Version of Record: https://www.sciencedirect.com/science/article/pii/S0010465523004083
Manuscript_45283cfe8c5a2a563756180b8077a533

Expanding PyProcar for new features, maintainability, and reliability

Logan Lang[a,*], Pedram Tavadze[a], Andres Tellez[a], Eric Bousquet[b], He Xu[b], Francisco Muñoz[c], Nicolas Vasquez[c], Uthpala Herath[d], Aldo H. Romero[a]

[a]*Department of Physics and Astronomy, West Virginia University, Morgantown, WV 26505-6315, USA*
[b]*Physique Théorique des Matériaux, QMAT, CESAM, Université de Liège, B-4000 Sart-Tilman, Belgium*
[c]*Departamento de Física & CEDENNA, Facultad de Ciencias, Universidad de Chile, Santiago, Chile*
[d]*Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA*

Abstract

This paper presents a comprehensive update to PyProcar, a versatile Python package for analyzing and visualizing density functional theory (DFT) calculations in materials science. The latest version introduces a modularized codebase, a centralized example data repository, and a robust testing framework, offering a more reliable, maintainable, and scalable platform. Expanded support for various DFT codes broadens its applicability across research environments. Enhanced documentation and an example gallery make the package more accessible to new and experienced users. Incorporating advanced features such as band unfolding, noncollinear calculations, and derivative calculations of band energies enriches its analytic capabilities, providing deeper insights into electronic and structural properties. The package also incorporates PyPoscar, a specialized toolkit for manipulating POSCAR files, broadening its utility in computational materials science. These advancements solidify PyProcar's position as a comprehensive and highly adaptable tool, effectively serving the evolving needs of the materials science community.

Keywords: electronic structure; DFT; post-processing

NEW VERSION PROGRAM SUMMARY

Program Title: PyProcar
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link: https://github.com/romerogroup/pyprocar
Code Ocean capsule: (to be added by Technical Editor)
Licensing provisions(please choose one): GPLv3
Programming language: Python
Supplementary material: Pyprocar-Supplementary Information
*Journal reference of previous version: https://doi.org/10.1016/j.cpc.2019.107080**
Does the new version supersede the previous version?: Yes
Reasons for the new version: Changes in the directory structure, the addition of new features, enhancement of the manual and user documentation, and generation of interfaces with other

*Corresponding author.
E-mail address: lllang@mix.wvu.edu

electronic structure packages
*Summary of revisions: These updates enhance its capabilities and ensure developers' and users' maintainability, reliability, and ease of use.**
Nature of problem: To automate, simplify, and serialize the analysis of band structure and Fermi surface, especially for high throughput calculations.
Solution method: Implement a Python library able to handle, combine, parse, extract, plot, and even repair data from density functional calculations from diverse electronic structure packages. PyProcar uses color maps on the band structures or Fermi surfaces to give a simple representation of the relevant characteristics of the electronic structure.
Additional comments: PyProcar can produce high-quality figures of band structures and Fermi surfaces (2D and 3D), projection of atomic orbitals, atoms, and/or spin components.

1. Introduction

Density functional theory (DFT) has emerged as a prominent quantum mechanical methodology for investigating materials' electronic structures and properties from first principles. It is a powerful and extensively employed computational tool in condensed matter physics, materials science, and chemistry. With reasonable computational costs, DFT enables sensibly accurate predictions of material properties. For periodic solids, DFT calculations are commonly conducted using diverse software packages, including Abinit[1, 2, 3], VASP[4, 5, 6, 7], SIESTA [8] and Quantum Espresso[9, 10]. Each software package has distinct input and output formats, syntax, and data structures catering to specific research needs and preferences.

In recent years, significant strides have been made in developing packages, web interfaces, or application programming interfaces (APIs) aimed at handling high-throughput and streamlining workflows for property calculations such as PyMatgen[11], AIIDA [12], AFLOW [13], AbiPy [3], etc. These efforts have been directed towards creating user-friendly interfaces that seamlessly interact with electronic structure packages, essential tools in materials science, and computational chemistry. These advancements have undeniably enhanced the efficiency and accessibility of conducting simulations, computations, and modeling related to materials, providing better experiences for both experienced and inexperienced users. However, as the landscape of materials characterization evolves, a new challenge has emerged as a focal point: the intricate process of data extraction and subsequent analysis the so-called post-processing analysis. This phase constitutes a critical bottleneck in the overall workflow of materials characterization. While the tools for performing calculations have become increasingly sophisticated and user-friendly, extracting meaningful insights from the generated data presents challenges that cannot be overlooked.

Efforts are being directed towards developing methodologies and tools that can automate and streamline data extraction and analysis [14, 15, 16, 17]. Machine learning algorithms, for instance, are being harnessed to identify patterns within datasets that might be challenging to discern through manual methods. Even in these cases, images and specific data must be extracted from these electronic structure calculations [18, 19]. Therefore, there is a growing need for packages that support direct data extraction from these calculations. This will facilitate the analysis and practical interpretation into the broader context of materials

research. In particular, the significance of pre- and post-processing steps in DFT simulations is often overlooked despite their critical roles in the computational workflow. Pre-processing includes preparing input files and setting up initial conditions, while post-processing encapsulates the analysis and visualization of output data to deduce crucial information about material properties. The extensive data generated during DFT calculations can be daunting and necessitates efficient, user-friendly tools to facilitate the management and interpretation of this data

PyProcar was conceived to address this requirement. It offers comprehensive pre- and post-processing solutions for DFT calculations, frequently employed in predicting material properties. The PyProcar package eases the analysis and visualization of DFT results generated by the following widely used codes: Abinit [1, 2, 3], VASP [4, 5, 6, 7], and Quantum Espresso [10, 9], Siesta [8], ELK [20], and the density functional tight-binding code DFTB+ [21].

While several commendable packages such as FermiSurfer[22], Abipy[23, 3], pymatgen[11], and XCrySDen[24] offer similar functionalities, PyProcar stands out with its simple line interfaces. The PyProcar approach is designed in the spirit that powerful visualization and analysis should not come at the cost of complexity. Users, regardless of their level of experience, can effortlessly exploit the capabilities of PyProcar. A testament to its user-centric design is the ability to interface with multiple DFT codes by altering a single function call argument. This seamless adaptability ensures that researchers are not bound by the constraints of a particular DFT package, offering unparalleled flexibility and ease of use.

PyProcar has evolved from a single script for plotting projected band structures, derived from the PROCAR file output, to a broader utility tool. The publication of our first paper [25] led to its expansion for more general analyses of electronic structure codes, including Fermi surface plotting, band unfolding, property projections, and band structure comparisons from multiple DFT packages. Early versions of PyProcar were restricted to reading the PROCAR format for parsing band projections, limiting its broad applicability. This restriction required the parent DFT packages to convert their output to the PROCAR format, a feature only available in Abinit and VASP. However, PyProcar's parsing capabilities have since evolved to accommodate multiple electronic structure code formats, including Abinit, VASP, Quantum Espresso, Siesta, ELK, and BXSF. Additionally, PyProcar now includes classes and methods for manipulating and plotting the projected density of states, a feature absent in previous versions. We have introduced several core objects, like ElectronicBandStructure, DensityOfStates, Structure, and FermiSurface3D, to streamline data manipulation for analysis and visualization.

Moreover, we have significantly improved the documentation, enabling users to quickly and easily utilize the code. An example gallery has been added as a practical demonstration of the code's usability and showcasing the various features available in PyProcar. Collectively, these enhancements contribute to a more versatile, user-friendly, and powerful tool for electronic structure analysis, broadening its appeal and applicability across the scientific community.

For those interested in using PyProcar, detailed instructions for the installation process are provided in the Supplementary Information (SI) of this paper. Multiple installation avenues are supported to cater to varied user preferences, including the Python package manager (pip), the conda package manager, direct GitHub source code cloning

https://romerogroup.github.io/pyprocar/index.html, and more. Comprehensive documentation is available to assist users through the installation process and to provide further insights into the package's functionalities. Additionally, a supplementary Google Drive link offers extended resources, including example files. We encourage users to utilize these resources to use PyProcar in their research.

In this paper, we present the updated PyProcar package, which not only expands its support to multiple DFT codes but also enhances its overall maintainability, reliability, and usability. These updates are aimed at making PyProcar a more versatile and accessible tool, empowering researchers in the field of materials science to better analyze and visualize DFT results, ultimately contributing to the advancement of material property prediction and discovery.

2. Summary of Improvements

In this paper, we present significant updates and improvements to the PyProcar package. These updates enhance its capabilities and ensure maintainability, reliability, and ease of use for developers and users. The main contributions of this work are as follows:

1. **Enhanced and Expanded Support for DFT Codes**: PyProcar has improved its interface for previously supported codes—Abinit and VASP, while extending its support to include new codes: BXSF format, DFTB+, ELK, Quantum Espresso,and Siesta. This broadens its applicability across various computational platforms.
2. **Streamlined data processing**: The introduction of a core class object standardizes the parsed DFT code data, reducing code complexity and allowing for smoother integration with different DFT codes, even to those not included in our release.
3. **Modularized code structure**: By confining DFT code dependencies to the input/output (I/O) module, we have simplified the overall code structure, making it more maintainable and easier to extend for future developments, in particular for new properties for interested developers.
4. **Updated documentation**: We have updated and expanded the documentation to better assist users in understanding and utilizing PyProcar's features.
5. **Enhanced features**: We have incorporated several new functionalities, including plotting of energy levels with only one k-points (*e.g.* molecules, defects), calculation of the inverse participation ratio, the density of states plotting, refined 2D Fermi surface visualization, a 3D Fermi surface cross-section widget, an interactive 3D Fermi surface isosurface slider, a GIF creator for dynamic representation, and a class that handles all band structure information and allows for band derivatives and integrals to be calculated.
6. **Geometry analysis**: We have added functionalities that automatize the analysis of the input geometry. For instance, point defects, surfaces, etc. can be detected [26]. These features can be correlated with electronic structure allowing automatic recognition of defect/surface states or similar. Also geometry descriptors can be easily integrated. Currently the *Global Connectivity* descriptor is implemented[27].

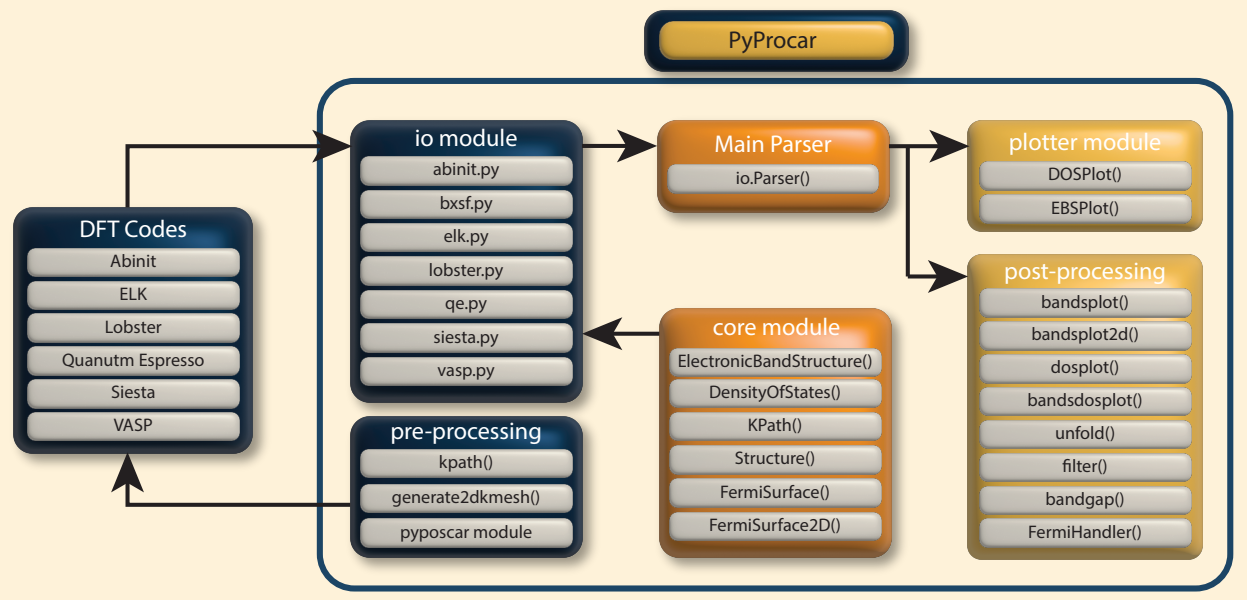


Figure 1: Structural overview of the PyProcar library

3. Refining Architecture, DFT Compatibility, and Usability

3.1. PyProcar Architecture

This section delves into the revised architecture of the PyProcar package, which has been overhauled for enhanced structure, easier maintenance, and an improved user experience. An overview of the architectural changes in the library is illustrated in Figure 1.

As with the previous rendition of PyProcar, this updated version continues to interface with DFT codes either as a pre-processing or post-processing instrument. To facilitate this interaction with other DFT codes, modifications were implemented for a more seamless integration with the existing codebase. The first step in this process involved the creation of core objects that standardize the output generated by the DFT codes. These core objects are instantiated within the input/output parsers, after which the primary Parser object extracts the initialized core object from the diverse DFT codes.

Subsequently, the Parser object is passed to the post-processing tools or the plotting module. Importantly, this procedure simplifies the integration of new code into our package. The only required inputs are the necessary information for the initialization of the core objects.

3.1.1. Modularized Library

The PyProcar library has been reorganized into more user-friendly module names, which improves the package's maintainability and facilitates easier extension and integration of new features. The main sub-packages in the updated library are:

'core': This sub-package hosts modules, which are responsible for representing main data objects used in plotting and handling data. It standardizes the parsed data from various DFT codes, ensuring consistency and compatibility across different input formats.

'plotter': This sub-package contains the Plotter class, which controls the generation of plots based on the processed DFT data. The Plotter class simplifies the creation of various types of plots and provides a unified interface for visualization tasks.

'io': This sub-package is responsible for parsing the output data from different DFT codes. It abstracts the parsing process, allowing users to work with various DFT codes without having to worry about the underlying differences in data formats.

'scripts': This sub-package contains easy-to-use functions that provide convenient access to common tasks and analysis routines. These scripts facilitate user interaction with the package and can serve as a starting point for custom analysis workflows. These scripts are also used in the command-line functionality of pyprocar

'utils': This sub-package contains specialized mathematical functions, plot customization configurations, orbital name indexing, and other utilities not provided in standard Python libraries

The updated modular architecture of PyProcar not only enhances its maintainability but also improves its usability for both developers and users. By clearly separating the different components of the package, the new structure allows for easier navigation, a better understanding of the code base, and efficient implementation of new features and improvements.

3.2. DFT Code Support

In this section, we discuss the expansion of PyProcar to support a wider range of DFT codes, specifically Abinit, VASP, Quantum Espresso, ELK, and Siesta. This support now includes the non-DFT electronic structure code DFTB+ package. The main challenges faced during this process was handling the varying data formats produced by each DFT code and addressing DFT code dependencies throughout the package. To overcome these challenges and enable integration with multiple DFT codes, we introduced several new classes to standardize the parsed data and created a unified Parser class to handle the parsing process.

3.2.1. Standardizing Data with Core Classes

We introduced the following core classes to standardize the data from different DFT codes:

1. **ElectronicBandStructure**: This class is designed to store and manage data related to electronic band structures derived from Density Functional Theory (DFT) calculations. It stores details such as k-points, band energies, and other potential characteristics that define a material's electronic structure, crucial for understanding the material's properties and behavior. The class serves as an interface for consistent manipulation and analysis of electronic band structure data, providing a standardized data structure across different Density Functional Theory (DFT) codes, thus allowing seamless data exchange and integration.
2. **DensityOfStates**: This class is designed to represent and handle data related to the density of states (DOS) resulting from density functional theory (DFT) calculations. This includes data about energies, total densities, and projected densities. It takes as inputs energies, total, and projected numpy arrays which represent points on the energy spectrum, densities at each point, and projections of elements, orbitals, spins,

respectively. It optionally accepts an interpolation_factor parameter that determines the density of states points' increment during the interpolation. By default, the interpolation factor is set to 1.

3. **BandStructure2D**: class is utilized to represent and manipulate the 2D band structure of a material, which is a critical aspect in examining the electronic properties of a material. It takes parameters such as the electronic band structure, spin quantum number, interpolation factor, and others to construct a 2D representation of the band structure. It allows for the generation of a 2D band structure and the combining of band surfaces to form a complete band structure, making it an indispensable tool in materials science and physics for studying the energy levels available for electrons to occupy in a material.
4. **FermiSurface3D**: This class is designed to represent and manipulate a 3D Fermi surface, which is crucial in analyzing the electronic structure of a material. This class takes in parameters such as the electronic band structure, Fermi energy, interpolation factor, and others to construct a 3D representation of the Fermi surface. It provides methods to generate isosurfaces for each band, combine them into a complete Fermi surface, and calculate the area of the Fermi surface. It's used for advanced materials science applications where understanding the behavior of electrons in a material, especially at the Fermi level, is of significant importance.
5. **Kpath**: This class is designed to store and manage k-path information necessary for band structure calculations in Density Functional Theory (DFT) simulations. It helps to standardize the representation of k-path data across different DFT software.
6. **BrouillinZone**: This class is a specific type of Surface that calculates the first Brillouin zone corresponding to a given reciprocal lattice. It uses the Wigner-Seitz method to calculate the vertices and faces of the Brillouin zone. Additional utility methods are provided for fixing normal direction to ensure a correct graphical representation.
7. **BrouillinZone2D**: This class extends the functionality of BrillouinZone to two dimensions. It takes an additional two parameters, e_min, and e_max, which are used to map the Z-coordinates of the 3D vertices to a 2D plane. This class can be used when a 2D representation of the Brillouin zone is required. Both BrillouinZone and BrillouinZone2D inherit from the Surface class and can therefore be plotted directly using pyvista.PolyData [28] functionalities.
8. **Structure**: This class is designed to encapsulate and manage the structural data of a material, storing crucial information such as atomic species, atomic coordinates, lattice parameters, and symmetry rotations. It offers a standardized way to represent structural information, facilitating seamless integration of data across different Density Functional Theory (DFT) codes. This consistent representation of structural data simplifies the analysis and modeling processes, enhancing compatibility and efficiency in materials science research and computational material simulations.

3.2.2. Unified Parsing with the Parser Class

To address the DFT code dependency issue, we created the Parser class, which handles the parsing of the ElectronicBandStructure, DensityOfStates, Kpath, and Structure objects. The Parser class serves as a central point for parsing the output data from various DFT codes, making it easier to integrate new DFT codes into the package.

Instead of calling multiple parsers throughout the codebase, users can now call a single Parser object and pass the DFT code string parameter to obtain the required core data object. This approach streamlines the parsing process and simplifies the code structure, making it more maintainable and easier to extend. This approach works even for DFTB+, which uses a different notation than usual DFT codes.

By implementing a Parser class for each DFT code, developers can easily expand PyProcar's support to additional DFT codes, further enhancing the package's versatility and applicability across various computational platforms.

The expanded DFT code support in PyProcar greatly broadens its user base and facilitates more efficient and consistent handling of DFT results, regardless of the underlying DFT code. This expansion significantly enhances PyProcar's capabilities and positions it as a comprehensive and versatile tool for the analysis and visualization of DFT calculations in the field of materials science.

3.2.3. Required Files for Supported Codes

To make the most out of PyProcar's functionalities, users must prepare specific output files generated by the various supported DFT codes. A comprehensive list of required files and versions for each supported code, along with more detailed information about the prerequisites of DFT codes, has been provided in the Supplementary Information (SI) of this paper. For further insights and elaboration on these requirements, readers are also encouraged to consult our official documentation at

https://romerogroup.github.io/pyprocar/dftprep/index.html

By ensuring that these files are generated by the respective DFT codes, users can more easily integrate their data into PyProcar for analysis and visualization.

3.3. Example Data Repository and Testing

To enhance the reliability and maintainability of PyProcar, we have implemented a example data repository stored on Google Drive, which users and developers can access at any time. This repository provides a consistent and up-to-date source of data for testing and development purposes, ensuring that the package remains robust and reliable.

3.3.1. Repository Structure

The example data repository is organized in a hierarchical structure: `<material>/<DFT code>/<magnetic calculation type>/<calculation type>`. This organization facilitates easy navigation and retrieval of data, allowing users and developers to quickly locate and download the relevant files for their specific needs.

Users and developers can download data from the centralized database using two convenient functions:

3.3.2. Data Download Functions

1. pyprocar.download_dev_data(): This function is intended for developers, providing access to the data necessary for testing and developing new features.
2. pyprocar.download_example(): This function is designed for users, offering example data sets that can be used to explore and demonstrate the capabilities of the PyProcar package.

3.3.3. Testing

The example data repository serves as the foundation for testing the package. By providing a consistent source of data for testing purposes, the repository ensures that the tests accurately reflect the package's performance under a variety of conditions and inputs. This testing framework helps identify and address potential issues, contributing to the overall reliability and robustness of PyProcar.

The implementation of the data repository and a testing framework significantly enhances the reliability and maintainability of PyProcar. By providing a consistent source of data and facilitating rigorous testing, these features ensure that the package remains dependable, enabling users and developers to confidently rely on PyProcar for their DFT analysis and visualization needs.

4. New Features

This section highlights new features introduced to PyProcar, each of which enhances its versatility and expands its capabilities for analyzing and visualizing Density Functional Theory (DFT) calculations. These new features are detailed in individual subsections.

4.1. Configuration Files

One of the significant updates in the latest version of PyProcar is the introduction of centralized configuration files that govern the plotting options for various visualization modules, including the Fermi Surface 2D, Fermi Surface, Band Structure 2D, Band Structure, and the Density of States. Previously, plotting options were hard-coded or passed as parameters, which, while functional, lacked flexibility and scalability.

The configuration files, written in the human-readable YAML format, organize plotting parameters under distinct sections for each visualization module. Each section contains key-value pairs that define various plotting attributes such as color schemes, axis limits, labels, and more. This structured approach provides a clear overview of all available options, making it intuitive even for users unfamiliar with the internal workings of PyProcar. Furthermore, the modular design of the configuration management system paves the way for future enhancements. As PyProcar continues to evolve, adding new plotting options or even entirely new visualization modules becomes substantially more straightforward. Developers can simply extend the configuration file with new sections or parameters, ensuring that the software remains adaptable to emerging visualization needs.

The following examples illustrate this feature:

1. **Band Structure Plotting (`bandsplot`):** By default there is color scheme set in the configuration file for band structures. For a specific plot, this can be changed by overwriting it as the keyword argument `color`. This can be achieved as follows:

   ```
   bandsplot(data, color='blue')
   ```

2. **Density of States Plotting (`dosplot`):** Similarly, to adjust the Fermi line width and linestyles for a particular Density of States plot:

   ```
   dosplot(data, fermi_linewidth=[2.5], fermi_linestyle='solid')
   ```

These examples underscore the ease with which users can customize their plots in PyProcar. By merely adding keyword arguments to the function calls, users can swiftly tailor their visualizations without the need to modify the configuration file or the core code base.

For users keen on exploring the numerous plotting options available in PyProcar, there are multiple ways to do this. Firstly, the configuration file itself serves as a comprehensive guide. Embedded within it are descriptions of each option, providing users with immediate clarity on the purpose and potential values of each parameter. Additionally, for a more interactive approach, users can opt to print the available plotting options directly from their function calls. By simply passing the argument `print_plot_opts`, a detailed list of all available options, along with their current values, will be displayed. This feature is particularly handy for users who wish to quickly adjust their plots without delving into external documentation. Lastly, for a thorough understanding, users are encouraged to consult the official PyProcar documentation. The docs are meticulously curated to offer detailed explanations, examples, and potential use cases for each plotting option, ensuring that users can harness the full potential of PyProcar's visualization capabilities.

4.2. Atom-like Energy Levels Plot

The *Atom-like Energy Levels Plot* feature of PyProcar provides an avenue to visualize atomic, molecular and defect energy levels, with only one k-point. Also it can be used with for plotting the energy bands of a material at one specific k-point. This functionality proves particularly useful for dissecting the electronic structure and discerning the impact of defects in materials. For demonstration purposes, we use hexagonal boron nitride with carbon defects[29] (hBN-C_NC_N) as an example.

The Python code snippet below illustrates how to invoke this feature. The function `pyprocar.bandsplot` is set to 'atomic' mode', enabling visualization of energy levels for specified carbon atoms. As seen in Figure 2, this allows for a nuanced analysis of the role of the carbon defect in the energy landscape of the material. Notably, carbon atoms introduce states in close proximity to the Fermi level, specifically at spin-0 bands-195, spin-0 bands-196, spin-1 bands-195, and spin-1 bands-196. Listing 1 shows the script used to generate Figure 2.

```
import pyprocar

# Plot atomic bands using VASP as the DFT code
# Specify the directory, mode, energy limits, and atoms for the plot
pyprocar.bandsplot(
    code='vasp',           # DFT code used for calculations
    dirname=data_dir,      # Directory containing data
    mode='atomic',         # Mode for plotting
    elimit=[-0.4, 3],      # Energy range for the plot
    atoms=[96, 97]         # Atoms to be plotted
)
```

Listing 1: Example code for plotting atomic bands using PyProcar and VASP.

4.3. Inverse Participation Ratio

Often it is needed to search for *localized* modes within the band structure, typical examples are surface/interface states and defect levels. The usual procedure for detecting them

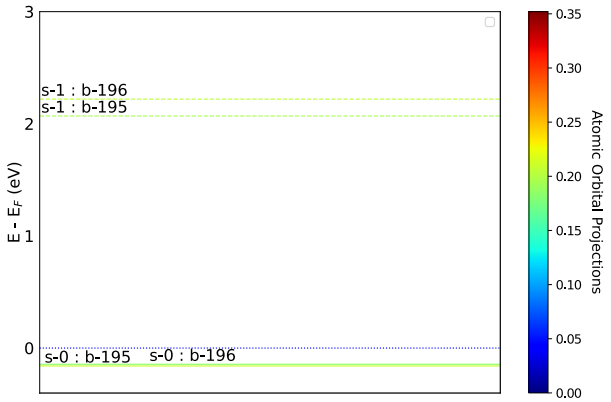


Figure 2: Visualization of atom-like energy levels in close vicinity to the Fermi level for hBN-CNCN. The color scale corresponds to the carbon atom's contribution to the band levels near the Fermi level (spin-0 bands-195, spin-0 bands-196, spin-1 bands-195, spin-1 bands-196).

is looking for bands with a large projection around the atoms at the surface or defect. This procedure is both cumbersome for the user and error-prone. For instance, the lowest unoccupied levels of the neutral C_N defect in h-BN has practically no projection on the defect atom and its nearest neighbors. This delayed its identification as a single-photon emitter [30, 31]. A much simpler way to detect these localized levels is by means of the *Inverse Participation Ratio*, defined as

$$IPR_{nk} = \frac{\sum_a |c_{nka}|^4}{\left(\sum_a |c_{nka}|^2\right)^2}, \tag{1}$$

where c are the wavefunction's coefficient, the indexes n, k, a are the band, k-point and atom, respectively. This function has been applied in the context of Anderson localization [32, 33, 34]. However, it can capture any kind of localization.[35, 36, 37] A perfectly localized state — *i.e.* localized in a single atom — would have $IPR = 1$, but a fully extended state has $IPR = \frac{1}{N}$, with N being the total number of atoms.

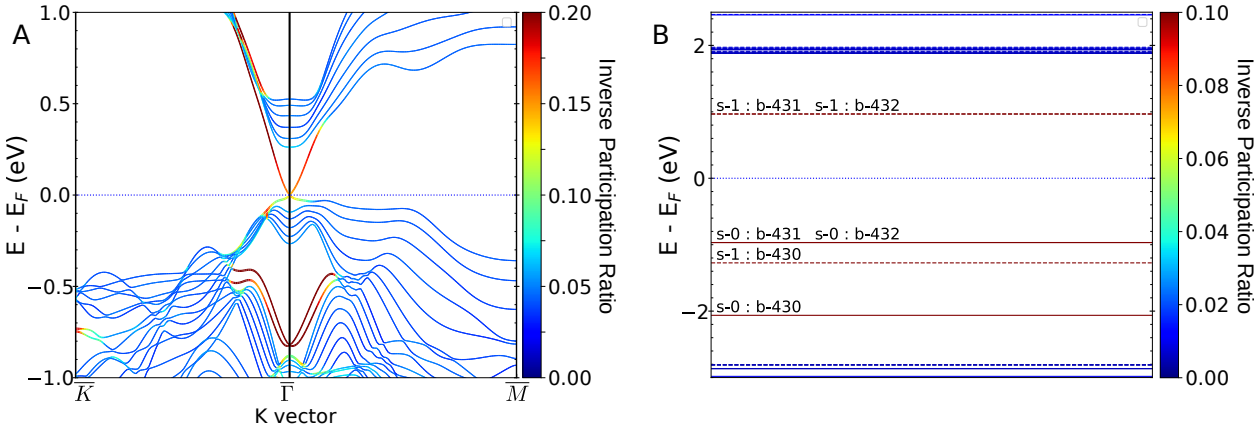


Figure 3: Examples of band structures colored by their inverse participation ratio, IPR. (a) Topological insulator Bi_2Se_3, its surface states are in red. (b) Defect levels of the NV^- color center in diamond, several localized states (dark red) exist within the fundamental band gap, particularly the spin minority has two degenerate unoccupied levels enabling the optical transition.

In PyProcar, using the IPR in `bandsplot` is straightforward, it can be done simply by setting `mode='ipr'`. By definition, all orbitals and atoms are summed. We have included two typical, and very different examples. The first example is the detection of topologically protected surface states in Bi_2Se_3,[38] see Fig. 3a. The whole slab has six van der Waals layers (quintuple layers), each is five atom thick. The surface states localize on the outer quintuple layers, in contrast, an extended state covers the six quintuple layers. The ratio between the localization of both types of states is 1 to 3, and the IPR has enough resolution to provide a clear visual identification. Listing 2 shows an example of this process.

```
import pyprocar

pyprocar.bandsplot(
    dirname='.',  # Actual directory
    elimit=[-1.0, 1.0],
    mode='ipr',  # Selecting the 'IPR'
    code='vasp',
    spins=[0],
    clim=[0, 0.2]  # Set the colormap range for IPR
)
```

Listing 2: Example code for plotting bands with Inverse Participation Ratio (IPR) using PyProcar and VASP.

The second example is the NV^- defect in diamond, it is a negatively charged N substitution plus an adjacent vacancy. This defect is of interest as a source of single photons. Its ground state is a triplet, allowing the control of the spin by microwave radiation [39]. The energy levels of a diamond supercell hosting the defect are portrayed in Fig. 3b. The supercell has 215 atoms, hence $IPR \to 0$ for bulk states (blue lines). Several defect levels lie within the fundamental band gap of diamond (dark red lines). The closest levels to the Fermi energy are double degenerate (*i.e.* triplet), but only occupied for the spin majority. Hence, according to the optical transition takes place between the bands with index $430 \to 431$ or $430 \to 432$ of the spin channel labeled 'spin-1'. The calculation of the main emission line involves a calculation of the excited state, which can be simulated by fixing the occupations of the mentioned levels, *i.e.* the Δ SCFmethod [40]. Listing 3, shows an example of this process.

```
import pyprocar

pyprocar.bandsplot(
    dirname='.',  # Actual directory
    elimit=[-3.0, 2.5],
    mode='ipr',  # Selecting the 'IPR'
    code='vasp',
    spins=[0, 1],  # I prefer both spin values in a single figure
    clim=[0, 0.1]  # Set the optimal colormap range
)
```

Listing 3: Example code for plotting bands with Inverse Participation Ratio (IPR) for both spin values using PyProcar and VASP.

4.4. Automatic correlation between geometrical and electronic 'features'

One of the most powerful capabilities of PyProcar is allowing one to correlate the real space with the electronic structure, for instance, finding surface states or defect levels. Tra-

ditionally, this task is done by the user, providing a list of atoms representing the surface or the defect (parameter `'atom'` in `bandsplot`). Also, the user needs to choose a relevant energy window for the plot and set the boundaries of the color scale to highlight the relevant states. That process is both tedious and error prone: for instance, the user needs to find the special atoms (*e.g.* defect, surface, etc.) and take care of whether the indexes are 0- or 1-based.

In the latest update, we have introduced the `autobandsplot` function, designed to automate several key tasks for enhanced visualization and analysis. Specifically, the function aims to:

- Determine an optimal energy window for the plot, which includes bulk-like bands both above and below the fundamental band gap for insulators, as well as any localized states within that gap.
- Identify important real-space features, such as defects, surfaces, and van der Waals layers.
- Locate the localized electronic states within the selected energy window.
- Calculate suitable values for the color map to emphasize these localized states.

All these tasks are executed without requiring user intervention. The identification of real-space features is carried out using PyPoscar, as detailed in Section 4.12. Localized states are identified through the Inverse Participation Ratio (IPR), as explained in Section 4.3. The function correlates the geometry and electronic structure by evaluating the participation of relevant atoms in the IPR calculations.

This automated identification of key features is most effective when the atoms of interest are statistically distinct from the rest of the system, both in real space and in electronic structure. In scenarios where such distinctions are not readily apparent, the function will default to generating a standard band structure plot. It is important to note that while our current implementation is robust, there may be some geometrical features it does not yet capture. However, we anticipate that the function will continue to improve based on user feedback.

As an example of this functionality, we calculate a slab of a theoretically-predicted phase Bi, which is topologically nontrivial[41]. It features surface states with Dirac cones at high-symmetry points, see Fig. 4. As a reference, the surface states of the stable bulk phase Bi are difficult to calculate with DFT[42]. Returning to the predicted topological crystalline phase of Bi, Listing 4 shows an example of the code used to obtain the figure.

```
import pyprocar

pyprocar.autobandsplot(code='vasp')
```

Listing 4: Example code for automatically plotting band structures using PyProcar and VASP.

The title of Fig. 4 is `Defect 0`, which correspond to the upper surface of the slab, the other surface generates a second figure, with title `Defect 1`. When running the Listing 4, a file `report.txt` is generated with information about the atoms comprising each defect, and the associated localized states.

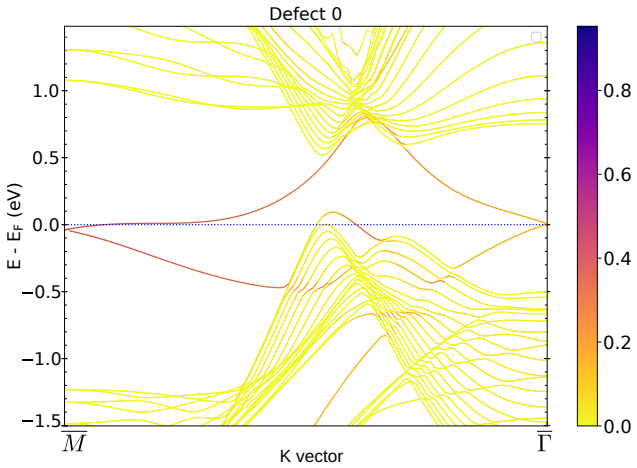


Figure 4: Surface states of a theoretically predicted topological crystalline phase of Bi. The color coding reflects the projection into 'defects', such as a surface. These 'defects' are automatically detected.

4.5. Support for Density of States and Projected Density of States

The PyProcar package now has enhanced capabilities for computing and visualizing both the density of states (DOS) and projected density of states (PDOS). These new functionalities deepen our understanding into the electronic structure of a material, providing a more comprehensive analysis of the system. To showcase the capabilities of these features, we performed a colinear spin-polarized calculation on $SrVO_3$. The resulting visualizations for DOS and PDOS are shown in Figure 6. The orbital indexing is given in Figure 5

The `dosplot` function facilitates plotting the density of states and accepts the directory containing the calculation data and the type of DFT code used as arguments. Furthermore, this function provides a variety of modes that can be selected via the **mode** keyword argument. The orbital indexing are given in Figure 5.

For instance, Listing 5 outlines how one can plot the parametric density of states, highlighting contributions specifically from d orbitals.

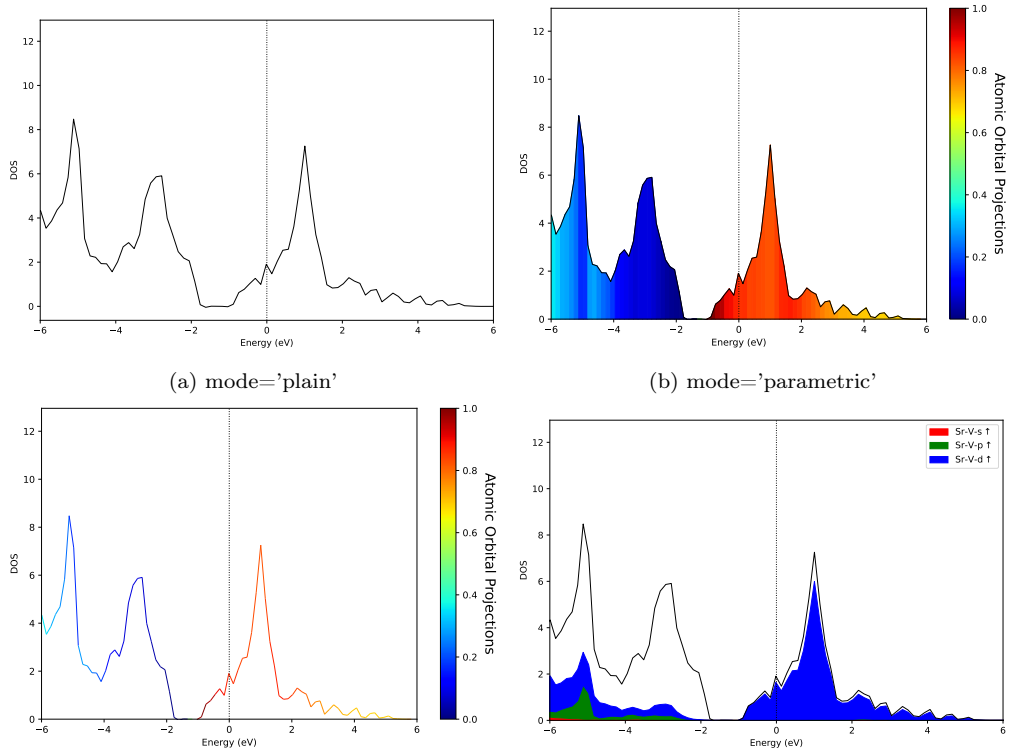
s	p_y	p_z	p_x	d_{xy}	d_{yz}	d_{z^2}	d_{xz}	$d_{x^2-y^2}$	$f_{y(3x^2-y^2)}$	f_{xyz}	f_{yz^2}	f_{z^3}	f_{xz^2}	$f_{z(x^2-y^2)}$	$f_{x(x^2-3y^2)}$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 5: Orbital indexing for the `'orbitals'` keyword argument

```
import pyprocar

pyprocar.dosplot(
    code='vasp',
    dirname=data_dir,
    mode='parametric',
    atoms=[0, 1],  # Target atoms for the plot
    orbitals=[4, 5, 6, 7, 8]  # Specifying d orbitals
)
```

Listing 5: Example code for plotting density of states (DOS) for specific atoms and d orbitals using PyProcar and VASP.



(a) mode='plain'

(b) mode='parametric'

(c) mode='parametric-line'

(d) mode='stack-orbitals'

Figure 6: Examples of different modes available in `dosplot` for $SrVO_3$. The subfigures illustrate various modes such as *plain*, which presents a straightforward visualization of the DOS, and *parametric* or *parametric-line*, that emphasize contributions from specific atomic orbitals—here, the d-orbitals of Sr and V atoms are under focus. *stack-orbitals* mode provides another representation by scaling the area under the DOS curve based on the specified projections.

4.6. Unified Plotting of Band Structure and Density of States

A notable enhancement in PyProcar is the introduction of a feature that allows for the simultaneous plotting of the EBS and DOS, as illustrated in Figure 7. We demonstrate this capability using colinear spin-polarized calculations on $SrVO_3$. This combined view offers a more intuitive and comprehensive understanding of a material's electronic structure. Critically, the energy axes for both the EBS and DOS are perfectly aligned, allowing for visual comparison between the two.

The `bandsdosplot` function serves to create these unified plots. It accepts two primary arguments—*bands_settings* and *dos_settings*—which are dictionaries containing settings specific to *bandsplot* and *dosplot*, respectively. This ensures that the various functionalities of the two methods are seamlessly integrated into one unified plot.

To further clarify, the Listing 6 demonstrates how to utilize this feature.

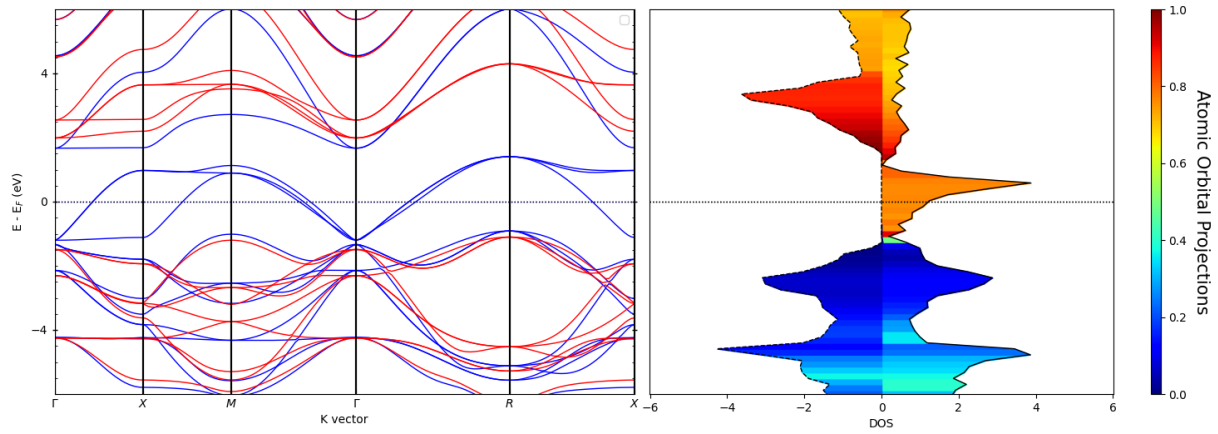


Figure 7: Simultaneous visualization of the band structure and density of states for $SrVO_3$. The band structure is rendered in plain mode, employing different colors to distinguish between spin channels. The DOS plot, on the other hand, represents both spin channels, with color scales indicating the contributions from the d-orbitals of Sr and V atoms.

```
import pyprocar

bands_settings = {
    'mode': 'plain',
    'dirname': bands_dir,
    'linestyle': ['solid', 'solid'],
    'elimit': [-6, 6]
}

dos_settings = {
    'mode': 'parametric',
    'dirname': dos_dir,
    'atoms': [0, 1],
    'orbitals': [4, 5, 6, 7, 8],
    'spins': [0, 1],
    'elimit': [-6, 6],
    'clim': [0, 1]
}

pyprocar.bandsdosplot(
    code='qe',
    bands_settings=bands_settings,
    dos_settings=dos_settings
)
```

Listing 6: Example code for plotting both band structures and density of states (DOS) using PyProcar and Quantum ESPRESSO.

4.7. Simultaneous Comparison of Band Structures from Different DFT Codes

A unique capability of PyProcar is the option to compare multiple band structures within the same plot. This is particularly useful for assessing similarities or discrepancies between calculations performed with different Density Functional Theory codes. This feature is made

possible by PyProcar's support for reusing the `matplotlib.Axes` object across multiple *bandsplot* calls, thereby providing a unified axis for comparison.

To demonstrate, the following example contrasts the band structures of $SrVO_3$ as calculated using VASP and Quantum Espresso. In this case, the 'mode' is set to 'parametric' for the VASP data and 'plain' for the Quantum Espresso data.

```
import pyprocar
import os

# Plot bands using VASP data and store the plot object
ebs_plot = pyprocar.bandsplot(
    code='vasp',
    dirname=vasp_data_dir,
    mode='parametric',
    elimit=[-10, 10],
    orbitals=[4, 5, 6, 7, 8],
    show=False
)

# Overlay bands using Quantum ESPRESSO data on the same plot
pyprocar.bandsplot(
    code='qe',
    dirname=qe_data_dir,
    mode='plain',
    elimit=[-10, 10],
    color='k',
    ax=ebs_plot.ax,
    show=False,
    savefig=f"{figure_dir}{os.sep}comparing_bands.pdf"
)
```

Listing 7: Example code for comparing band structures generated by VASP and Quantum ESPRESSO using PyProcar.

4.8. 2D Fermi Surface and Projected Fermi Surface Plotting

PyProcar introduces advanced capabilities for plotting 2D cross-sections of the Fermi surface, offering a powerful tool for understanding electronic properties near the Fermi level. This is crucial for predicting material behavior, including transport properties and applications in various scientific domains. To illustrate, Figure 9 showcases the k_z=0 plane of Iron.

The feature is accessed through the *fermi2D* function, which accepts various modes for detailed visualization. Listing 8 demonstrates how to use this feature with 'plain' mode for Iron.

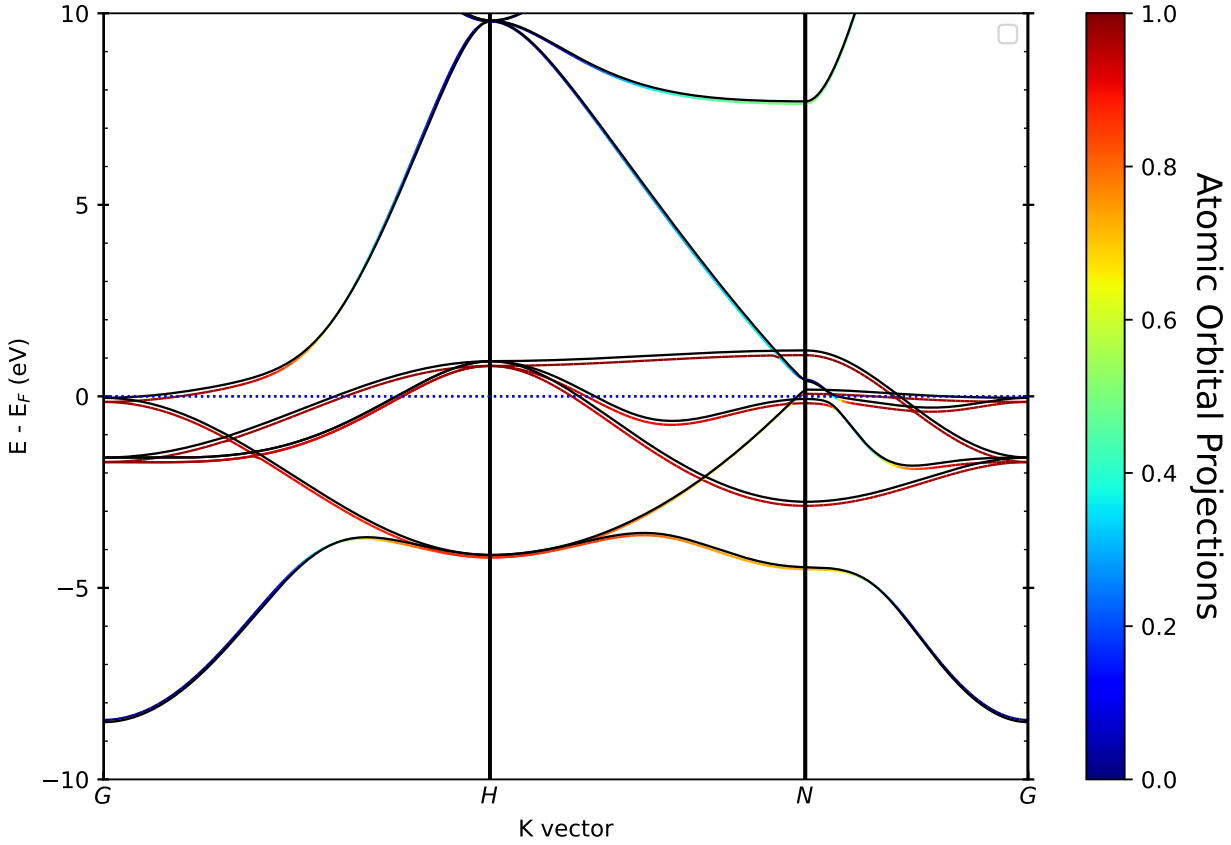


Figure 8: Comparison of $SrVO_3$ band structures computed using Quantum Espresso and VASP. While the VASP results are represented in parametric mode to show orbital contributions, the Quantum Espresso data is depicted in plain mode, rendered in black for clarity.

```
import pyprocar
import os

# Define the data directory path
data_dir = os.path.join(PROJECT_DIR, 'data', 'examples', 'Fe', 'qe', 'spin
    -polarized-colinear', 'Fermi')

# Generate a 2D Fermi surface plot
pyprocar.Fermi2D(
    code='qe',
    dirname=data_dir,
    mode='plain'
)
```

Listing 8: Example code for generating a 2D Fermi surface plot using PyProcar and Quantum ESPRESSO.

4.9. Visualizing 3D and Projected Fermi Surfaces with PyProcar

PyProcar has enhanced its plotting capabilities of the 3D Fermi surface. This advancement enhances not only visualization but also enables in-depth analysis of electronic behavior near the Fermi level. Such insights are crucial for predicting material properties like transport phenomena and potential real-world applications. In the subsections that follow, we will showcase the extensive features of this new addition, focusing particularly on iron as a case study.

Iron serves as an exemplary subject for this demonstration due to its unique electronic structure. As a transition metal, iron's valence electrons populate d orbitals, significantly influencing its material characteristics.

To access these advanced plotting features, one must initialize the 'FermiHandler' object. Doing so will fetch relevant data from the specified calculation directory, allowing you to employ various class methods for Fermi surface plotting.
Example usage to initialize the 'FermiHandler' object is as follows:

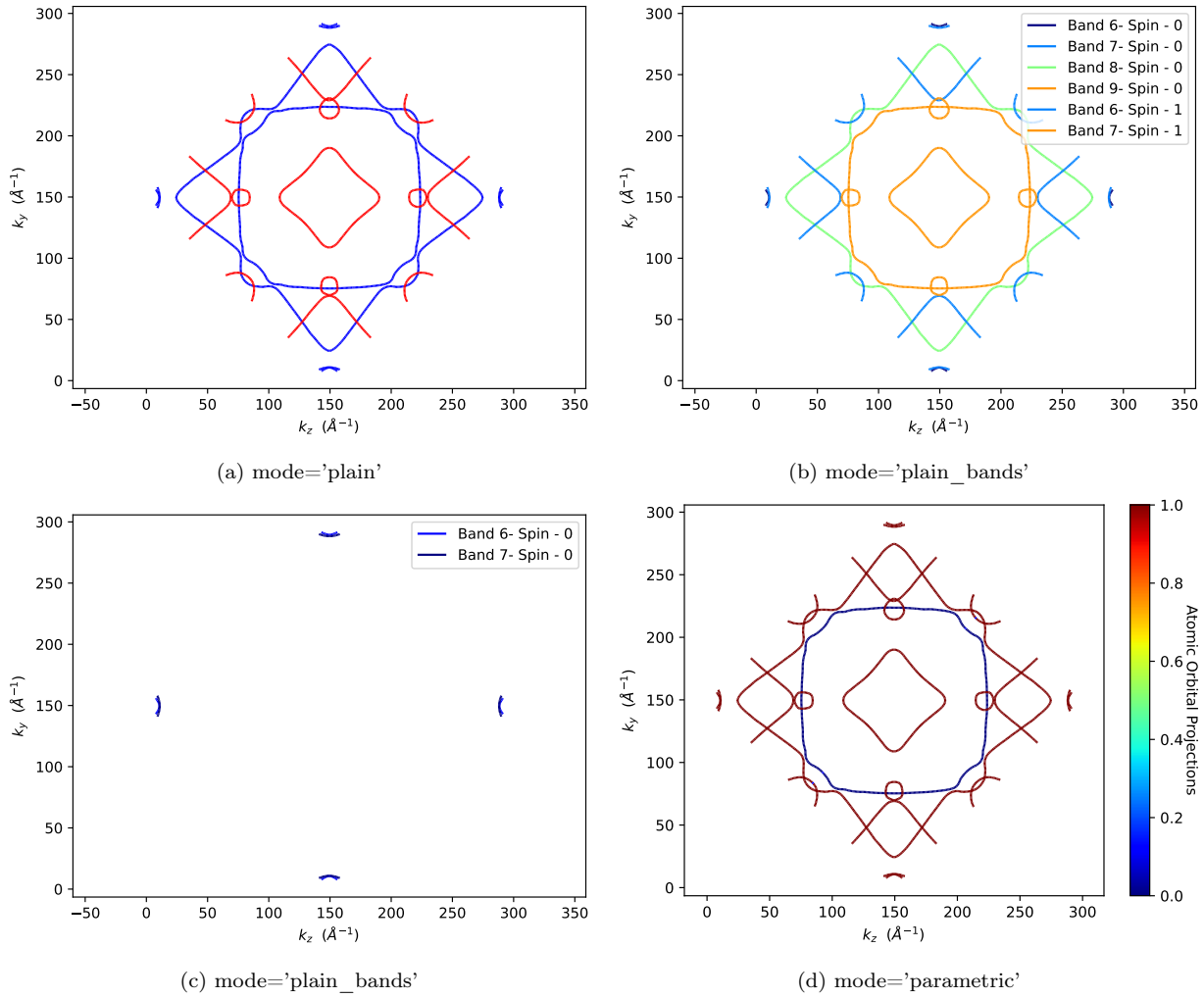
```
FermiHandler = pyprocar.FermiHandler(
    code="qe",
    dirname=data_dir,
    apply_symmetry=True)
```

Listing 9: Example code for initializing the FermiHandler object using PyProcar and Quantum ESPRESSO.

4.9.1. plot_fermi_surface

The plot_fermi_surface function is key to visualizing the intricacies of iron's Fermi surface within the first Brillouin zone. The mode argument allows for the setting of different modes of operation for visualizing the Fermi surface.

By setting **mode='plain'**, we obtain a visualization of the Fermi surface (Figure 10a) where colors differentiate the bands that compose the surface. This figure offers a detailed depiction of the band structure of iron at the Fermi level, with each color corresponding to a different band. Here, the bands include both spin-polarized bands. The **bands** argument enables the selection of specific bands to be plotted. We also use the keyword argument **extended_zone_directions** to extend the Fermi surface along the reciprocal axis.



(a) mode='plain'

(b) mode='plain_bands'

(c) mode='plain_bands'

(d) mode='parametric'

Figure 9: Various modes for 2D Fermi surface visualization in Iron. *plain* mode shows the Fermi surface in uniform color, differentiating spin channels in blue and red. *plain_bands* mode adds band and spin channel coloration. Subfigure c demonstrates the option to display only the first spin channel. In *parametric* mode (Subfigure d), the plot highlights contributions from specific atomic orbitals, shown here for the d-orbitals of Fe across both spin channels.

Figure 10b is generated when **mode='parametric'** is used. This mode plots the orbital-resolved projections onto the Fermi surface, providing an in-depth look at the contribution of specific orbitals to the Fermi surface. For iron, the d orbitals are particularly important and are, therefore, our focus. To specify the d orbitals for projection, additional keyword arguments are used: **atoms**, **orbitals**, and **spins**. In this plot, we look at one of the spin-polarized bands (**spins=[1]**).

The below snippet provides an example of how to generate a 'plain' Fermi surface plot for iron using the **plot_fermi_surface** function:

```
FermiHandler.plot_fermi_surface(mode="plain",
    spins=[1],
    extended_zone_directions=[[0,0,1],[0,1,0]],
    show=True,)
```

Listing 10: Example code for plotting the 'plain' fermi surface using PyProcar and Quantum ESPRESSO.

4.9.2. plot_fermi_isoslider

In our continued exploration of iron's electronic structure, the plot_Fermi_isoslider function plays a pivotal role. This function equips users with the ability to interactively navigate between various isosurfaces around the Fermi level interactively, providing a comprehensive view of the material's electronic landscape.

As illustrated in Figure 10d 10e 10c, the iso-slider enables the visualization of iron's electronic states within an energy range of interest around the Fermi level. The advantage of this dynamic representation is two-fold. First, it provides an in-depth understanding of the electronic structure and transport properties of iron, providing clues to potential applications. Second, it enables the creation of animated GIFs, which provide a fluid, real-time exploration of the electronic state shifts with the varying Fermi level, further enhancing our understanding of the material's properties.

The **plot_fermi_isoslider** method takes in the arguments **iso_range** and **iso_surfaces**. The **iso_range** sets the energy range around the Fermi level (in eV), while **iso_surfaces** determines the number of surfaces to be generated within this energy range.

Below is an example of calling the **plot_fermi_isoslider** function, which will generate an iso-slider for iron with an energy range of 2eV around the Fermi level and display five different isosurfaces:

```
fermiHandler.plot_fermi_isoslider(
    iso_values=[5.5989-0.1,5.5989,5.5989+0.1],
    spins=[0,1],
    spin_colors=['red','blue'],
    mode="plain",
    show=True)
```

Listing 11: Example code for plotting the fermi surface with an interactive isoslider using PyProcar.

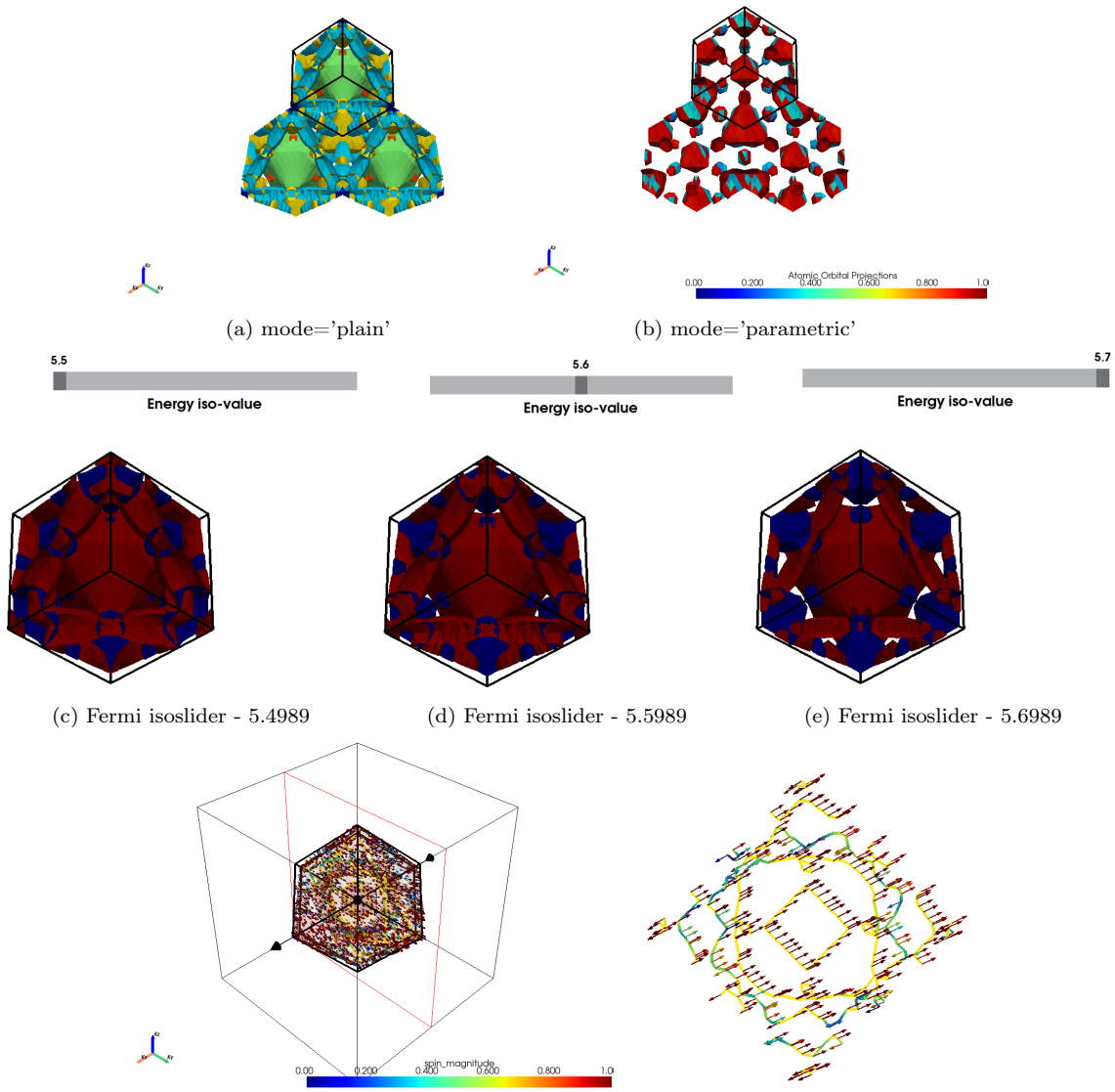
4.9.3. create_isovalue_gif

The create_isovalue_gif method allows the user to create a gif animation of the surfaces around the Fermi Level. Like in the isoslider, takes as arguments **iso_range** and **iso_surface** which sets the energy range in eV and the number of surfaces around the Fermi level generated. In addition, instead of specifying the range and the number of surfaces, the argument **iso_values** can be used to generate surfaces at exact energy values.

Below is an example of how this feature can be used:

```
FermiHandler.create_isovalue_gif(
    iso_values=[5.5989-0.1,5.5989,5.5989+0.1],
    spins=[0,1],
    spin_colors=['red','blue'],
    mode="plain")
```

Listing 12: Example code for plotting the fermi surface as a gif using PyProcar.



(a) mode='plain' (b) mode='parametric'

(c) Fermi isoslider - 5.4989 (d) Fermi isoslider - 5.5989 (e) Fermi isoslider - 5.6989

(f) mode='plain', cross_section (g) mode='plain', cross_section_slice

Figure 10: Demonstration of 3D Fermi surface plotting for Iron. Here, the subfigures demonstrate the functionalities of the 2D Fermi surface plotting. Subfigure a, *plain* mode plots the bands of the 3D Fermi surface for both spin channels by color. Subfigure b, *parametric* mode shows the contribution of the d-orbitals of iron on the Fermi surface. In subfigures a and b, both surfaces are extended to the next zone, given a direction vector. Subfigures c, d, and e show how to use the Fermi_isoslider. Here, we choose to generate three surfaces around Fermi energy. (e)-(E_{Fermi}?-0.1) (d)-(E_{Fermi}?), (e)-(E_{Fermi}?+0.1). The colors here represent the spin channels. Subfigures f and g show the cross-section widget used on the Fermi surface. In this case, we are slicing the Fermi surface of Iron for a non-colinear calculation. The arrows represent the magnitude of the spin textures. Subfigure f shows how the widget looks. The red outline plane can be interacted with to slice the Fermi surface. Subfigure g will be the 2d slice saved once the render window is closed.

4.9.4. plot_fermi_cross_section

The **plot_fermi_cross_section** method provides an interactive interface for users to explore the Fermi surface, particularly useful when studying materials like iron, where spin properties play a significant role. This interactive cross-section widget allows users to create cross-sections of the Fermi surface in real time, offering an in-depth examination of the material's electronic structure.

One notable feature of this method is the option to display spin textures on the cross-sectional slice. This is particularly useful for studying materials with intricate spin characteristics, such as iron. The **spin_texture=True** argument activates this feature, while the **arrow_size** parameter allows control over the visualization of the spin texture.

Below is an example of how this feature can be used:

```
FermiHandler.plot_fermi_cross_section(
  slice_normal=(1,0,0),
  slice_origin=(0,0,0),
  cross_section_slice_linewidth=5.0,
  mode="spin_texture",
  spin_texture=True,
  texture_size=0.25,
  surface_opacity=0.10,
  surface_clim=[0,1],
  show=True)
```

Listing 13: Example code plotting the fermi surface with a cross section widget using PyProcar.

This example creates a cross-section of the Fermi surface with a normal in the z-direction, originating at the origin of the plot. It then plots the resulting cross-section with a spin texture overlay, with each arrow's size set to 0.5.

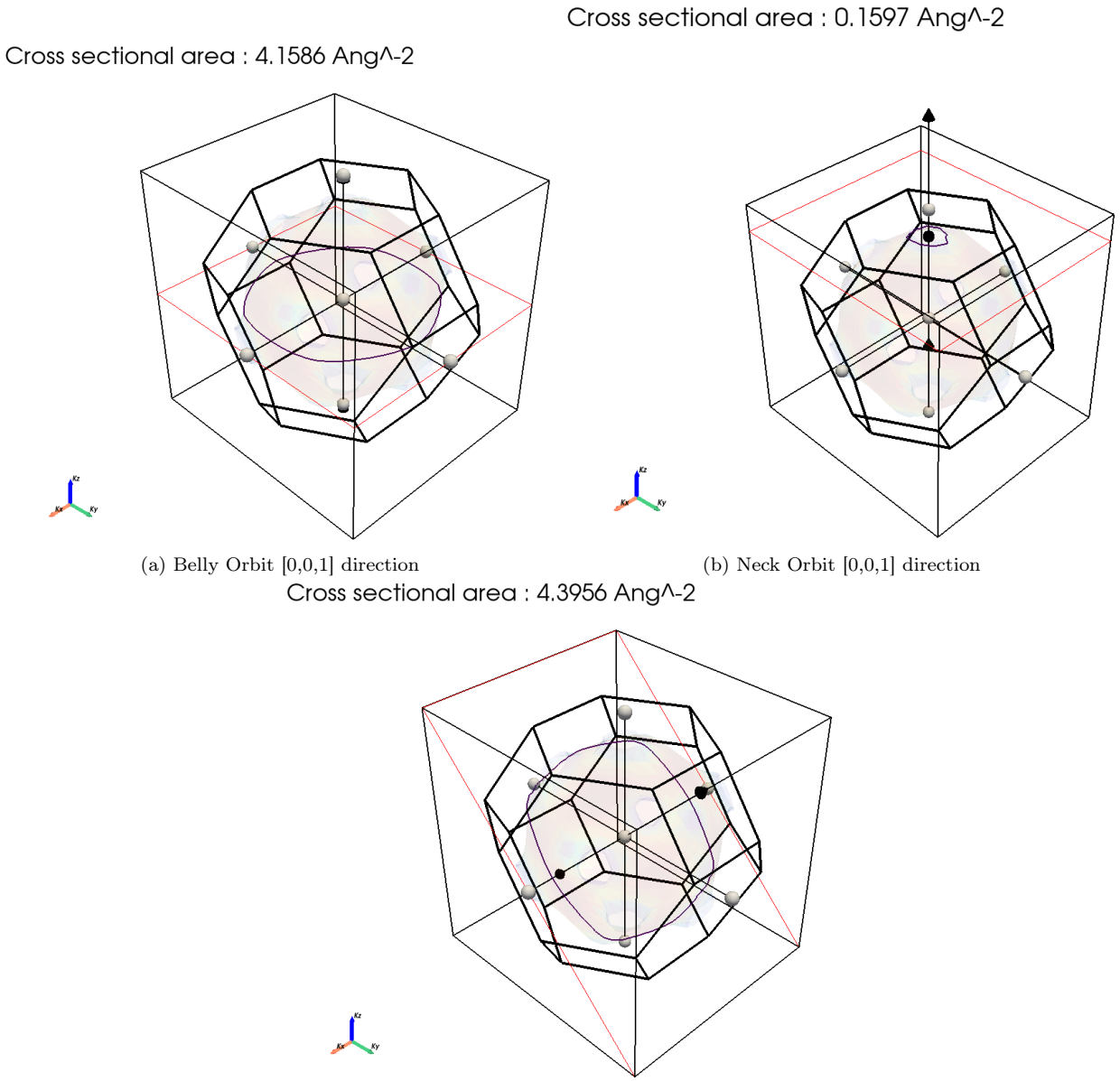
In Figure 10f, we demonstrate the cross-section widget. The first subplot (Figure 10f) shows the widget in its initial state, while the second subplot (Figure 10g) shows a cross-sectional slice of the Fermi surface with spin texture. This live and interactive feature can significantly enhance the understanding of the material's electronic and spin properties.

4.9.5. plot_fermi_cross_section_box_widget

The method **plot_fermi_cross_section_box_widget** operates in a similar vein to its predecessor, **plot_fermi_cross_section**, but provides a supplementary feature - a box clipping widget. This enhancement allows users to inspect discrete slices of the Fermi surface. An additional attribute, **show_cross_section_area**, facilitates the acquisition of the largest cross-sectional area of the inspected slice. This feature is particularly invaluable in discerning de Haas-van Alphen frequencies, which serve as a tool for corroborating experimental findings and deepening the comprehension of material characteristics at the quantum scale.

Figure 11 demonstrates the utility of the tool in identifying extremal cross-section areas of the Fermi surface. In his paper in 1952, Onsager showed that the frequencies of these oscillations were determined by the following simple relationship. These areas have a direct relationship with the Van Alphen frequencies, which the formula can be written as :

$$F = \frac{c\hbar A_e}{2\pi e} \quad \text{(cgs)} \quad (2)$$



(a) Belly Orbit [0,0,1] direction

(b) Neck Orbit [0,0,1] direction

(c) Belly Orbit [0,1,1] direction

Figure 11: Van Alphen extremal cross section areas for Gold.

, where the units of this equation of given in centimetre–gram–second (cgs) system of units. c, $\hbar$, and e are the speed of light, Plank's constant, and the electron charge, respectively. A_e is the extremal cross section Area of the fermi surface in a plane normal to the direction of the magnetic field. These extremal cross sections are what we measure below for gold. Table 1 catalogs the frequencies predicted using this formula. Here, we juxtapose these predicted frequencies with the experimental values extracted from the paper by David Shoenberg [43]. Shoenberg calculates the Fermi surfaces of copper, silver, and gold using the Van Alphen Effect in his work.

This comparison effectively validates the utility of our tool in predicting and comparing Van Alphen frequencies, thereby providing a reliable method to enhance our understanding

Orbit Type	A_e (Å^{-2})	F (G)	F$_{exp}$ (G)
Belly 001	4.1586	4.365*10^8	4.50*10^8
Neck 001	0.1597	1.68*10^7	1.50*10^7
Belly 011	4.3956	4.61*10^8	4.85*10^8

Table 1: Comparison of experimental and predicted Van Alphen frequencies for Gold

of quantum material properties. Here is the example code used to generate the slice shown below.

```
FermiHandler.plot_fermi_cross_section_box_widget(
      slice_normal=[0,0,1],
      slice_origin=[0,0,0],
      surface_opacity=0.05,
      add_scalar_bar=False,
      cross_section_slice_linewidth=False,
      cross_section_slice_show_area=True,
      mode="parametric",
      show=True,)
```

Listing 14: Example code plotting a fermi surface with cross section and box widget with PyProcar

4.10. 2D Bandstructure Plotting

Pyprocar now supports the generation of 2D band structure plots, providing a clear and concise representation of the electronic structure of materials. This feature lets users visualize and analyze the relationship between the electronic bands and the underlying crystal structure.

The key novelty of the 2D band structure plotting feature lies in its ability to elucidate the complex relationship between the electronic bands and the underlying crystal structure visually intuitively. Such a feature is particularly useful for investigating and understanding intricate material behaviors such as anisotropy, where material properties depend on the direction of interest.

To access the 2D band structure plotting feature, users should utilize the BandStructure2DHandler object, a central component of our package's updated functionality. This will load the data from the calculation directory. The plotting can then be done by by calling plot_band_structure. Below is how we initiate the BandStructure2DHandler object. Usage:

```
handler = pyprocar.BandStructure2DHandler(
    code="qe",
    dirname=data_dir,
    apply_symmetry=False)

atoms=[0]
orbitals=[4,5,6,7,8]
handler.plot_band_structure(mode='spin_texture',
   spin_texture=True,
   atoms=atoms,
   orbitals=orbitals,
   surface_clim=[-1,1])
```

Listing 15: Example code for initializing the BandStructure2D object using PyProcar and Quantum ESPRESSO, and then plotting the spin texture band strucure.

In this code snippet, the code parameter refers to the code used in performing the Density Functional Theory (DFT) calculations, and **dirname** is the directory where the data of these calculations reside. The **apply_symmetry** parameter allows users to decide whether or not to apply symmetry to the band structure plots, which can provide enhanced visual clarity and aid in identifying particular patterns.

The **plot_band_structure** method is responsible for generating the actual 2D band structure plot. The **mode** parameter defines the plot mode; 'spin_texture' mode has been used in this instance, illustrating the bands' spin texture. The **spin_texture** parameter activates the depiction of spin texture, a vital tool in spintronics and quantum computing research. The atoms and orbitals parameters allow users to select specific atoms and orbitals for the plot, providing flexibility and versatility. The **save_2d** parameter is where you can specify the name and location of the saved plot. Lastly, **surface_clim** allows for the adjustment of the color scale range on the plot.

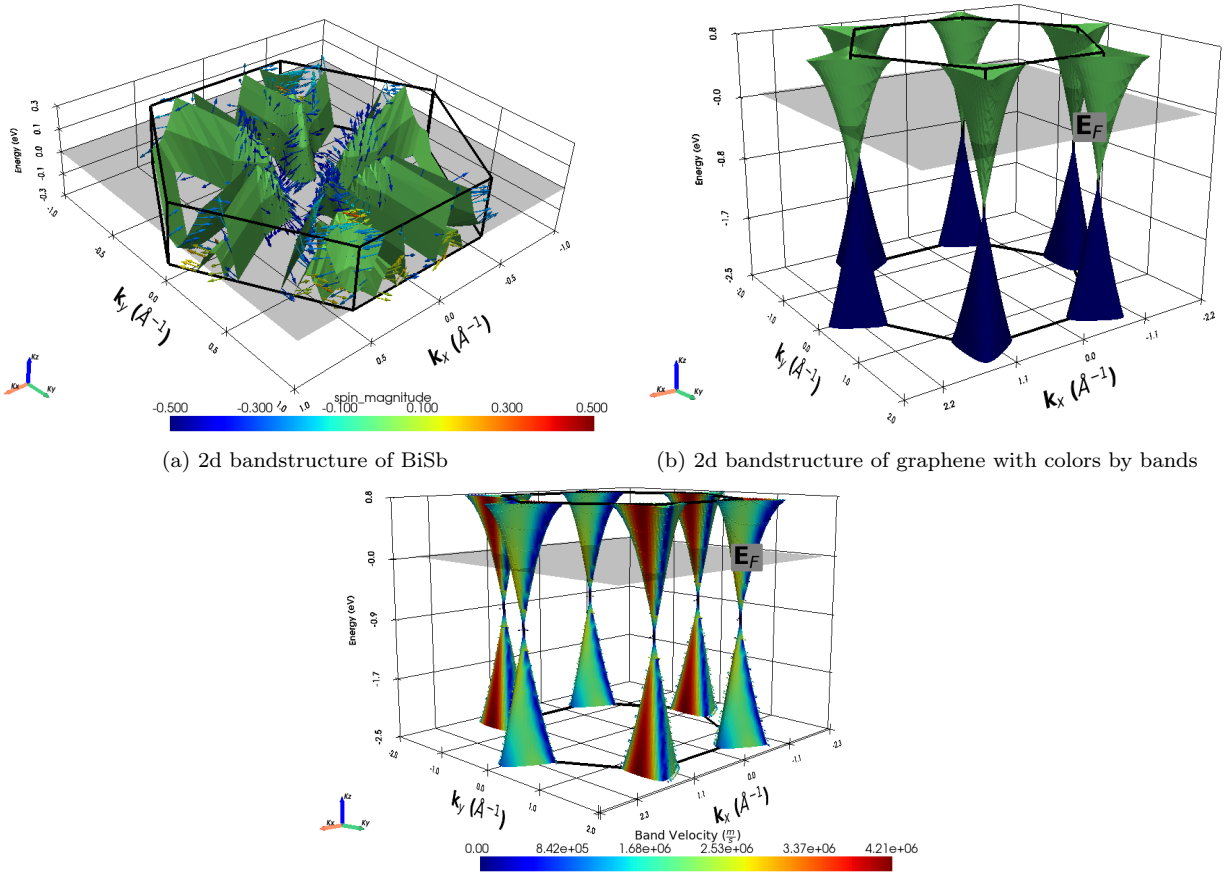
As seen in Figure 12, the implementation of 2D band structure plotting has proven to be highly effective in visualizing the electronic structure of two sample materials: a BiSb monolayer and graphene.

These sample plots demonstrate the effectiveness of the feature in providing a clear and detailed overview of the electronic structures of the materials. As such, the new 2D band structure plotting capability is a significant upgrade to our package, offering users a powerful tool for the efficient visualization and analysis of electronic band structures.

4.11. Derivative Calculation of Band Energies

PyProcar has recently been enriched with a method for computing the first and second derivatives of band energies concerning the k-mesh. This feature allows users to discern better and analyze crucial band structure attributes, including band gaps, band curvatures, and effective masses. As a result, it offers valuable insights into the electronic properties of materials, marking a significant stride in our understanding of material science.

This enhancement propels PyProcar further ahead as a robust and versatile tool in the visualization and analysis of Density Functional Theory (DFT) calculations in material science research. By integrating these capabilities, the package caters to a broader range of research applications and promotes a more comprehensive understanding of materials' electronic structure and properties.



(a) 2d bandstructure of BiSb

(b) 2d bandstructure of graphene with colors by bands

(c) 2d bandstructure of graphene with band velocity projections

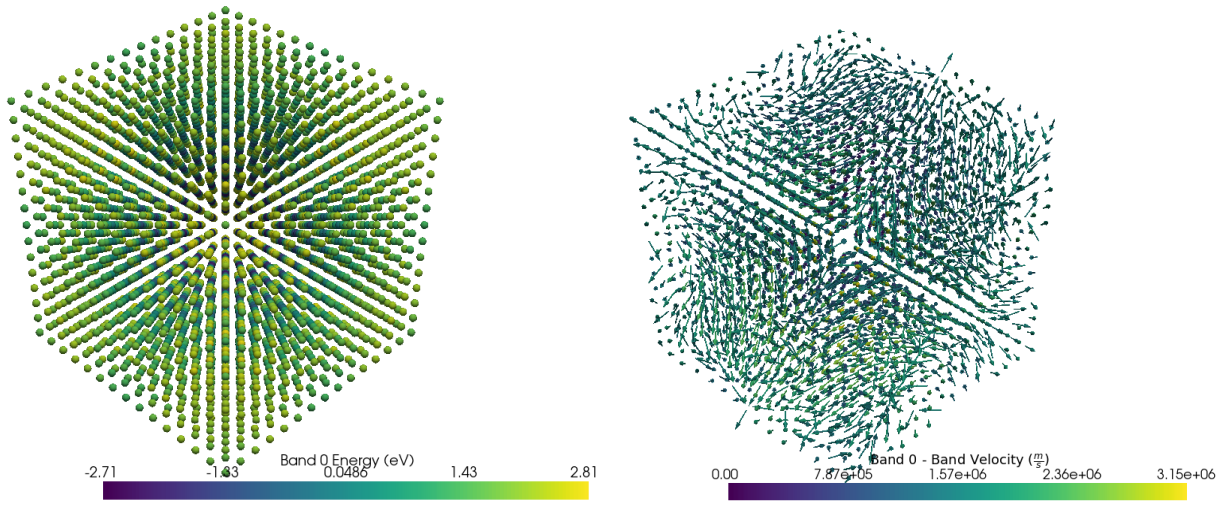
Figure 12: Demonstration of plotting of the 2d bandstructure for Graphene and monolayer BiSb

To clearly depict these newly incorporated features, refer to Figure 13. The first subplot, Figure 13a, presents the bands on the k-points mesh. This visualization of the first band reveals the shape and distribution of energy bands in the material's Brillouin zone.

In the second subplot, Figure 13b, band gradients, represented by arrows, are superimposed on the k-points mesh. Each arrow depicts the direction and magnitude of the band velocity at a given k-point, providing a powerful visual tool for understanding the variations in band energies across the Brillouin zone.

4.12. PyPoscar

PyPoscar is a specialized subpackage within PyProcar, designed to offer an extensive toolkit for the manipulation and analysis of crystal geometry files. More importantly, it also offers advanced features as defect recognition. Here, PyPoscar employs sophisticated algorithms to identify defects within the crystal lattice, offering valuable insights into material properties, or similarly, it can also perform cluster identification, where PyPoscar can also discern disjoint clusters, whether they exist in the original lattice—such as van der Waals layers—or among defect atoms. The identification of defects, surfaces, and other anomalies is achieved through a series of tests that examine the local environment of each atom. Atoms



(a) Bands on k-points mesh

(b) Band Gradients on k-points mesh

Figure 13: Comprehensive Visualization of Band Structures using the ElectronicBandStructure Class in PyProcar: This figure showcases two aspects of band structure analysis facilitated by PyProcar. Subfigure (a) 13a presents the electronic band energies plotted across the k-points mesh, offering a clear view of the energy distribution and band dispersion in the material's Brillouin zone. (b) 13b extends this analysis by overlaying the band velocity, represented as arrows, onto the same k-points mesh. Each arrow indicates the direction and magnitude of the band velocity at specific k-points.

exhibiting statistically significant deviations are classified as defects. This comparison is facilitated by machine learning methods from the scikit-learn package[26]. A pivotal aspect of our methodology is the identification of neighboring atoms. This is accomplished using the radial distribution function, $g(r)$, in conjunction with a table of maximum values for specific interactions. Kernel density estimation[44] of $g(r)$ -an algorithm to estimate the probability density of a function- is employed to cluster similar values without the need for binning (*e.g.* the layer structure of van der Waals can be identified, for instance the quintuple layers of Bi_2Se_3 are recognized as different clusters) or among the defect atoms (*e.g.* the both surfaces of a slab belong to different clusters). The identification of defects, clusters, and other features is invoked through the pyprocar.autobandsplot function (see Section 4.4). The results, including a list of special atoms and their associated electronic states, are documented in a report.txt file.

PyPoscar extends its utility beyond basic functionalities by incorporating advanced descriptors rooted in the crystal geometry of materials. A notable example of such a descriptor is the Global Connectivity (Ω), a quantifiable measure reflecting the comprehensive bonding network within a material. This descriptor is particularly significant because it has been demonstrated, as cited in ref. [27], to offer accurate predictions regarding the material's reactivity, especially in hydrogen intercalation processes. One of the key advantages of using Ω is its independence from Density Functional Theory (DFT) calculations. This implementation is a substantial benefit, as DFT calculations are typically resource-intensive and complex. By bypassing the need for these calculations, Ω provides a more efficient yet reliable method for assessing material reactivity, making it an invaluable tool in materials science and computational chemistry. This capability of PyPoscar to leverage such descriptors enhances its

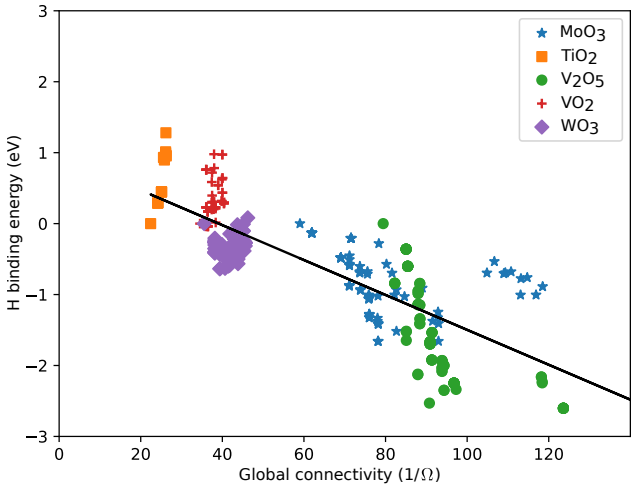


Figure 14: Binding energy of H in metal oxides as a function of the global connectivity (Ω). Each point corresponds to different H concentrations and absorption sites. For details see Ref. [27].

applicability and effectiveness in the study and prediction of material properties, particularly in scenarios where quick and accurate assessments are crucial. The usage is:

```
from pyprocar.pyposcar import globalConectivity
from pyprocar.pyposcar import poscar

# Nearest neighbors cutoff distances
custom_distances = {"Mo-O":2.59 , "H-O":1.2, "H-Mo": 2.01}

POSCAR = poscar.Poscar('POSCAR_MoO3')
POSCAR.parse()
gc = globalConectivity(POSCAR= POSCAR, custom_nn_dist=custom_distances)
print("Global conectivity", gc.GC)
```

Listing 16: Example code for the global connectivity.

This study analyzes the same metal oxides cited in ref. [27]. Our primary objective was calculating the Global Connectivity (Ω) for these oxides to demonstrate the efficiency of using PyProcar's advanced capabilities. Miu *et al.* integrated these Ω values with the binding energies of hydrogen atoms derived from meticulous Density Functional Theory (DFT) calculations. This integration was pivotal in establishing a clear and quantifiable correlation between the Global Connectivity of the metal oxides and their affinity for hydrogen atom binding. The significance of this correlation is visually represented in Fig. 14. This figure is not merely an illustration but a crucial piece of evidence that underscores the relationship between the structural properties of metal oxides (as denoted by Ω) and their chemical reactivity towards hydrogen atoms. The correlation depicted in the figure provides valuable insights into the fundamental interactions at play, offering a more nuanced understanding of the material properties. Combining PyProcar's computational efficiency in determining Ω with the precision of DFT calculations for hydrogen binding energies, this approach exemplifies the power of integrating different computational methods. It highlights how such synergistic use of computational tools can lead to a deeper understanding of material properties, essential in advancing materials science and chemistry.

5. Outlook

In the ongoing development of PyProcar, our focus is on refining the software to enhance its usability and functionality. A key aspect of this process involves regular code refactoring, ensuring that PyProcar remains efficient and accessible for both users and developers. We are also planning to collaborate with developers of various DFT codes to enable compatibility with PyProcar's features. For example, we want to extend our service to users of the package Lobster [45] used for chemical-bonding analysis in periodic systems. For instance, facilitating band unfolding in other DFT codes, similar to what is currently achievable in VASP, is a significant goal. This requires these codes to output the phase of the projections, a capability we intend to work on collaboratively. Another important development is incorporating support for VASP's PROCAR files in HDF5 format, which will be instrumental in handling large-scale DFT calculations more effectively. Furthermore, we aim to enhance the functionalities of our existing plotters, including bandsplot, dosplot, fermi3d, bands_2d, and fermi_2d, to provide users with advanced tools for material property and electronic structure analysis. In addition, we will continue to develop PyPoscar, focusing on improving its capability in POSCAR file manipulation for applications such as defect recognition and structural analysis. For instance, the recent integration of the global connectivity descriptor into PyPoscar represents a significant advancement, illustrating how innovative different geometrical analysis techniques can be incorporated into this software. This enhancement is particularly beneficial for conducting more comprehensive structural analyses in PyPoscar before delving into electronic analysis. By leveraging these new geometrical analysis capabilities, researchers can gain deeper insights into structural properties, which is crucial for a more accurate and thorough understanding of the material's behavior at the electronic level. This development not only broadens the scope of PyPoscar's applications but also underscores the ongoing evolution in computational tools used in materials science, paving the way for more sophisticated and precise research methodologies. These development initiatives are aligned with our commitment to keeping PyProcar a versatile and valuable tool in the field of DFT analysis and visualization.

6. Conclusion

This manuscript presents a thorough update on the improvements to the PyProcar package and its ecosystem, rendering it more maintainable, versatile, and scalable. The architectural improvements, alongside the centralized repository and testing framework introduction, have fortified the package's backbone, creating a foundation that promises a user-friendly and developer-friendly experience.

The extension in support of a broader range of DFT codes marks an important milestone in enhancing PyProcar's adaptability. Our enriched documentation further simplifies user interaction and provides valuable resources for those looking to get the most out of PyProcar's extensive feature set. Additionally, incorporating new features such as band unfolding, noncollinear calculation support, density of states, projected density of states, Fermi surface plotting, and 2D bandstructure plotting not only diversifies its applicability but also deepens its analytic capabilities. The recent introduction of derivative calculations of band energies marks a substantial advance, offering insightful analysis into material electronic properties

like band gaps, effective masses, and band curvatures. Expanding the PyProcar ecosystem to include PyPoscar, a specialized toolkit for manipulating and analyzing POSCAR files, brings added functionality and integrative capabilities. With features such as coordinate system conversion, structural generation, and property calculations, PyPoscar establishes itself as an indispensable tool in computational material science, especially in conjunction with PyProcar.

7. Acknowledgments

This was supported by the U.S. Department of Energy (DOE), Office of Science, Basic Energy Sciences (BES), under Award DE-SC0021375. This work used Bridges2 and Expanse at the Pittsburgh Supercomputer and the San Diego Supercomputer Center through allocation DMR140031 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which National Science Foundation supports grants 2138259, 2138286, 2138307, 2137603, and 2138296. Some of the computational resources were provided by the WVU Research Computing Dolly Sods HPC cluster, which is funded in part by NSF OAC-2117575. E.B. and X. H. acknowledge the FNRS and the Excellence of Science program (EOS "ShapeME", No. 40007525) funded by the FWO and F.R.S.-FNRS (theory and algorithm development) and the CECI supercomputer facilities funded by the F.R.S-FNRS (Grant No. 2.5020.1) and the Tier-1 supercomputer of the Fédération Wallonie-Bruxelles funded by the Walloon Region (Grant No. 1117545). This work was also partially supported by Fondecyt Grants No. 1231487 and 1220715, by the Center for the Development of Nanoscience and Nanotechnology CEDENNA AFB220001, from Conicyt PIA/Anillo ACT192023 and by the supercomputing infrastructure of the NLHPC (ECM-02).

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve the clarity of the writing. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] X. Gonze, F. Jollet, F. A. Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk, et al., Recent developments in the abinit software package, Comput. Phys. Commun. 205 (2016) 106–131.

[2] X. Gonze, B. Amadon, G. Antonius, F. Arnardi, L. Baguet, J.-M. Beuken, J. Bieder, F. Bottin, J. Bouchet, E. Bousquet, et al., The abinit project: Impact, environment and recent developments, Computer Physics Communications 248 (2020) 107042.

[3] A. H. Romero, D. C. Allan, B. Amadon, G. Antonius, T. Applencourt, L. Baguet, J. Bieder, F. Bottin, J. Bouchet, E. Bousquet, F. Bruneval, G. Brunin, D. Caliste, M. Côté, J. Denier, C. Dreyer, P. Ghosez, M. Giantomassi, Y. Gillet, O. Gingras, D. R. Hamann, G. Hautier, F. Jollet, G. Jomard, A. Martin, H. P. C. Miranda, F. Naccarato, G. Petretto, N. A. Pike, V. Planes, S. Prokhorenko, T. Rangel, F. Ricci, G.-M. Rignanese, M. Royo, M. Stengel, M. Torrent, M. J. van Setten, B. Van Troeye,

M. J. Verstraete, J. Wiktor, J. W. Zwanziger, X. Gonze, Abinit: Overview and focus on selected capabilities, The Journal of Chemical Physics 152 (12) (2020) 124102. doi:10.1063/1.5144261.
URL https://doi.org/10.1063/1.5144261

[4] G. Kresse, J. Hafner, Ab initio molecular dynamics for liquid metals, Phys. Rev. B 47 (1993) 558–561.

[5] G. Kresse, J. Hafner, Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium, Phys. Rev. B 49 (1994) 14251–14269.

[6] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6 (1) (1996) 15 – 50.

[7] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169–11186.

[8] A. García, N. Papior, A. Akhtar, E. Artacho, V. Blum, E. Bosoni, P. Brandimarte, M. Brandbyge, J. I. Cerdá, F. Corsetti, et al., Siesta: Recent developments and applications, The Journal of chemical physics 152 (20) (2020).

[9] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. D. Corso, S. de Gironcoli, P. Delugas, R. A. D. Jr, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. O. de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, S. Baroni, Advanced capabilities for materials modelling with quantum espresso, Journal of Physics: Condensed Matter 29 (46) (2017) 465901.
URL http://stacks.iop.org/0953-8984/29/i=46/a=465901

[10] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, Quantum espresso: a modular and open-source software project for quantum simulations of materials, Journal of Physics: Condensed Matter 21 (39) (2009) 395502 (19pp).
URL http://www.quantum-espresso.org

[11] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, G. Ceder, Python materials genomics (pymatgen): A robust, open-source python library for materials analysis, Computational Materials Science 68

(2013) 314–319. doi:https://doi.org/10.1016/j.commatsci.2012.10.028.
URL https://www.sciencedirect.com/science/article/pii/S0927025612006295

[12] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, B. Kozinsky, Aiida: automated interactive infrastructure and database for computational science, Computational Materials Science 111 (2016) 218–230.

[13] S. Curtarolo, W. Setyawan, G. L. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, et al., Aflow: An automatic framework for high-throughput materials discovery, Computational Materials Science 58 (2012) 218–226.

[14] V. Wang, N. Xu, J.-C. Liu, G. Tang, W.-T. Geng, Vaspkit: A user-friendly interface facilitating high-throughput computing and analysis using vasp code, Computer Physics Communications 267 (2021) 108033.

[15] A. M. Ganose, A. Searle, A. Jain, S. M. Griffin, Ifermi: A python library for fermi surface generation and analysis, Journal of Open Source Software 6 (59) (2021) 3089.

[16] M. J. Rutter, C2x: A tool for visualisation and input preparation for castep and other electronic structure codes, Computer Physics Communications 225 (2018) 174–179.

[17] K. Boguslawski, A. Leszczyk, A. Nowak, F. Brzęk, P. S. Żuchowski, D. Kędziera, P. Tecmer, Pythonic black-box electronic structure tool (pybest). an open-source python platform for electronic structure calculations at the interface between chemistry and physics, Computer Physics Communications 264 (2021) 107933.

[18] K. Choudhary, K. F. Garrity, A. C. Reid, B. DeCost, A. J. Biacchi, A. R. Hight Walker, Z. Trautt, J. Hattrick-Simpers, A. G. Kusne, A. Centrone, et al., The joint automated repository for various integrated simulations (jarvis) for data-driven materials design, npj computational materials 6 (1) (2020) 173.

[19] K. Choudhary, B. DeCost, Atomistic line graph neural network for improved materials property predictions, npj Computational Materials 7 (1) (2021) 185.

[20] Elk, http://elk.sourceforge.net/, accessed: 11/26/2023 (2023).

[21] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Deshaye, T. Dumitrică, A. Dominguez, et al., Dftb+, a software package for efficient approximate density functional theory based atomistic simulations, The Journal of chemical physics 152 (12) (2020).

[22] S. B. Dugdale, Life on the edge: a beginner's guide to the fermi surface, Physica Scripta 91 (5) (2016) 053009. doi:10.1088/0031-8949/91/5/053009.
URL https://dx.doi.org/10.1088/0031-8949/91/5/053009

[23] G. Petretto, abipy, https://github.com/abinit/abipy (2013).

[24] A. Kokalj, Xcrysden—a new program for displaying crystalline structures and electron densities, Journal of Molecular Graphics and Modelling 17 (3) (1999) 176–179. doi:https://doi.org/10.1016/S1093-3263(99)00028-5.
URL https://www.sciencedirect.com/science/article/pii/S1093326399000285

[25] U. Herath, P. Tavadze, X. He, E. Bousquet, S. Singh, F. Muñoz, A. H. Romero, Pyprocar: A python library for electronic structure pre/post-processing, Computer Physics Communications 251 (2020) 107080. doi:https://doi.org/10.1016/j.cpc.2019.107080.
URL http://www.sciencedirect.com/science/article/pii/S0010465519303935

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[27] E. V. Miu, J. R. McKone, G. Mpourmpakis, Global and local connectivities describe hydrogen intercalation in metal oxides, Phys. Rev. Lett. 131 (2023) 108001. doi:10.1103/PhysRevLett.131.108001.
URL https://link.aps.org/doi/10.1103/PhysRevLett.131.108001

[28] C. B. Sullivan, A. A. Kaszynski, Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk), Journal of Open Source Software 4 (37) (2019) 1450. doi:10.21105/joss.01450.
URL https://doi.org/10.21105/joss.01450

[29] F. Pinilla, N. Vásquez, I. Chacón, J. R. Maze, C. Cárdenas, F. Munoz, Spin-active single photon emitters in hexagonal boron nitride from carbon-based defects, Physica Scripta 98 (9) (2023) 095505. doi:10.1088/1402-4896/aceb1d.
URL https://dx.doi.org/10.1088/1402-4896/aceb1d

[30] C. Jara, T. Rauch, S. Botti, M. A. L. Marques, A. Norambuena, R. Coto, J. E. Castellanos-Águila, J. R. Maze, F. Munoz, First-principles identification of single photon emitters based on carbon clusters in hexagonal boron nitride, The Journal of Physical Chemistry A 125 (6) (2021) 1325–1335. doi:10.1021/acs.jpca.0c07339.
URL https://doi.org/10.1021/acs.jpca.0c07339

[31] P. Auburger, A. Gali, Towards ab initio identification of paramagnetic substitutional carbon defects in hexagonal boron nitride acting as quantum bits, Phys. Rev. B 104 (2021) 075410. doi:10.1103/PhysRevB.104.075410.
URL https://link.aps.org/doi/10.1103/PhysRevB.104.075410

[32] F. Evers, A. D. Mirlin, Fluctuations of the inverse participation ratio at the anderson transition, Phys. Rev. Lett. 84 (2000) 3690–3693. doi:10.1103/PhysRevLett.84.3690.
URL https://link.aps.org/doi/10.1103/PhysRevLett.84.3690

[33] N. C. Murphy, R. Wortis, W. A. Atkinson, Generalized inverse participation ratio as a possible measure of localization for interacting systems, Phys. Rev. B 83 (2011) 184206.

doi:10.1103/PhysRevB.83.184206.
URL https://link.aps.org/doi/10.1103/PhysRevB.83.184206

[34] C. Pashartis, O. Rubel, Localization of electronic states in iii-v semiconductor alloys: A comparative study, Phys. Rev. Appl. 7 (2017) 064011. doi:10.1103/PhysRevApplied.7.064011.
URL https://link.aps.org/doi/10.1103/PhysRevApplied.7.064011

[35] F. Munoz, F. Pinilla, J. Mella, M. I. Molina, Topological properties of a bipartite lattice of domain wall states, Scientific reports 8 (1) (2018) 17330. doi:10.1038/s41598-018-35651-6.

[36] C. Linderälv, W. Wieczorek, P. Erhart, Vibrational signatures for the identification of single-photon emitters in hexagonal boron nitride, Phys. Rev. B 103 (2021) 115421. doi:10.1103/PhysRevB.103.115421.
URL https://link.aps.org/doi/10.1103/PhysRevB.103.115421

[37] M. Malki, G. S. Uhrig, Delocalization of edge states in topological phases, Europhysics Letters 127 (2) (2019) 27001. doi:10.1209/0295-5075/127/27001.

[38] H. Zhang, C.-X. Liu, X.-L. Qi, X. Dai, Z. Fang, S.-C. Zhang, Topological insulators in Bi2Se3, Bi2Te3 and Sb2Te3 with a single dirac cone on the surface, Nature physics 5 (6) (2009) 438–442.

[39] M. W. Doherty, N. B. Manson, P. Delaney, F. Jelezko, J. Wrachtrup, L. C. Hollenberg, The nitrogen-vacancy colour centre in diamond, Physics Reports 528 (1) (2013) 1–45, the nitrogen-vacancy colour centre in diamond. doi:https://doi.org/10.1016/j.physrep.2013.02.001.
URL https://www.sciencedirect.com/science/article/pii/S0370157313000562

[40] Y. Jin, M. Govoni, G. Wolfowicz, S. E. Sullivan, F. J. Heremans, D. D. Awschalom, G. Galli, Photoluminescence spectra of point defects in semiconductors: Validation of first-principles calculations, Phys. Rev. Mater. 5 (2021) 084603. doi:10.1103/PhysRevMaterials.5.084603.
URL https://link.aps.org/doi/10.1103/PhysRevMaterials.5.084603

[41] F. Munoz, M. Vergniory, T. Rauch, J. Henk, E. V. Chulkov, I. Mertig, S. Botti, M. A. Marques, A. Romero, Topological crystalline insulator in a new bi semiconducting phase, Scientific Reports 6 (1) (2016) 21790. doi:10.1038/srep21790.

[42] I. Aguilera, H.-J. Kim, C. Friedrich, G. Bihlmayer, S. Blügel, F_2 topology of bismuth, Phys. Rev. Mater. 5 (2021) L091201. doi:10.1103/PhysRevMaterials.5.L091201.
URL https://link.aps.org/doi/10.1103/PhysRevMaterials.5.L091201

[43] D. Shoenberg, The fermi surfaces of copper, silver and gold. i. the de haas-van alphen effect, Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 255 (1052) (1962) 85–133. arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1962.0011, doi:10.1098/

rsta.1962.0011.
URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1962.0011

[44] E. Parzen, On estimation of a probability density function and mode, The annals of mathematical statistics 33 (3) (1962) 1065–1076. doi:10.1214/aoms/1177728190.

[45] R. Nelson, C. Ertural, J. George, V. L. Deringer, G. Hautier, R. Dronskowski, Lobster: Local orbital projections, atomic charges, and chemical-bonding analysis from projector-augmented-wave-based density-functional theory, Journal of Computational Chemistry 41 (21) (2020) 1931–1940.