Simulation Science Laboratory 2018

An Analysis Tool for Materials Design

Students:
Praneeth Katta Venkatesh Babu
Christian Partmann
Johannes Wasmer

Supervisors:
Stefan Blügel Prof. Dr.
Stefan Rost MSc
Quantum Theory of Materials PGI-1
Forschungszentrum Jülich

Abstract — Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Keywords Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor.

AICES
Schinkelstr. 2
Rogowski Building
4th Floor
52062 Aachen

Acknowledgments

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Contents

1 Introduction **1**

2 Theoretical Background **3**

3 Implementation **5**

3.1 HDF Preprocessor Module 6
  3.1.1 Interface 6
  3.1.2 Implementation for Band Structure Visualization 8
3.2 Visualization Module & Interactive Graphical Frontends 8
  3.2.1 Visualization Module Interface 8
  3.2.2 Desktop Frontend Implementation 9
  3.2.3 Web Frontend Implementation 9

4 Manuals **11**

4.1 User Manual 11
  4.1.1 System Requirements & Installation 11
  4.1.2 Input Data Formats 11
  4.1.3 GUI Usage 11
  4.1.4 Troubleshooting 11
4.2 Developer Manual 11
  4.2.1 Extending the Preprocessor 11
  4.2.2 Extending the Visualization & Frontends 11

5 Applications **13**

5.1 Web Frontend: $\mathrm{MoSe_2}$ Crystal 13
5.2 Desktop Frontend: Co Crystal 14
5.3 Derived physical Quantities: Differentiation 14
5.3.1 Differentiation 14

6 Conclusion **17**

List of Figures

3.1 Module Design 6

3.2 The preprocessor module. 7

3.3 Visualization Module Design: Example Structure for two applications `BandPlot` and `DOSPlot`, and two plotting libraries `matplotlib` and `bqplot`. 9

List of Abbreviations

Chapter 1

Introduction

Chapter 2

Theoretical Background

...some introduction scentence depending on into chapter...

Fleur computes the electronic structure in crystals using the Density Functional Theory approach (DFT), which is the state of the art method for this problem. The many-body Schrödinger equation, that can be used to describe electrons in solids, is almost impossible to solve directly, because the storage of the wavefunctions of each of the N electrons in the system at each spatial coordinate exceeds the available memory of any currently available computer for even quite small N. This motivates the DFT approach, that uses two fundamental theorems to reduce the computational complexity of the many-body problem significantly. The Hohenberg-Kohn theorem allows to use the electron density instead of the N-electron wavefunctions to uniquely characterize the ground state of a system. With the Kohn-Sham equations, the interacting Hamiltonian of the system can be replaced by non-interacting equations with an effective potential. This approach reduces the dimensionality of the problem from N^3 to 3 and trades the interacting Hamiltonian for a set of non-interacting equations that have to be solved self-consistently. In general, the DFT approach is not just limited to computations of electrons in crystals, but is also for example used in chemestry to compute nonperiodic molecules. The output of a DFT calculation includes various physical quantities (for example the 3-dimensional spatial electron density, which is the central quantity in DFT calculations), but for this project we focus on two quantities that are easy to interpret, already reduced in dimensionality and frequently used in both experimental and theoretical physics.

The band structure $E(\vec{k})$ represents the eigenenergies of the eigenfunctions of the Hamiltonian for each crystal momentum $\vec{k}$. It represents the dispersion relation of electrons in the crystal and relates allowed momenta and energies. In general, $E(\vec{k})$ is defined at any point inside the Brillouin zone, which is a special choice of the unit cell in the reciprocal lattice of the crystal. Both, the real-space lattice and the reciprocal lattice are shown for a face centered cubic (fcc) crystal in figure **TODO**. For larger unit cells with fewer symmetries, the Brillouin zone can be much more complicated than in the shown example. In order to reduce the dimensionality of $E(\vec{k})$ with $k \in \mathbb{R}^3$, the dispersion relation is only sampled along a discrete one-dimensional path between high symmetry points in the Brillouin zone. This path still contains most of the relevant physical features.

The second central quantity in this project is the Density of States (DOS) $D(E)$, which describes the number of states per energy interval, which is independent from its corresponding crystal momentum. In this sense, the DOS is also derived from $E(\vec{k})$, but instead of just selecting a subset, the $\vec{k}$ dependency is summed out. Both complementary quantities together are a common choice for comprehensively visualizing electronic structure data while still capturing most of the important physics.

For applications, it is useful to investigate where the contributions to $E(k)$ and $D(E)$ come from. Therefore, the data contains the weights of each basis function of the DFT calculation belonging to the individual atom groups and the orbitals. This means, spatial information about the system can be restored by considering only contributions from certain atom groups. (An atom group contains all atoms that are equivalent with respect to symmetries of the real space lattice.) On the other hand, the projection on the hydrogen orbitals s, p, d, f encode information about the shape of the wavefunction at each atom. These contributions are stored in the form of relative weights that can be summed to include contributions for multiple groups and orbitals. In case of distinct spins in the crystal, $E(k)$, $D(E)$ and the weights can be different for both spins and are therefore stored individually.

Chapter 3

Implementation

As per the requirements expounded upon in the introduction, the deliverable of the project should be a finished software product. The software is written in Python so as to integrate easily with the research group's ongoing software projects around the Fleur code [Blü+18]. These are chiefly the group's materials science tool collection `masci-tools` [RBR18] , where also this project's code is hosted, and the 'Automated Interactive Infrastructure and Database for Computational Science' (AiiDA) [Piz+16]. The product clients split into frontend users and code developers. In order to accommodate this, the product is organized into three unidirectionally dependent subpackages or -modules, see Figure 3.1.

An important design consideration was to account for unknown use cases. This has been realized in each submodule by the decoupling of **interface** and **implementation**. The interfaces do not rely on any specific input file format, visualization method or package, unlike the implementations for a specific task or **application**. The word 'application' in this section denotes the band structure and density of states visualization, and for these, implementations are provided.

This design choice was also one reason why the product does not reuse any of the `masci-tools` routines which partly solve quite similar problems, but seemed to be too specialized in an cursory code review. For these developers, one added value of the project product could be to inspire the hopefully easy integration into a common interface, where the current abstraction level could serve as a starting point.

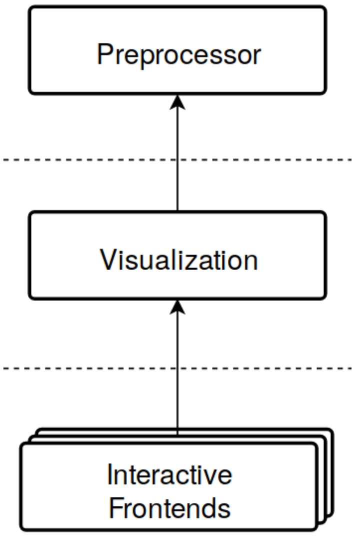

Figure 3.1: Module Design

3.1 HDF Preprocessor Module

3.1.1 Interface

This is the 'backend' of the tool. It is basically a file reader for the input data, for example a Fleur simulation output. Supported formats are the Hierarchical Data Format (HDF) [Kor11] for the band structure, and a simple Fleur-specific comma-separated values (CSV) format for the density of states (DOS).

The HDF format is basically a binary flexible container for all kinds of common binary and text file formats, each of which constitutes a `Dataset` inside the HDF file. The format supports metadata annotation and high-throughput input/output (I/O). As a consequence, it is considered by some developers in some application domains relying on numerical simulation codes, to be one possible base for the establishment of common domain-specific rich data exchange standards in order to increase code interoperability. These developers are in the process of extending their codes' I/O capabilities towards that end. However, HDF's flexibility comes at the price of a relatively complex Application Programming Interface (API) as the keyhole for all operations.

The preprocessor module tries to hide that complexity by offering the `Recipes` interface, see Figure 3.2. A specific application `Recipe` is a dictionary that aims to describe a complete Extract-Transform-Load (ETL) pipeline for one specific application. The 'extract' is the reading of a dataset from HDF, the 'transform' a sequence of once-through functions applied to the the dataset, and the 'load' the aggregation of all transformed datasets into one runtime object that has all the methods for operations on the data that are going to be used later on in the intended application.

The 'transform' and 'output' type methods are defined in hierarchical `Transform` and `Output_Type` classes, which sort them from general to application-specific applicability. This structure is built using Python's `AbstractBaseClass` (`ABC`) interface and multiple inheritance. The advantages of the 'Recipes approach' are:

- All ETL processes for one application are collected in one simple list (the recipe), not locked in different code locations with conflicting contexts. In this list, entries can be sorted in any manner, e.g. alphabetical for perusal.
- Recipes are de/serializable (can be read from and saved to disk) and thus be created and manipulated by code.
- The ETL processes declared in this way can be easily reused across applications. A recipe can combine different output types into a new type.

The feature that enables this flexibility is **type introspection**: the preprocessor processes the datasets listed in the recipe in the order of their mutual dependencies as found in the used transform and output methods. When all transformed datasets have been added to the object, all specified output types are searched and all their methods and attributes added. Thus the output object's type is defined at runtime, when the preprocessing is finished.

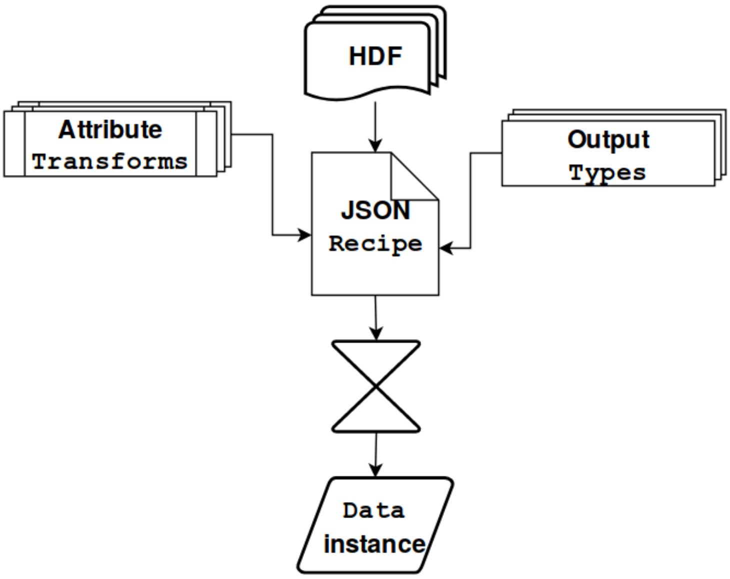

Figure 3.2: The preprocessor module.

Listing 3.1: Recipe `FleurBands` excerpt: dataset `BravaisMatrix`

```
      "bravaisMatrix": {
```

```
                "h5path": "/cell/bravaisMatrix",
                "description": f"Coordinate␣transformation␣internal␣to␣physical␣↩
                    ↪␣for␣atoms",
                "transforms": [’Transform.move_to_memory’,
                               [Transform.scale_with_constant, "bohr␣radius", "↩
                    ↪angstrom"]]
            },captionpos
```

3.1.2 Implementation for Band Structure Visualization

$$W^{\text{eff}}_{s,k,\nu} = \left(\frac{\sum\limits_{\substack{g \in \text{groups} \\ l \in \text{characters}}} n_{s,k,\nu,g,l} N_g}{\sum\limits_{\substack{g \in \text{all groups} \\ l \in \text{all characters}}} n_{s,k,\nu,g,l} N_g} \right) \left(W^{\text{unf}}_{s,k,\nu} \right)^{\alpha} \tag{3.1}$$

$W^{\text{eff}}_{s,k,\nu}$ is the effective weight. The indices denote: s spin, k point on k-path, ν band, g group, l character. $n_{s,k,\nu,g,l}$ is the state-specific l-like charge. N_g denotes the number of atoms in a group. $W^{\text{unf}}_{s,k,\nu}$ is the unfolding weight and α it's exponent.

TODO describe how bandstructure data is preprocessed for visualization including user selections AND optimizations

3.2 Visualization Module & Interactive Graphical Frontends

TODO Frontends: combine into one subsection when finished. usage will be in user manual next section.

3.2.1 Visualization Module Interface

The Python visualization landscape [VR17] abounds with a plethora of plotting libraries for different application contexts and technology stacks. Thus the project's visualization module first design objective is to account for that by decoupling library use and modularizing applications. This structure again is built using Python's `AbstractBaseClass` (`ABC`) interface and multiple inheritance. Each application is represented by an abstract base class that contains the common plotting method signatures. Each plotting library is represented by an abstract base class that contains library-specifics. An implementation inherits both from one or more application- and a library-base class. See Fig. 3.3 for clarification. Thus switching the library in a frontend should require minimal adjustment, and a new application can be build using existing ones. The second design objective is for the plotting methods to to hide all interactions with the actual plotting library used under the hood, while the method parameters only relates to the preprocessed data being plotted. Thus different frontend implementations only need call one identical method for one plot and receive the identical visualization with identical behavior.

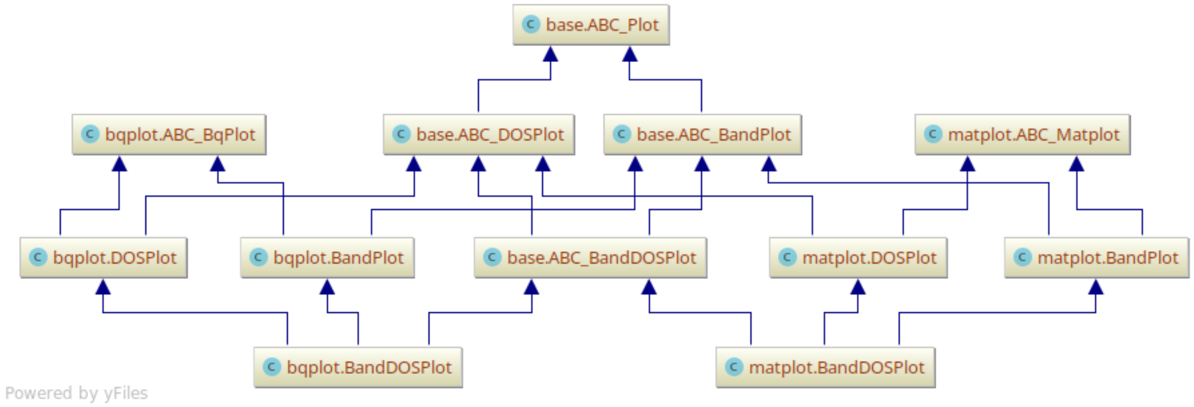
base.ABC_Plot
bqplot.ABC_BqPlot
base.ABC_DOSPlot
base.ABC_BandPlot
matplot.ABC_Matplot
bqplot.DOSPlot
bqplot.BandPlot
base.ABC_BandDOSPlot
matplot.DOSPlot
matplot.BandPlot
bqplot.BandDOSPlot
matplot.BandDOSPlot
Powered by yFiles

Figure 3.3: Visualization Module Design: Example Structure for two applications `BandPlot` and `DOSPlot`, and two plotting libraries `matplotlib` and `bqplot`.

3.2.2 Desktop Frontend Implementation

3.2.3 Web Frontend Implementation

TODO describe tool selection process, refer to [Was19].

Chapter 4

Manuals

4.1 User Manual

4.1.1 System Requirements & Installation

4.1.2 Input Data Formats

4.1.3 GUI Usage

The Desktop and Web Frontend are functionally identical and use the same graphical descriptors. Thus these points hold true for both alike.

4.1.4 Troubleshooting

4.2 Developer Manual

4.2.1 Extending the Preprocessor

4.2.2 Extending the Visualization & Frontends

Chapter 5

Applications

To illustrate the use of the graphical user interface, two different physical applications are shown in the desktop and the web frontend, respectively. From the physics point of view, the example in the desktop version focuses more on the density of states and the visualization of spin contributions, while the dataset in web frontend focuses on the band structure $E(k)$ and the visualization of defect states in supercells.

5.1 Web Frontend: $\mathrm{MoSe_2}$ Crystal

Figure **TODO** shows the visualization of a band structure calculation of a 3 dimensional Molybdenum diselenide ($\mathrm{MoSe_2}$ bulk) crystal using the default settings of the GUI. Even with the default settings the band structure plots clearly indicate, that $MoSe_2$ is a semiconductor, since there are no states at the Fermi level. Because the minimum of the conduction band is located at an other k as the maximum of the valence band, the plot shows that $\mathrm{MoSe_2}$ has an indirect band gap. This indicates that for the transition with the smallest energy difference between valence and the conduction band, both, energy and momentum have to change.

In contrast to the 3 dimensional extended $MoSe_2$ crystal, a $MoSe_2$ monolayer (see Fig. **TODO**) has a direct band gap but is still a semiconductor. Furthermore, the $\mathrm{MoSe_2}$ monolayer has a defect atom in every 9th unit cell and the DFT computation is therefore done in a 3×3 supercell to restore periodicity. This is the reason for the much greater number of states in the band structure plot.

Since the Brillouin zone of the supercell is smaller than the Brillouin zone of the same crystal without the defect, the supercell Brillouin zone is unfolded to the same size as the Brillouin zone of the unperturbed lattice. To account for the fact that the defect is only present in every 9th cell and its relative importance for the spectrum is therefore degraded, an unfolding weight is introduced to visualize the relative importance of bands in the unfolded Brillouin zone. By default, the unfolding weight is used by our visualization tool, but it can gradually be turned off in order to highlight the impact of defect states. This is shown in figure **TODO**. To even better visualize the defect state, it would also be possible to select the atom group belonging the defect atoms only. In this example, the analysis with reduced band unfolding shows, that there are many more direct band gaps origination from the defect state.

5.2 Desktop Frontend: Co Crystal

5.3 Derived physical Quantities: Differentiation

The kind of datasets handled in the scope of the project did not lend themselves readily to applications of automatic differentiation techniques. Nevertheless, it is possible to derive meaningful physical quantities from the band structure using numerical differentiation techniques.

The effective mass m^* represents the mass, that an electron appears to have due to the inter-atomic forces in the crystal. At every $\vec{k}_0$, where $E(\vec{k})$ has a local extremum, $E(\vec{k})$ can be expanded in a Taylor series with a vanishing first order term $E(k) = E_0 + \frac{\partial E(\vec{k})}{\partial \vec{k}} \cdot (\vec{k} - \vec{k}_0)^2$. Comparing this to the dispersion relation of a free electron $E(\vec{k})_{free} = \frac{\hbar^2 \vec{k}^2}{2m_e}$ motivates the general definition

$$m^*_{k_i,k_j} = \hbar^2 \left(\frac{\partial^2 E(\vec{k})}{\partial k_i \partial k_j} \right)^{-1} \tag{5.1}$$

Since $\frac{\partial^2 E(k)}{\partial \vec{k}_i \partial \vec{k}_j}$ depends on the direction of the partial derivatives, $m^*\Big|_{i,j}$ is a tensor. Because the band structure files only contain a discrete sampled path in the Brillouin zone, only the derivatives that correspond to the direction from one high-symmetry point to the next can be computed. We are only interested in the diagonal terms of $m^*\Big|_{i,j}$.

A second potentially interesting quantity is the group velocity $v_G(\vec{k})$ associated to each band. The group velocity $v_G(\vec{k})$ at the Fermi energy $E = E_F$ is called Fermi velocity.

$$v_G(\vec{k})\Big|_i = \frac{1}{\hbar} \frac{\partial E(\vec{k})}{\partial \vec{k}_i} \tag{5.2}$$

5.3.1 Differentiation

Since the k-mesh in DFT calculations is potentially very sparse, low order finite difference schemes are not expected to work well. Alternatively, one way to exploit all data points efficiently is to use fast Fourier transform methods, which are equivalent to the derivation of a truncated Fourier series. This method is expected be well-suited for the problem since the graph of the band structure $(k, E(|k|))$ with $k \in \{-\Gamma, ..., H, ..., \Gamma\}$ is periodic, where Γ and H are arbitrary representatives of the high symmetry points. This periodicity in the reciprocal space is a direct consequence of the periodicity of the crystal.

In Fourier space, spatial derivatives transform into multiplications, which can easily be shown by partial integration.

$$f^{(n)}(x) = \mathcal{F}^{-1}\left((ik)^n \mathcal{F}(f(x))\right) \tag{5.3}$$

To test the FFT differentiation method, a band was selected that did not have intersections with other bands within the interval between two high symmetry points and a stationary point at the high symmetry points. Then a resolution study was done to investigate the impact of the number of points within the interval.

The comparison between the FFT and a first-order central difference approximation of the second derivative (FD) is shown in Fig. **TODO**, where different k-mesh resolutions are compared to a derivative that is almost fully converged. The comparison indicates that for small N, the error of the FFT method is significantly smaller than the error of the FD derivative. This is especially striking in the vicinity of the high symmetry points, where the error of the differentiation is required to be small in order to get good approximations for m^*, which is only meaningful close to the high symmetry points.

When using more points, the difference between both approximation schemes is not significant. It is interesting to note, that the FFT method does not work anymore in the limit of extremely many points. In this case the derivative is dominated by Gibbs oscillations. These are most likely caused by small discontinuities in $E(k)$, due to basis changes inside the DFT computation.

At the current state the question remains to be answered weather the computation of m^* and v_g is useful. The number of bands, which the computation can be applied to is extremely limited. Since the bands in the data files are not labeled according to their corresponding eigenfunction but sorted by value, there are discontinuities at each point, where two bands intersect. This problem might be solved in the future.

Chapter 6

Conclusion

Bibliography

[Blü+18] Stefan Blügel et al. *Fleur: full potential linearized augmented planewave code*. http://www.judft.de. 2018.

[Kor11] Sandeep Koranne. "Hierarchical Data Format 5 : HDF5". In: *Handbook of Open Source Tools*. Boston, MA: Springer US, 2011, pp. 191–200. ISBN: 978-1-4419-7719-9. DOI: 10.1007/978-1-4419-7719-9_10. URL: https://doi.org/10.1007/978-1-4419-7719-9_10.

[Piz+16] Giovanni Pizzi et al. "AiiDA: automated interactive infrastructure and database for computational science". In: *Computational Materials Science* 111 (2016), pp. 218–230. ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2015.09.013. URL: http://www.sciencedirect.com/science/article/pii/S0927025615005820.

[RBR18] Philipp Rüßmann, Jens Bröder, and Stefan Rost. *masci-tools*. https://github.com/JuDFTteam/masci-tools. 2018.

[VR17] Jake VanderPlas and Nicolas Rougier. *Python Visualization Landscape*. https://github.com/rougier/python-visualization-landscape. 2017.

[Was19] Johannes Wasmer. *Software Development Notes*. https://github.com/Irratzo/notes. 2019.