Guide to Vaspirin 1.1

Instituto Tecnológico de Aeronáutica
Group of Semiconductor Materials and Nanotechnology
Daniel S. Koda[*]
Ivan Guilhon[†]

May 2016

Abstract

Vaspirin is a post-processsing tool for VASP made in Python. Its goals are to simplify the analysis of simulations, make standard and beautiful figures easily and create a continuous development and centralization of the tool throughout the years for the Group of Semiconductor Materials and Nanotechnology (GMSN). This document describes the capabilities of the software and provides a beginner's guide to its utilization. We cover the installation of the script and demonstrate the use of this tool with examples, such as band structure and density of states plotting.

Contents

[*] `danielskoda@gmail.com`
[†] `ivanguilhonn@gmail.com`

1 Introduction

1.1 About Vaspirin

Origins: Vaspirin has been created to unify and document scripts made by members of GMSN in a way which benefits both experienced students with its pragmatic and well-tested approach as well as beginner undergraduate students by lessening the initial difficulties of juggling programming languages, solid state physics, undergraduate studies AND simulations at once.

Authors: Daniel S. Koda (MSc student in 2016) and Ivan Guilhon (PhD student in 2016) are responsible for the beginning of this project and the documentation.

Software: The raw code and the installation package can be found at: `https://github.com/gmsn-ita/vaspirin.git`. New versions will be constantly updated in this link.

1.2 Installation

Requirements: Vaspirin uses Python 3 as standard language. The packages `numpy`, `matplotlib`, `scipy`, `pylab`, and `setuptools` are required in order to take full advantage of Vaspirin. XMGrace is also recommended to plot `.bfile` scripts generated by Vaspirin. `git` is necessary if you want to contribute to the software. Please use your package manager to install these packages before proceeding with the installation.

Installing: Directly download the Vaspirin code from `https://github.com/gmsn-ita/vaspirin.git` or simply execute:
`git clone https://github.com/gmsn-ita/vaspirin.git`.

Once downloaded, go to the directory `vaspirin/` and execute execute:
`# python setup.py install` or `$ sudo python setup.py install`

If Python 3 is not the default Python language in your operating system (confirm by executing `python --version`), you may want to try:
`# python3 setup.py install`

From this point on, you can execute `vaspirin` as a normal command-line interface software.

2 Running Vaspirin

2.1 Flags

Introduction: Vaspirin was created with an eye to flexibility. Therefore, most of its features are activated by running the script with flags starting with a dash `-`. All possibilities are briefly listed here and demonstrated in the examples section.

Further development may take advantage of these functions and classes put together in order to make a more complex script.

`-bs outcarFile`	Reads the eigenvalues from the `outcarFile` specified. If the file is not given, the default `OUTCAR` file is read from the current directory.
`-dos doscarFile`	Reads the density of states from the `doscarFile` specified. If the file is not given, the default `DOSCAR` file is read from the current directory.
`-char procarFile`	Reads the character of simulated band structures from the `procarFile` specified. If the file is not given, the default `PROCAR` file is read from the current directory.
`-proj procarFile PROJECTION`	Reads the character of simulated band structures from the `procarFile` specified and the projection description in the `PROJECTION` file. If the file is not given, the default `PROCAR, PROJECTION` file are read from the current directory. The format of the `PROJECTION` file are specified in subsection 2.2.
`-compare OUTCAR1 OUTCAR2`	Plot two different OUTCARs in a same band structure. Assumes both calculations are made with the same k-points line, imported from the KPOINTS file.
`-plot`	Plots the specified function using XMGrace. A `.dat` (or a directory replete with `.dat` files) and a `.bfile` are generated. The `.bfile` should be executed as batch by XMGrace after using Vaspirin so that the plot can be fully generated.
`-pyplot`	Plots the specified function using PyPlot.
`-quiet`	Do not print the introduction message.
`-yaxis y1 y2`	Define the y-scaling as the range [min(`y1`,`y2`), max(`y1`,`y2`)].
`-ignore N`	Ignore the first N k-points when plotting band structures. Useful for HSE calculations, in which k-points for band structures are included in the same file as the self-consistent k-points mesh.
`-ref reference`	Define an arbitrary reference for the 0 eV level in band structures. Recognized arguments are: `vbm`, `efermi`, `e-fermi` and any floating-point number. Default: `vbm`.
`-soc`	Import OUTCAR with spin-orbit coupling for setting correctly the valence band based on occupations.
`-interpolate N`	Linearly interpolate N points between each point in band structures. Useful for coloring projected band structures. Default: 0
`-markersize size`	Changes the marker size for projected and character band structures. Default: 0.5.
`-ps psFile`	Enables direct exporting from XMGrace to PS files. Requires the use of a XMGrace command after `vaspirin`

2.2 Standards requiring manual entry

`KPOINTS` header:	To create a band structure specifying symmetry k-points, a header describing which are these special k-points should be written in the `KPOINTS` file. The header has the following format: `SymPoint1 Index1, SymPoint2 Index2, SymPoint3 Index3, ...`

A specific example is shown in section 3.1.

`PROJECTION` file: Projected band structures are creating by summing all contributions from atoms which belong to specified materials. Therefore, which atoms are part of each material is an information supplied by the user. The `PROJECTION` file is responsible for conveying that data to `vaspirin`. It is a plain text file with the following format:

```
Material1Label atomsBelongingToMaterial1
Material2Label atomsBelongingToMaterial2
⋮
```

An example of application of this technique is shown in section 3.3.

`.dat` files: These files are output from `vaspirin` and contain information about the simulation loaded. For band structures (`eigenv.dat`), this file is a double-column table, in which the first column is the normalized k-point (based on the length and the size of the Brillouin zone) and the second is the corresponding eigenvalue. Bands are organized in blocks. In cases such as projection and orbital character, a folder named `bands_character` or `bands_projected` is created, and one `.dat` file is written for each band. This allows plotting special band structures with XMGrace.

`.bfile` files: This kind of file contain batch instructions for plotting data with XMGrace. They should be executed with this software, for example:

```
xmgrace -batch bands.bfile
```

3 Tutorial

In this section, actual examples of calculations will be supplied in order to clarify the use of `vaspirin`. The files for this example are available in the folder `tests/` in the `vaspirin` directory. No VASP simulations will be done at this point.

3.1 Plotting standard band structures

Changing the directory: As a first step, the default configurations from `vaspirin` are used. This is the more favorable situation, in which `vaspirin` is executed in the same folder as the VASP simulation. Therefore, all files have their default name and are all in the same directory. To start using `vaspirin`, please change your directory to that of interest.

`KPOINTS` header: The `KPOINTS` file must specify the path made in the band structure calculation, as well as their indexes. In this case, 130 k-points were used to form the entire path, distributed so that the Γ point were the 1st and the 130th points, the M point was the 41th point and the K point was the 70th point. Thus, the `KPOINTS` header should be:

```
G 0, M 40, K 69, G 129
```

Executing `vaspirin` and obtaining the data: To plot a simple band structure using `vaspirin`, just execute:

```
vaspirin -bs -plot
```

This tells `vaspirin` to acquire band structure data (`-bs` flag) from the default `OUTCAR` file and then plot the correspondent bands. This generate the files `eigenv.dat` and `bands.bfile`.

Generating the bands using XMGrace: Finally, you can create the plot by executing:

```
xmgrace -batch bands.bfile
```

The result of these operations for the given example files is shown in Fig. 1.

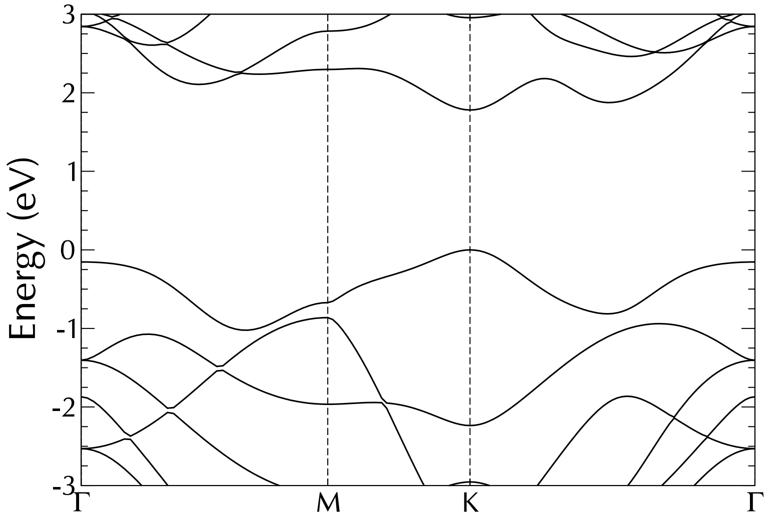

Figure 1: Example of a monolayer MoS_2 band structure plotted using `vaspirin -bs -plot`.

Changing the y-axis scale: `vaspirin` puts the 0 eV reference on the top of the valence band (additional features for Fermi energy, vacuum level and custom energies are being implemented). If the default range of [-3,3] eV is not enough, it is possible to customize these values simply by using the tag `-yaxis`. Continuing the example above with this tag:

```
vaspirin -bs -plot -yaxis -2 4
```

This new band structure is shown in Fig. 2.

It should be noted that `vaspirin` does not care about the order in which the numbers after `-yaxis` are written. Therefore, running `vaspirin -bs -plot -yaxis 4 -2` would yield exactly the same result.

Running files other than the default: If your files have names other than the default for VASP, or if they are in other directory, it suffices to call `vaspirin` with arguments specified after the tags, as in:

```
vaspirin -bs mos2/OUTCAR -plot
```

3.2 Plotting band structures projected on orbitals

In many cases, it is useful to analyze the system based on the formation of the bands via orbitals. `vaspirin` allows this kind of analysis to be done. Creating this requires a `PROCAR` file, which has the contributions of all points from the band structure.

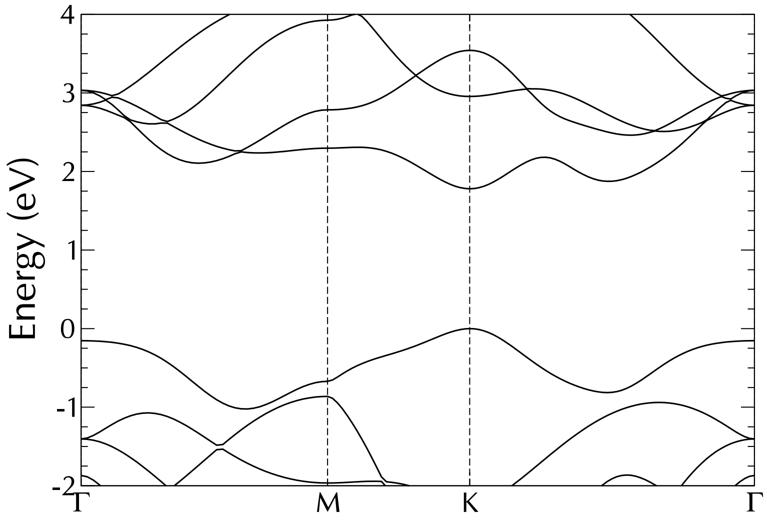

Figure 2: Example of a monolayer MoS_2 band structure plotted using `vaspirin -bs -plot -yaxis -2 4`.

Generating band structures with orbital character: To create band structures with orbital character, `vaspirin` should be executed with the `-char` tag active:
`vaspirin -bs -char -plot`

This loads the default `OUTCAR` and `PROCAR` files from the current directory. To use another files as input, an argument can be specified after the flags `-bs` and `-char`. `vaspirin` outputs the data to the folder `bands_character/` and the XMGrace batch `bandsCharacter.bfile`.

Plotting with XMGrace: To plot the resulting file with XMGrace, execute the following line:
`xmgrace -batch bandsCharacter.bfile`

The result is shown in Fig. 3.

Interpreting the figure: Orbital contributions scale the colored circular markers in the band structure. Therefore, bigger circles imply predominance of the specific orbital to the formation of the band exactly in that k-point. Red, green, blue and yellow circles depict contributions from s, $p_x + p_y$, p_z and d orbitals, respectively.

3.3 Plotting band structures projected on specific ions

Sometimes, as in van der Waals heterostructures, it is useful to know which bands come from each of its constituents. This not only helps visualize effects on electronic structures but also creates a pleasant interpretation to the viewer.

Creating the `PROJECTION` file: The `PROJECTION` file is a user-made text file which specifies which atoms compose each material. In the case of the MoS_2, only two different atoms

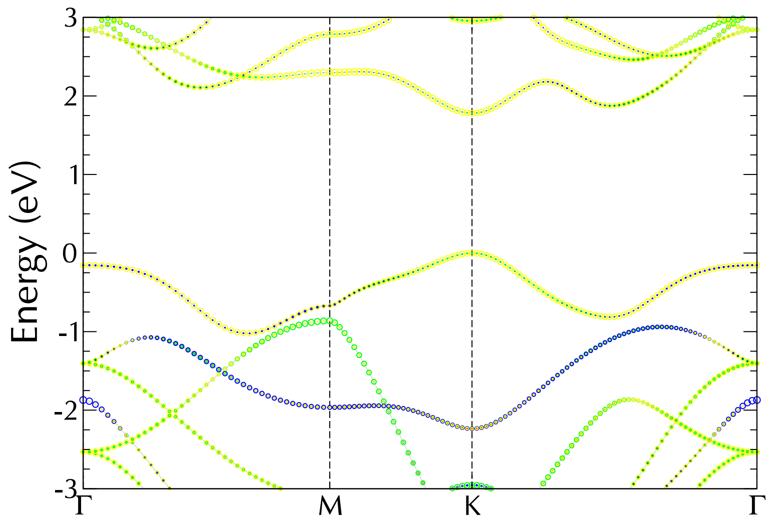

Figure 3: Example of a monolayer MoS_2 band structure plotted using `vaspirin -bs -char -plot`. Red, green, blue and yellow circles depict contributions from s, $p_x + p_y$, p_z and d orbitals, respectively.

are simulated, and they are part of a single material. If we wanted to analyze contributions of the molybdenum atom to the bands formation, as well as the sulfur atom to it, we would have to build the following `PROJECTION` file:

```
Mo 1
S 2..3
```

A 2×2 supercell of MoS_2, therefore with 12 atoms inside it, would have the following `PROJECTION` file:

```
Mo 1..4
S 5..12
```

Another equivalent way of specifying the information above would be:

```
molybdenum 1,2,3,4
sulfur 5,6,7..12
```

Atoms must be numbered according to their order in the `POSCAR` file and be separated by commas. The `7..12` entry indicates all atoms between the 7th and the 12th (including these both) belong to the material named `sulfur`. The labels can be chosen arbitrarily and is more like a guide for humans than a useful information for `vaspirin`. All atoms should be specified in the `PROJECTION` file.

Generating the data with `vaspirin`: To use the simulation and the specification from the `PROJECTION` file, just run the following command:

```
vaspirin -bs -proj -plot
```

A folder named `bands_projected` and a XMGrace batch named `bandsProjected.bfile` are created upon the execution of this command.

Plotting with XMGrace: To plot the resulting file with XMGrace, execute the following line:

```
xmgrace -batch bandsProjected.bfile
```

The result is shown in Fig. 4.

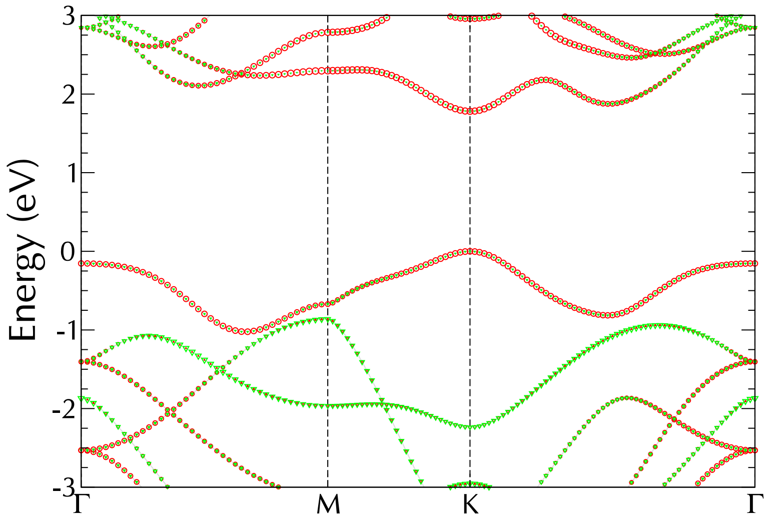

Figure 4: Example of a monolayer MoS_2 band structure plotted using `vaspirin -bs -proj -plot`. Red and green markers depict contributions from Mo and S atoms, respectively.

Interpreting the figure: Each material contributions scale the colored markers in the band structure. Therefore, bigger markers imply predominance of the specific material to the formation of the band exactly in that k-point. The materials are represented with red, green, blue, yellow, brown, gray, violet, cyan, etc. according to their order in the `PROJECTION` file.

3.4 Comparing two different band structures

Comparing two different band structures using `vaspirin` is an easy task. `vaspirin` still has a limitation and assumes that only bands executed with the same k-points mesh will be compared. Otherwise, results cannot have any meaning.

Comparing band structures: To compare band structures, `vaspirin` should be executed with the `-compare` tag instead of the `-bs` tag:

```
vaspirin -compare -plot
```

This loads the default `OUTCAR1` and `OUTCAR2` files from the current directory. To use another files as input, an argument can be specified after the flags `-compare`, as in `vaspirin -compare mat1-OUTCAR mat2-OUTCAR`.

`vaspirin` outputs the data to two `.dat` files, nominally `eigenv1.dat` and `eigenv2.dat`, and the XMGrace batch to `bandsComparison.bfile`.

Plotting with XMGrace: To plot the resulting file with XMGrace, execute the following line:

```
xmgrace -batch bandsComparison.bfile
```

An example is shown in Fig. 5.

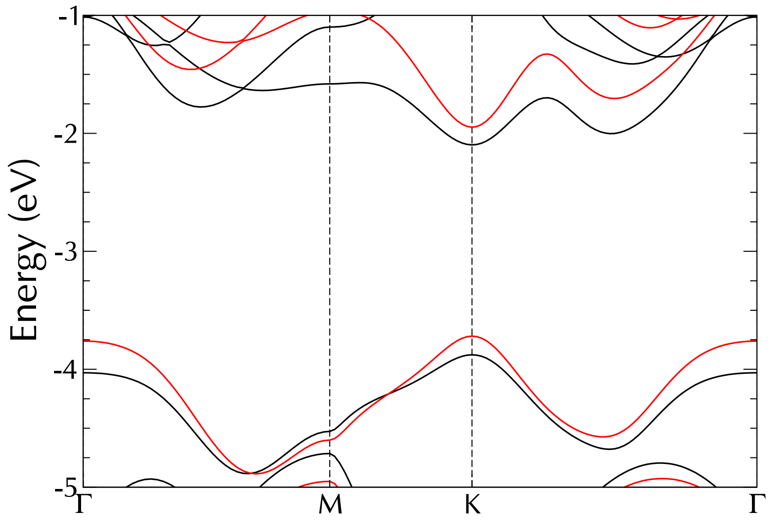

Figure 5: Example of a monolayer MoS_2 band structure compared with a WS_2 band structure plotted using `vaspirin -compare -ref 0 0 -plot`. Black lines are from the first band structure (in this case, MoS_2) and red lines are from the second band structure (WS_2).

Interpreting the figure: Bands from the second OUTCAR are displayed in red, while bands from the first OUTCAR are displayed in black. `vaspirin` does not yet supports changing colors in bands. Pay attention to the reference: if nothing is specified within the `-ref` flag, the valence bands maxima from both OUTCARs will be set as the 0 eV reference.

4 Other results

`vaspirin` also allows users to plot using the `matplotlib` Python 3 library. Sample results for the usage of this method are shown below.

4.1 Standard density of states with `-pyplot`

Density of states:

4.2 Standard band structure with `-pyplot`

Band structure:

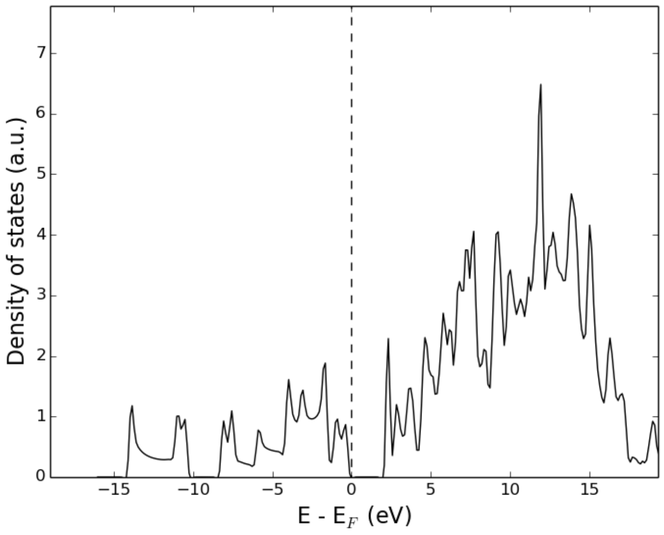

Figure 6: Example of result using -dos -pyplot tags.

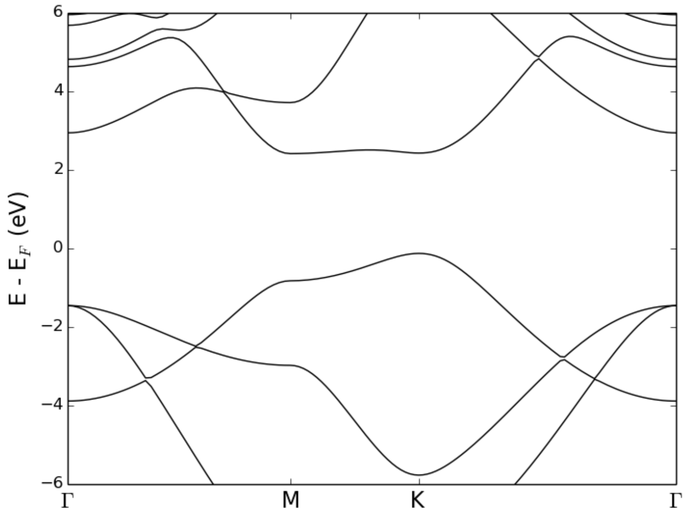

Figure 7: Example of result using -bs -pyplot tags.

4.3 BS + DOS plot with `-pyplot`

Band structure with DOS:

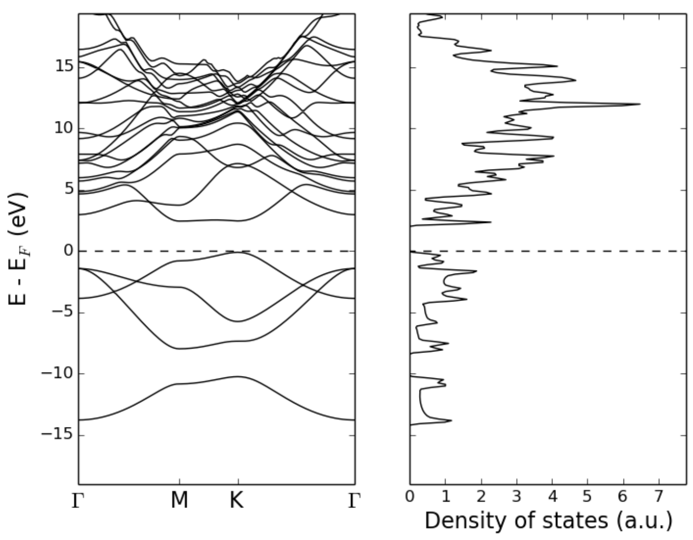

Figure 8: Example of result using -bs -dos -pyplot tags.