accessible (these numbers will be updated after real use-case tests).

2.3 What are reasons for these benefits?

- The older C++ atoms class is deprecated. Instead, it is store as python dictionary. Therefore the copying between python and C++ sides is avoided.
- The atoms python dictionary is directly exposed to the C++ side. The dictionary is passed by reference, which allows in-place modification directly.

2.4 What are the other feature updates?

The new version includes a number of new features and quality of life improvements. Please check the examples for details.

CHAPTER

THREE

EXAMPLES

The gallery of examples below cover different ways in which calphy can be used to calculate free energies.

Getting started with pyscal

Creating structures

More Atoms methods

CHAPTER

FOUR

SUPPORT, CONTRIBUTING AND EXTENDING

pyscal welcomes and appreciates contribution and extension to the module. Rather than local modifications, we request that the modifications be submitted through a pull request, so that the module can be continuously improved.

4.1 Reporting and fixing bugs

In case a bug is found in the module, it can be reported on the issues page of the repository. After clicking on the new issue button, there is template already provided for Bug report. Please choose this and make sure the necessary fields are filled. Once a bug is reported, the status can once again monitored on the issues page. Additionally, you are of course very welcome to fix any existing bugs.

4.2 New features

If you have an idea for new feature, you can submit a feature idea through the issues page of the repository. After choosing new issue, please choose the template for feature request. As much as information as you can provide about the new feauture would be greatly helpful. Additionally, you could also work on feature requests already on the issues page. The following instructions will help you get started with local feature development.

4.2.1 Setting up local environment

1. The first step is to fork pyscal. A detailed tutorial on forking can be found here. After forking, clone the repository to your local machine.
2. We recommend creating a virtual environment to test new features or improvements to features. See this link for help on managing environments.
3. Once the environment is set up, you can create a new branch for your feature by `git checkout -b new_feauture`.
4. Now implement the necessary feature.
5. Once done, you can reinstall pyscal by `python setup.py install`. After that please make sure that the existing tests work by running `pytest tests/` from the main module folder.
6. If the tests work, you are almost done! If the new feature is not covered in existing tests, you can to write a new test in the tests folder. pyscal uses pytest for tests. This link will help you get started.
7. Add the necessary docstrings for the new functions implemented. pyscal uses the numpy docstring format for documentation.

8. Bonus task: Set up few examples that document how the feature works in the `docs/source/` folder and link it to the examples section.
9. Final step - Submit a pull request through github. Before you submit, please make sure that the new feature is documented and has tests. Once the request is submitted, automated tests would be done. Your pull request will fail the tests if - the unit tests fail, or if the test coverage falls below 80%. If all tests are successful, your feauture will be incorporated to pyscal and your contributions will be credited.

If you have trouble with any of the steps, or you need help, please send an email and we will be happy to help! All of the contributions are greatly appreciated and will be credited in Developers/Acknowledgements page.

CHAPTER

FIVE

HELP AND SUPPORT

In case of bugs and feature improvements, you are welcome to create a new issue on the github repo. You are also welcome to fix a bug or implement a feature. Please see the extending and contributing section for more details.

Any other questions or suggestions are welcome, please contact us.

CHAPTER

SIX

CITING THE CODE

If you use pyscal in your work, the citation of the following article will be greatly appreciated:

Sarath Menon, Grisell Díaz Leines and Jutta Rogal (2019). pyscal: A python module for structural analysis of atomic environments. Journal of Open Source Software, 4(43), 1824, <https://doi.org/10.21105/joss.01824

Click to copy citation in bib format.

CHAPTER

SEVEN

ACKNOWLEDGEMENTS

7.1 Developer

- Sarath Menon
  sarath.menon@pyscal.org

7.2 Contributers

- Jan Janßen - developing and maintaining a conda-forge recipe.
- Pedro Antonio Santos Flórez - addition of the pairwise multicomponent short range order parameter.

7.3 Acknowledgements

We acknowledge Bond order analysis code for the inspiration and the base for what later grew to be `pyscal`. We are also thankful to the developers of *Voro++* and pybind11 for developing the great tools that we could use in `pyscal`. We are grateful for the help and support received during the E-CAM High Throughput Computing ESDW held in Turin in 2018 and 2019. This module was developed at the Interdisciplinary Centre for Advanced Materials Simulation, at the Ruhr University Bochum, Germany.

In addition, the following people are acknowledged:

- Grisell Díaz Leines
- Jutta Rogal
- Alberto Ferrari
- Abril Azócar Guzmán
- Matteo Rinaldi
- Yanyan Liang
- David W.H. Swenson
- Alan O'Cais

CHAPTER

EIGHT

LICENSE

8.1 pyscal License

`pyscal` uses the BSD 3-Clause New License . A full description is available in the above link or in the repository.

In addition, `pyscal` license of other codes that `pyscal` uses are given below-

8.2 Voro++ license

Voro++ Copyright (c) 2008, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University of California, Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code ("Enhancements") to anyone; however, if you choose to make your Enhancements available either publicly, or directly to Lawrence Berkeley National Laboratory, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative works thereof, in binary and source code form.

8.3 pybind 11 license

8.4 sphinx theme license