minushalf

Release 1.0

Henrique Fernandes Feitosa

Mar 06, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 The DFT -1/2 method

DFT-1/2, an alternative way of referring to the LDA -1/2[1][12] and GGA -1/2[2] techniques, is a method for approximate self-energy corrections within the framework of conventional Kohn-Sham DFT which can be used not only with the local density approximation (LDA), but also with the generalized gradient approximation (GGA)[11][2][12].

The method aims to predict energy gaps results with the same precision[2] as the quasiparticle correction[9] algorithm, considered the state of the art for calculating energy gap of semiconductors. In addition, the computational effort of the method is equivalent to the standard DFT approach and and is three orders of magnitude lower than the aforementioned GW method[16] , which allows the technique to be applied to complex systems.

[1] L. G. Ferreira, M. Marques, and L. K. Teles, Phys. Rev. B 78, 125116 (2008).

[2] L. G. Ferreira, M. Marques, and L. K. Teles, AIP Adv. 1, 032119 (2011).

[11] R. R. Pelá, M. Marques, L. G. Ferreira, J. Furthmüller, and L. K. Teles, Appl. Phys. Lett. 100, 202408 (2012).

[12] I. Guilhon, D. S. Koda, L. G. Ferreira, M. Marques, and L. K. Teles 1, *Phys. Rev. B 95, 045426 – Published 24 January 2018 <https://journals.aps.org/prb/abstract/10.1103/PhysRevB.97.045426>*

[9] G. Onida, L. Reining, and A. Rubio, Rev. Mod. Phys. 74, 601 (2002).

[16] R. R. Pelá, M. Marques, and L. K. Teles, J. Phys.: Condens. Matter 27 505502.

[13] G. Kresse and J. Furthmüller, Phys. Rev. B 54, 11169 (1996).

[14] G. Kresse and J. Furthmüller, Comput. Mater. Sci. 6, 15 (1996).

[15] P. Blaha, K. Schwarz, P. Sorantin, and S. B. Trickey, Comput. Phys. Communications. 59, 399 (1990), see www.wien2k.at.

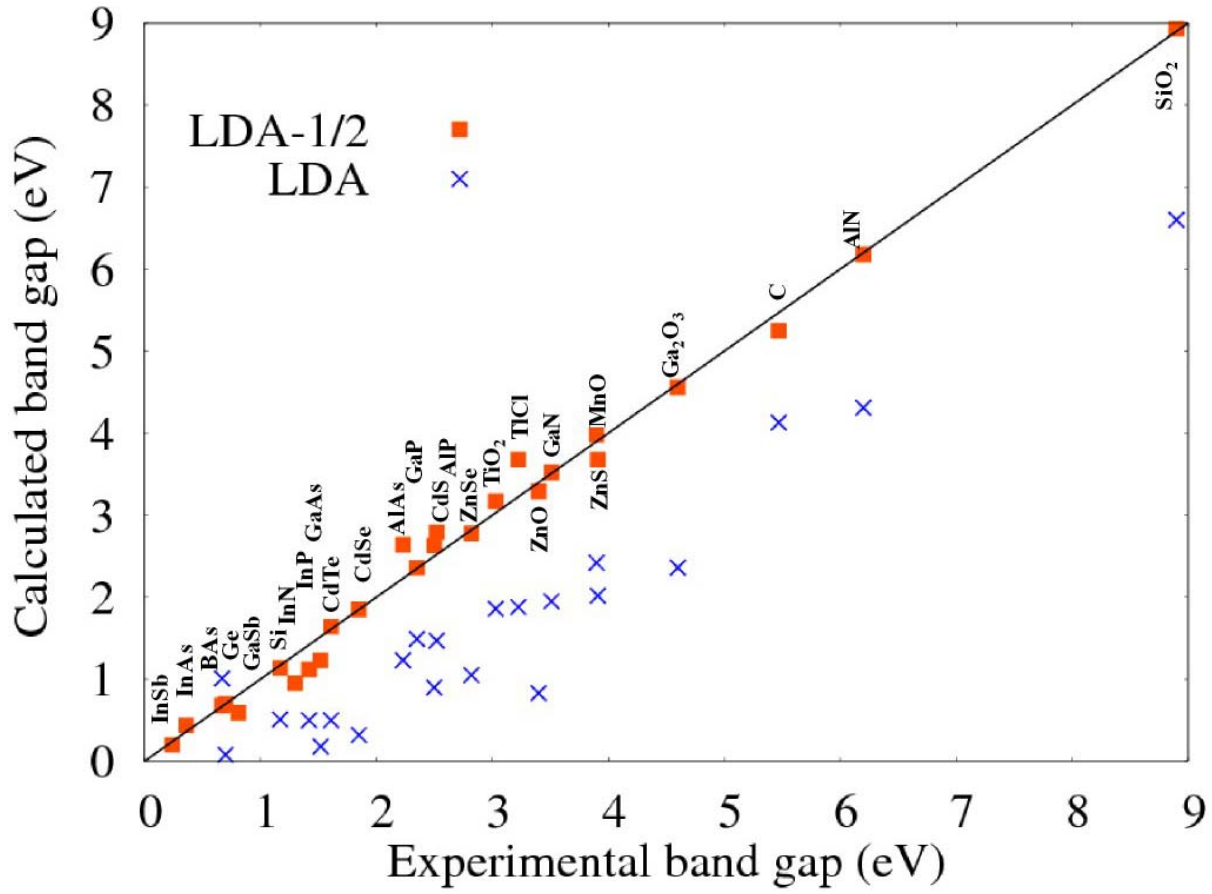

Fig. 1: Fig 1. Comparison of calculated band gaps with experiment. The red square are the SCF LDA-1/2 (standard LDA-1/2). The crosses are standard LDA. The small gap semiconductors are metals (negative gaps), when calculated with LDA. LDA-1/2 corrects the situation. The band structure calculations were made with the codes VASP[13](Page 1, 14) and WIEN2k.(Page 1, 15)(Page 1, 2)

1.2 What is minushalf?

Minushalf is a command line interface (CLI) developed by the group of semiconductor materials and nanotechnology (GMSN) that aims to automate the application of the DFT -1/2 method. The commands available in this CLI automate both the entire process and each of its steps in order to be used for several purposes.

1.3 An intuitive explanation of the DFT -1/2 method

This method aim to expand the half-occupation technique[3][4][5], formalized by Janak's theorem, to crystals using modern exchange-correlation approaches[6][7].

The Slater half-occupation scheme has already proven to be quite efficient for calculating atomic ionization energies values close to the experimental ones[Page 3, 5]. However, this technique cannot be applied blindly to extended systems like crystals, since the crystal is described by means of Bloch waves and removing the population of just one Bloch state would have no consequences[Page 1, 1]. Moreover, removing the population of one Bloch State and set periodic conditions would result in a infinitely charged system.

Thus, the proposed solution is to apply the Slater procedure to crystalline energy bands. The intuition for this application comes from the fact that the energy bands of a crystal are formed by the overlap of atomic orbitals, mainly by those that constitute the outermost layers[8]. This relationship can be quantified by the projection of the wave function in a given orbital, Figure 2 shows the p character for each atom in the band structure of the h-BN, the bigger the blue dots, the stronger the character. Thereby, considering this existing relationship, self-energy corrections performed in atoms could propagate and shift the energy of the bands, resulting in a band gap correction.

Fig. 2: Fig 2. p character for each atom in the h-BN band structure. The bigger the blue dots, the stronger the character.

1.4 How to perform potential correction in crystals

In this section, calculations were developed using some approximations in order to demonstrate intuitively how the potential correction in crystals is made. To access the rigorous demonstration, consult the references[Page 1, 1][Page 1, 2] .

Following the Slater half occupation procedure for atoms, a change in charge density is required to obtain the potential for semi-occupation and perform the consistent calculations using the Kohn-Sham equation.

Altough in extended systems like crystals a change in charge density in a unit cell would result in an infinitely charged system, which would lead to a divergence in the Kohn-Sham calculations. Furthermore, it would also be irrelevant to be able to modify only a finite amount of electrons in the crystal since the charge would become irrelevant to the infinite amount of electrons present in the system. To bypass this problem, it is necessary to find a new way to derive the semi-occupied potential.

[3] J.C. Slater and K. H. Johnson, Phys. Rev. B 5, 844 (1972).
[4] J.C. Slater, Adv. Quantum Chem. 6, 1 (1972).
[5] J. C. Slater and J. H. Wood, Int. J. Quant. Chem. Suppl. 4, 3 (1971).
[6] J. P. Perdew and A. Zunger, Phys. Rev. B 23, 5048 (1981).
[7] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. 77, 3865 (1996) .
[8] Holgate, Sharon Ann (2009). Understanding Solid State Physics. CRC Press. pp. 177–178. ISBN 978-1-4200-1232-3.

Firstly, one have to define the system that corresponds to the semi-occupied potential for a solid. For an atom containing N electrons in its ground state, the semi-occupied potential is defined as the potential of the atom with $N - \frac{1}{2}$ electrons. Similarly, we should consider that the semi-occupied potential of a solid would be the potential generated by a solid with $M - \frac{1}{2}$ electrons per primitive cell, where M is the number of electrons of the unit cell in the ground state, as shown in Figure 2.

Fig. 3: Fig 3. Scheme representing the unit cells of a solid that would generate the potential semi-occupied.

So, to outline the solution, suppose one have N independent charge distributions, where the mth is given by:

$$\begin{aligned} \rho_m(\vec{r}) &= (1 + f_m)n_m(\vec{r}) \\ n_m(\vec{r}) &= -q \cdot \eta_m \\ \int \eta_m(\vec{r})d\vec{r} &= 1 \end{aligned}$$

Where ρ represents the density of the distribution m, q represents the charge of the electron, η a normalized function in space and f represents an occupancy factor that varies continuously from 0(occupied) to -1(unoccupied).

Considering the charge density represented above, one can find the Coulomb potentials for each distribution by the Poison equation:

$$\nabla^2 V_m(\vec{r}) = \frac{q\rho_m(\vec{r})}{\epsilon_0}$$

Now, suppose another situation where one only alternate the occupation of the α level and the same charge distribution remains. In this scenario, the mth potential is given by:

$$\begin{aligned} \nabla^2 {V_m}'(\vec{r}) &= \frac{q{\rho_m}'(\vec{r})}{\epsilon_0} \\ f_i = {f_i}', i \neq \alpha \\ f_\alpha \neq {f_\alpha}' \end{aligned}$$

Thus, one want to calculate the potential for all distributions, which is obtained by adding of the potential of all distributions, as shown in the equations below.

$$\begin{aligned} \nabla^2 V(\vec{r}) &= \frac{q\sum_{m=1}^{N}\rho_m}{\epsilon_0} \\ \nabla^2 V'(\vec{r}) &= \frac{q\sum_{m=1}^{N}{\rho_m}'}{\epsilon_0} \end{aligned}$$

Subtracting these two equations:

$$\nabla^2(V(\vec{r}) - V'(\vec{r})) = \frac{q(f_\alpha - {f_\alpha}')n_\alpha(\vec{r})}{\epsilon_0}$$

Using the above equation for the specific case of $f_\alpha = 0$ and ${f_\alpha}' = -1/2$, the following equation is obtained:

$$\nabla^2(V(\vec{r}) - V'(\vec{r})) = \frac{qn_\alpha(\vec{r})}{2\epsilon_0} \Rightarrow V'(\vec{r}) = V(\vec{r}) - V_\alpha^{f_\alpha = -1/2}$$

Hence, using the equation above, one can calculate the potential semi-occupied from other potentials, which discards the need for modify the charge density. For a crystal, the equation is written as follows:

$$V_{crystal}^{-1/2} = V_{crystal} - V_{1/2e}$$

Where $V_{crystal}^{-1/2}$ is the potential of the semi-occupied crystal, $V_{crystal}$ is the potential of the crystal in the ground state and $V_{1/2e}$ is the potential of the respective level occupied with half an electron.

There is a problem with adding $-V_{1/2e}$ to all the atoms of an infinite crystal: the potential will diverge. $-V_{1/2e}$ is a potential of an excess charge of 1/2 proton and has a tail of 0.5/r that cannot be summed in an infinite lattice. Therefore the tail has to be trimmed by a step function[Page 1, 2]. Besides, it is worth mentioning that the values for CUT and A must not be chosen arbitrarily, by means of variational arguments it can be proved that the optimal values for these parameters are those that maximize the band gap of the crystalline system[Page 1, 1Page 1, 2] , as shown in Figure 3.

Hence, to obtain $V_{1/2e}$, the following equation is used for the atoms that compose the crystal[Page 1, 1Page 1, 2]:

$$V_{1/2e} = (V_{atom} - V_{atom}^{f_\alpha=-1/2}) \cdot \theta(r)$$

$$\theta(r) = \begin{cases} \theta(r) = A \cdot [1 - (\frac{r}{CUT})^8]^3), r \leq CUT \\ \theta(r) = 0, r > CUT \end{cases}$$

Where V_{atom} is the potential of the atom in the ground state, $V_{atom}^{f_\alpha=-1/2}$ is the potential of the atom with the level α occupied, $\theta(r)$ is a trimming function, CUT is the cut radius and A is a scale factor named amplitude.

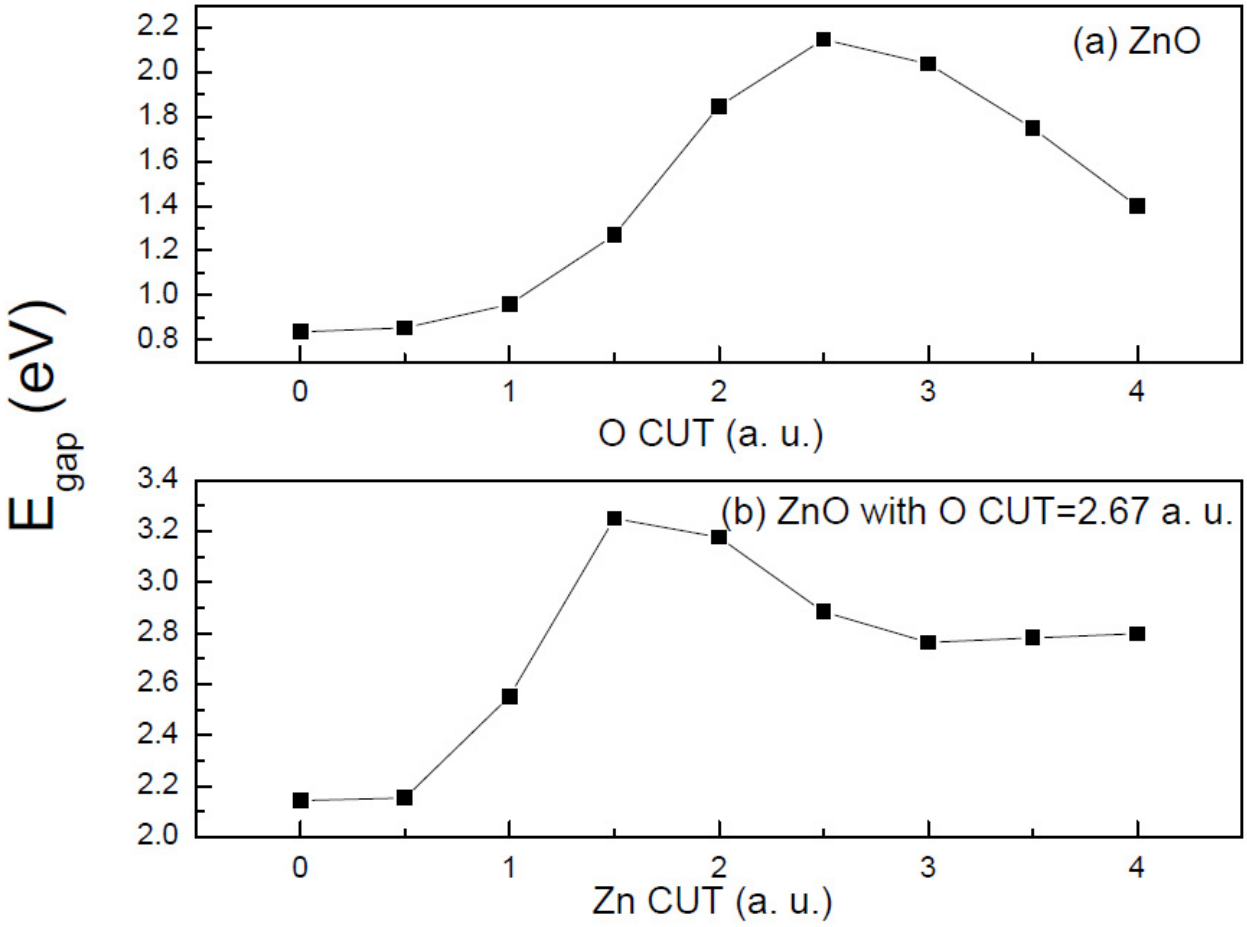

Fig. 4: Fig 3. The figure depicts the choice of CUT for O and Zn in the case of ZnO. First we maximize the gap varying CUT^O , then we vary CUT^{Zn} to reach a band gap value near the correct value[Page 1, 2].

Finnaly, since the atoms repeats in each unit cell, the potential $V_{1/2e}$ is periodic, joining this information with the fact that $V_{crystal}$ is periodic, one can conclude that $V_{crystal}^{-1/2}$ is periodic, which implies that the boundary conditions remain periodic and the Kohn-Sham calculations can be applied to the system.

1.5 The CUT parameter

As observed for several 3D and 2D materials[Page 1, 2][Page 1, 12], one may observe that a CUT parameter has a strong dependence on the considered element where the half occupation is applied, as shown in Figure 4 for two-dimensional materials, which is consistent with the environment neglect and the isolated atom approximation for the self-energy potential.The significant differences between CUT parameters for one element in distinct materials is explained by the difference between bond lengths in the studied materials. In general, materials with smaller first-neighbor lengths exhibit smaller cutoff parameters. A linear relation between the CUT parameter and the bond lengths (d) is observed for several classes of 2D materials[Page 1, 12] and 3D materials[Page 1, 1].

1.6 Where to perform semi-occupation?

There are two types of correction, simple and fractional, and they must be performed in the last valence band (VBM) and the first conduction band (CBM). The choice of which correction cannot be made blindly, it requires an analysis of the band's composition. To explain these two corrections, suppose that we have a matrix where the atoms of the unit cell are represented as lines and the types of atomic orbitals $(s, p, d, f...)$ as columns , each value a_{ij} represents, in percentage, how much that orbital j of a given atom i contributes to the total module of the wave function.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots \\ \vdots & \ddots & \\ a_{N1} & & a_{NK} \end{bmatrix}$$

Where:

$$\sum_{i=1}^{N}\sum_{j=1}^{K} a_{ij} = 100$$

1.6.1 Simple correction

The simple correction method is applied when an index a_{ij} mainly represents the composition of the band, so that the influence of the other orbitals is negligible. Thus, the correction of half an electron is done only in the orbital j of the atom i.

1.6.2 Fractional correction

The fractional correction method is applied when different atomic orbitals have a significant influence in the composition of the band, it can be observed in the conduction bands of Figure 5, where the p and d orbitals compose the band simultaneously. To distribute half an electron, a threshold is chosen ϵ, which represents the minimum value of a_{ij} considered in the correction. Given these values,the half an electron will be divided among the atoms, proportionally to the coefficient a_{ij}.

[10] C. A. Ataide, R. R. Pelá, M. Marques, L. K. Teles, J. Furthmüller, and F. Bechstedt *Phys. Rev. B 95, 045126 – Published 17 January 2017 <https://journals.aps.org/prb/abstract/10.1103/PhysRevB.95.045126>*

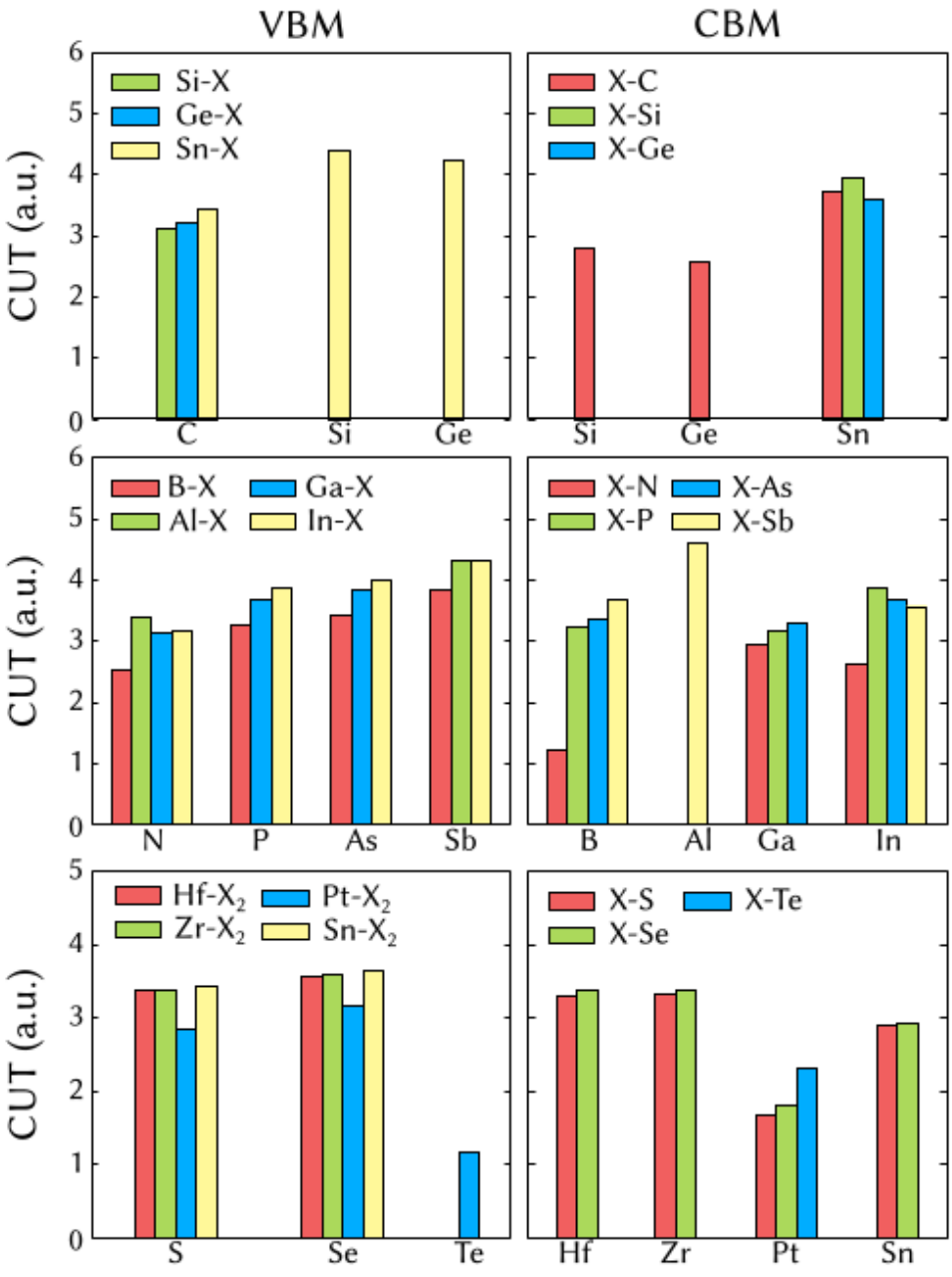

Fig. 5: Fig 4. Cutoff parameter comparison for a selected set of 2D materials. The cutoff parameters of the VBM (CBM) states on the anions (cations) are represented on the left (right)[Page 1, 12].

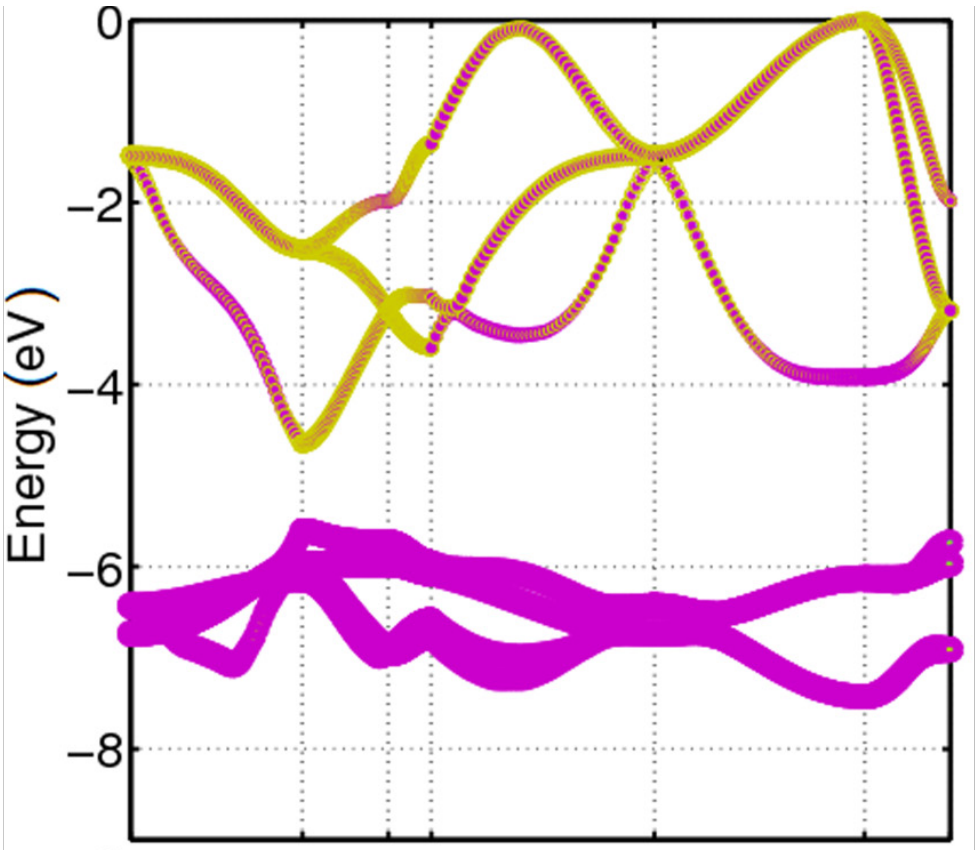

Fig. 6: Fig 5. Orbital character for CdO valence bands. The character p is represented in yellow and the character d in a magnet[10].

1.6.3 Is conduction band correction always necessary?

In many cases, the correction in the valence band already returns satisfactory and close enough results, which rules out the need for an additional correction in the conduction band.

1.6.4 Final considerations

After applying the correction, the optimum cut must be found for each corrected atom to find the final value for the gap.

1.7 References

CHAPTER

TWO

INSTALLATION

The minushalf CLI can be easily installed by PyPI with the following command.

```
$ pip install minushalf
```

2.1 Requirements

The minushalf CLI was built in order to automate the application of the DFT -1/2[1] method. Thus, as the method requires the calculation of eigenvalues for each kpoint and band, it is necessary to install some software that performs ab initio calculations. Currently, the following softwares are supported by the program:

- VASP[2][3]

2.2 References

[1] L. G. Ferreira, M. Marques, and L. K. Teles, AIP Adv. 1, 032119 (2011).

[2] G. Kresse and J. Furthmüller, Phys. Rev. B 54, 11169 (1996).

[3] G. Kresse and J. Furthmüller, Comput. Mater. Sci. 6, 15 (1996).

CHAPTER

THREE

COMMANDS

3.1 minushalf vbm-character

```
$ minushalf vbm-character --help

  Usage: minushalf vbm-character [OPTIONS]

  Uses output files from softwares that perform ab initio calculations to
  discover the last valence band (VBM) and extract, in percentage, its
  character corresponding to each orbital type (s, p, d, ... ). The
  names of the files required for each software are listed below, it is
  worth mentioning that their names cannot be modified.

  VASP: PROCAR, EIGENVAL, vasprun.xml

  Options:
  -s, --software [VASP]  Specifies the software used to perform ab initio calculations.
                         [default: VASP]

  -b, --base-path PATH   Path to folder where the relevant files are located.

  --help                 Show this message and exit.
```

3.1.1 Examples

To demonstrate the command usage, one calculated the character of the last valence band of GaN-2d .

VASP

The following input files were used:

```
GaN POSCAR
1.00000000000000
3.2180000000000004    0.0000000000000000    0.0000000000000000
-1.6090000000000002    2.7868697493783232    0.0000000000000000
0.0000000000000000    0.0000000000000000   20.0000000000000000
Ga   N
1     1
```

(continues on next page)

(continued from previous page)

```
Selective dynamics
Direct
0.3333000000000013  0.6666600000000003  0.5000000000000000   T   T   F
0.0000000000000000  0.0000000000000000  0.5000000000000000   F   F   F
```

```
PREC = Normal
EDIFF = 0.0001
ENCUT = 500.0
ISMEAR= -5
ISTART = 0
LREAL = .FALSE.
LORBIT=11
```

```
Kpoints
0
Gamma
12 12 1
0.0 0.0 0.0
```

Electronic properties are investigated within the DFT by applying the Perdew-Burke-Ernzerhof (PBE) functional within the general gradient approximation (GGA)[2]. After running the VASP program, the `minushalf vbm-character` command returned the following output:

```
$ minushalf vbm-character -s VASP

_ __ ___ (_)_ __  _   _ ___| |__   __ _| |/ _|
| '_ ` _ \| | '_ \| | | / __| '_ \ / _` | | |_
| | | | | | | | | | |_| \__ \ | | | (_| | |  _|
|_| |_| |_|_|_| |_|\__,_|___/_| |_|\__,_|_|_|


|    |   d |   p |   s |
|:---|----:|----:|----:|
| Ga |  11 |   0 |   0 |
| N  |   0 |  89 |   0 |
_____ _   _ ____
| ____| \ | |  _ \
|  _| |  \| | | | |
| |___| |\  | |_| |
|_____|_| \_|____/
```

As expected for honeycomb binary materials based on III-V elements, The VBM states located at the Kpoint are integrally derived from the anion p_z atomic orbitals[1].

[2] J. P. Perdew, M. Ernzerhof, and K. Burke, J. Chem. Phys. 105, 9982 (1996).

[1] I. Guilhon, D. S. Koda, L. G. Ferreira, M. Marques, and L. K. Teles Phys. Rev. B 97, 045426 .

3.1.2 References

3.2 minushalf cbm-character

```
$ minushalf cbm-character --help

  Usage: minushalf cbm-character [OPTIONS]

  Uses output files from softwares that perform ab initio calculations to
  discover the first conduction band (CBM) and extract, in percentage, its
  character corresponding to each orbital type (s, p, d, ... ). The
  names of the files required for each software are listed below, it is
  worth mentioning that their names cannot be modified.

  VASP: PROCAR, EIGENVAL, vasprun.xml

  Options:
  -s, --software [VASP]  Specifies the software used to perform ab initio calculations.
                         [default: VASP]

  -b, --base-path PATH   Path to folder where the relevant files are located.

  --help                 Show this message and exit.
```

3.2.1 Examples

To demonstrate the command usage, one calculated the character of the first conduction band of SiC-2d .

VASP

The following input files were used:

```
SiC POSCAR
1.0
3.100032 -0.000007 0.000001
-1.550022 2.684696 -0.000002
0.000006 -0.000010 20.000000
Si C
1 1
Selective dynamics
direct
0.666667 0.333335 0.295447 T T F
0.000000 0.999998 0.295392 F T F
```

```
PREC = Normal
EDIFF = 0.0001
ENCUT = 500.0
ISMEAR= -5
ISTART = 0
```

(continues on next page)

(continued from previous page)

```
LREAL = .FALSE.
LORBIT=11
```

```
Kpoints
0
Gamma
12 12 1
0.0 0.0 0.0
```

Electronic properties are investigated within the DFT by applying the Perdew-Burke-Ernzerhof (PBE) functional within the general gradient approximation (GGA)[2]. After running the VASP program, the `minushalf cbm-character` command returned the following output:

```
$ minushalf cbm-character -s VASP


           _                  _            _  __
 _ __ ___ (_)_ __  _   _ ___| |__   __ _| |/ _|
| '_ ` _ \| | '_ \| | | / __| '_ \ / _` | | |_
| | | | | | | | | | |_| \__ \ | | | (_| | |  _|
|_| |_| |_|_|_| |_|\__,_|___/_| |_|\__,_|_|_|


|    |   d |   p |   s |
|:---|----:|----:|----:|
| Si |   0 |  85 |   0 |
| C  |   0 |  15 |   0 |
_____ _   _ ____
| ____| \ | |  _ \
|  _| |  \| | | | |
| |___| |\  | |_| |
|_____|_| \_|____/
```

As expected for honeycomb binary materials based on the IV group, The CBM states located at the Kpoint can be associated with the p_z orbitals of the least electronegative element[1].

3.2.2 References

3.3 `minushalf band-character`

```
$ minushalf band-character --help

  Usage: minushalf band-character [OPTIONS] KPOINT BAND

  Uses output files from softwares that perform ab initio calculations to
```

(continues on next page)

[2] J. P. Perdew, M. Ernzerhof, and K. Burke, J. Chem. Phys. 105, 9982 (1996).

[1] I. Guilhon, D. S. Koda, L. G. Ferreira, M. Marques, and L. K. Teles Phys. Rev. B 97, 045426 .

(continued from previous page)

```
  read projections in a specific kpoint band and extract, in percentage,
  its   character corresponding to each orbital type (s, p, d, ... ). The
  names of the files required for each software are listed below, it is
  worth mentioning that their names cannot be modified.

  VASP: PROCAR, EIGENVAL, vasprun.xml

  Options:
  -s, --software [VASP]  Specifies the software used to perform ab initio calculations.
                         [default: VASP]

  -b, --base-path PATH   Path to folder where the relevant files are located.

  --help                 Show this message and exit.
```

3.3.1 Examples

To demonstrate the command usage, one calculated the character of the sixth band of the second kpoint of SiC-2d .

VASP

The following input files were used:

```
SiC POSCAR
1.0
3.100032 -0.000007 0.000001
-1.550022 2.684696 -0.000002
0.000006 -0.000010 20.000000
Si C
1 1
Selective dynamics
direct
0.666667 0.333335 0.295447 T T F
0.000000 0.999998 0.295392 F T F
```

```
PREC = Normal
EDIFF = 0.0001
ENCUT = 500.0
ISMEAR= -5
ISTART = 0
LREAL = .FALSE.
LORBIT=11
```

```
Kpoints
0
Gamma
12 12 1
0.0 0.0 0.0
```

Electronic properties are investigated within the DFT by applying the Perdew-Burke-Ernzerhof (PBE) functional within the general gradient approximation (GGA)[1]. After running the VASP program, the `minushalf band-character` command returned the following output:

```
$ minushalf band-character 2 6 -s VASP


          _                  _           _  __
_ __ ___ (_)_ __  _   _ ___| |__   __ _| |/ _|
| '_ ` _ \| | '_ \| | | / __| '_ \ / _` | | |_
| | | | | | | | | | |_| \__ \ | | | (_| | |  _|
|_| |_| |_|_|_| |_|\__,_|___/_| |_|\__,_|_|_|


|    |   d |   p |   s |
|:---|----:|----:|----:|
| Si |   0 |   3 |   0 |
| C  |   0 |  97 |   0 |
 _____ _   _ ____
| ____| \ | |  _ \
|  _| |  \| | | | |
| |___| |\  | |_| |
|_____|_| \_|____/
```

As one can see, band 6 of kpoint 2 has a strong character of carbon $p - type$ orbitals.

3.3.2 References

3.4 minushalf band-gap

```
$ minushalf band-gap --help

  Usage: minushalf band-gap [OPTIONS]

  Uses output files from softwares that perform ab initio calculations to
  provide the locations of VBM, CBM and the Gap value in electronvolts.The
  names of the files required for each software are listed below, it is
  worth mentioning that their names cannot be modified.

  VASP: PROCAR, EIGENVAL, vasprun.xml

  Options:
  -s, --software [VASP]  Specifies the software used to perform ab initio calculations.
                         [default: VASP]

  -b, --base-path PATH   Path to folder where the relevant files are located.

  --help                 Show this message and exit.
```

[1] J. P. Perdew, M. Ernzerhof, and K. Burke, J. Chem. Phys. 105, 9982 (1996).

3.4.1 Examples

To demonstrate the command usage, one calculated the positions of CBM, VBM and the Gap value of SiC-2d .

VASP

The following input files were used:

```
SiC POSCAR
1.0
3.100032 -0.000007 0.000001
-1.550022 2.684696 -0.000002
0.000006 -0.000010 20.000000
Si C
1 1
Selective dynamics
direct
0.666667 0.333335 0.295447 T T F
0.000000 0.999998 0.295392 F T F
```

```
PREC = Normal
EDIFF = 0.0001
ENCUT = 500.0
ISMEAR= -5
ISTART = 0
LREAL = .FALSE.
LORBIT=11
```

```
Kpoints
0
Gamma
12 12 1
0.0 0.0 0.0
```

Electronic properties are investigated within the DFT by applying the Perdew-Burke-Ernzerhof (PBE) functional within the general gradient approximation (GGA)[2]. After running the VASP program, the `minushalf band-gap` command returned the following output:

```
$ minushalf band-gap -s VASP


           _                  _           _  __
 _ __ ___ (_)_ __  _   _ ___| |__   __ _| |/ _|
| '_ ` _ \| | '_ \| | | / __| '_ \ / _` | | |_
| | | | | | | | | | |_| \__ \ | | | (_| | |  _|
|_| |_| |_|_|_| |_|\__,_|___/_| |_|\__,_|_|_|


VBM: Kpoint 48, band 4 and eigenval -3.683426
CBM: Kpoint 68, band 5 and eigenval -1.141163
```

(continues on next page)

[2] J. P. Perdew, M. Ernzerhof, and K. Burke, J. Chem. Phys. 105, 9982 (1996).

(continued from previous page)

```
Gap: 2.542eV
 _____ _   _ ____
| ____| \ | |  _ \
|  _| |  \| | | | |
| |___| |\  | |_| |
|_____|_| \_|____/
```

As expected, the Gap found is worth 2,542eV[1] .

3.4.2 References

3.5 minushalf run-atomic

The atomic software used in this command is a modified version of the program ATOM by professor Luiz Guimarães Ferreira. The respective modifications are listed below:

- In this version the maximum number of interactions ('maxit') is read, just after the valence orbitals. Thus, the input files INP.pg and INP.pt must be renamed to INP.
- Potential was generated to be added to the pseudopotential given by the program. The potential to be added is in the 'adiciona' file. The following instruction verifies that the file exists and, if it exists, is opened and read.

```
inquire(file='adiciona',exist=lexist)
if(lexist)  open(unit=21,file='adiciona')
```

- Creates 'VTOTAL' file with the potential related to the Schrödinger or Dirac equation.
- Creates the psfun.Guima file with the wave functions ae, pg and pt.
- The pseudopotential averages are calculated for r^2 e r^4. Electrostatic auto energy calculation is also done to valence orbitals.

```
$ minushalf run-atomic --help
Usage: minushalf run-atomic [OPTIONS]

Run the atomic program. The program used is a
modified version of ATOM by professor Luiz Guimarães Ferreira

Requires:

    INP: The input file for the calculation.

Returns:

    VTOTAL.ae: Contains the atom potential.

    OUT: Contains detailed information about the run.

    AECHARGE: Contains in four columns values of r, the "up" and "down"
```

(continues on next page)

[1] I. Guilhon, D. S. Koda, L. G. Ferreira, M. Marques, and L. K. Teles Phys. Rev. B 97, 045426 .

(continued from previous page)

```
    parts of the total charge density, and the total core
    charge density (the charges multiplied by 4r^2 ).

    CHARGE: is exactly identical to AECHARGE and is generated for
    backwards compatibility.

    RHO: Like CHARGE, but without the 4r 2 factor

    AEWFNR0...AEWFNR3: All-electron valence wavefunctions as function of
    radius, for s, p, d and f valence orbitals (0,1, 2, 3, respectively - some channels␣
↪might not be available).
    They include a factor of r, the s orbitals also going to zero at the
    origin.

Options:
--quiet
--help   Show this message and exit.
```

3.6 minushalf occupation

```
$ minushalf occupation --help
Usage: minushalf occupation [OPTIONS] ORBITAL_QUANTUM_NUMBER
                            [OCCUPATION_PERCENTUAL]

Perform fractional occupation on the atom and generate the pseudopotential
for this occupation. The occupation can subtract any fraction of the
electron between 0 and 0.5, half occupation is the default.

    Requires:

        ORBITAL_QUANTUM_NUMBER: A string that defines the orbital(s) in which
        the occupation will be made, it can assume four values: (0: s | 1:
        p | 2: d | 3: f). if going to modify multiple orbitals, pass a
        string with numbers separated by commas : ("0,1,2,3").

        OCCUPATION_PERCENTUAL: A string that defines percentual of half an
        electron to be used in the occupation. The default is 100%, which
        states for 0.5e. For multiple occupations in different orbitals, pass
        a string separated by commas ("100,50,40,100"). For simplicity, to
        avoid the excessive repetition of the number 100, just replace the
        number with * ("*,30,*"). If this argument is not used, the occupation
        of half electron will be made for all orbitals passed as arguments.

        INP: Input file of the run-atomic command.

    Returns:

        INP_OCC : Input file modified for fractional occupation.

        INP.ae: A copy of the input file for the calculation.
```

(continues on next page)

(continued from previous page)

```
        VTOTAL_OCC: Contains the atom potential for fractional occupation.

        OUT: Contains detailed information about the run.

        AECHARGE: Contains in four columns values of r, the "up" and "down"
        parts of the total     charge density, and the total core charge
        density (the charges multiplied by 4r^2 ).

        CHARGE: is exactly identical to AECHARGE and is generated for
        backwards compatibility.

        RHO: Like CHARGE, but without the 4r^2 factor

        AEWFNR0...AEWFNR3: All-electron valence wavefunctions as function of
        radius, for s, p, d, and f valence orbitals (0, 1, 2, 3,
        respectively - some channels might not be available). They include
        a factor of r, the s orbitals also going to zero at the origin.

Options:
--quiet
--help   Show this message and exit.
```

3.6.1 Example of occupation in only one orbital

Suppose one need to generate a pseudopotential for the Ga atom with the occupation of half an electron in the p orbital. The following command can be used for this purpose:

```
$ minushalf occupation 1 100
```

Where the first argument represents the azimuthal quantum number for the p orbital and the second argument represents the fraction of half an electron that will be used in the occupation.

Initially, only the INP input file, which is shown below, needs to be provided.

```
      ae      Ga
 n=Ga c=pb
       0.0       0.0       0.0       0.0       0.0       0.0
    5    4
    4    0     2.000     0.000
    4    1     1.000     0.000
    3    2    10.000     0.000
    4    3     0.000     0.000
100 maxit
```

After running the command, the following files are created

```
.
├── AECHARGE
├── AEWFNR0
├── AEWFNR1
├── AEWFNR2
```

(continues on next page)

(continued from previous page)

```
├── AEWFNR3
├── CHARGE
├── fort.5
├── INP.ae
├── INP_OCC
├── OUT
├── psfun.guima
├── RHO
├── VTOTAL0
├── VTOTAL2
├── VTOTAL3
└── VTOTAL_OCC
```

Where VTOTAL_OCC represents the pseudopotential for the occupation carried out and the INP_OCC file represents the input file with the occupation of half an electron in the p orbital, as shown below.

```
    ae      Ga
 n=Ga c=pb
       0.0       0.0       0.0       0.0       0.0       0.0
    5    4
    4    0     2.000     0.000
    4    1     0.500     0.000
    3    2    10.000     0.000
    4    3     0.000     0.000
100 maxit
```

3.6.2 Example of occupation in multiple orbitals

Now, figure out a scenario where one need to generate a pseudopotential for the Ga atom with the electron medium equally divided between the orbitals p and d. The following command can be used for this purpose:

```
$ minushalf occupation '1,2' '50,50'
```

Where the first argument represents the azimuthal quantum numbers for the orbitals p and d, while the second argument represents the fraction of half an electron that will be used for each orbital. As the half an electron will be shared equally between the two orbitals, the fractions chosen will be 50% for both, which corresponds to an occupancy of a quarter of an electron for the orbitals.

Initially, only the INP input file, which is shown below, needs to be provided.

```
      ae      Ga
 n=Ga c=pb
       0.0       0.0       0.0       0.0       0.0       0.0
    5    4
    4    0     2.000     0.000
    4    1     1.000     0.000
    3    2    10.000     0.000
    4    3     0.000     0.000
100 maxit
```

After executing the command, the following files are created

```
.
├── AECHARGE
├── AEWFNR0
├── AEWFNR1
├── AEWFNR2
├── AEWFNR3
├── CHARGE
├── fort.5
├── INP.ae
├── INP_OCC
├── OUT
├── psfun.guima
├── RHO
├── VTOTAL0
├── VTOTAL2
├── VTOTAL3
└── VTOTAL_OCC
```

Where VTOTAL_OCC represents the pseudopotential for the occupation carried out and the INP_OCC file represents the input file with the occupation in the p and d orbitals, as shown below.

```
   ae      Ga
 n=Ga c=pb
       0.0       0.0       0.0       0.0       0.0       0.0
    5    4
    4    0     2.000     0.000
    4    1     0.750     0.000
    3    2     9.750     0.000
    4    3     0.000     0.000
100 maxit
```

3.7 minushalf create-input

This command creates the input files for the run-atomic command. Check *here* the list of available atoms.

```
$ minushalf create-input --help
Usage: minushalf create-input [OPTIONS] CHEMICAL_SYMBOL

  Create the input file for the run-atomic command.

  Requires:

      CHEMICAL_SYMBOL: Chemical symbol of the atom (H, He, Na, Li...). Check the list
                       of available atoms in the docs.

  Returns:

      INP: The input file for run-atomic command

Options:
  -e, --exchange_correlation_code [ca|wi|hl|gl|bh|pb|rp|rv|bl]
```

(continues on next page)

(continued from previous page)

```
                                  Represents the functional of exchange and
                                  correlation,it can assume the following
                                  values:

                                    ca: Ceperley-Alder

                                    wi: Wigner

                                    hl: Hedin-Lundqvist

                                    gl: Gunnarson-Lundqvist

                                    bh: Von Barth-Hedin

                                    pb: PBE scheme by Perdew, Burke, and
                                    Ernzerhof

                                    rp: RPBE scheme by Hammer, Hansen, and
                                    Norskov

                                    rv: revPBE scheme by Zhang and Yang

                                    bl: BLYP (Becke-Lee-Yang-Parr) scheme

                                      [default: pb]

  -c, --calculation_code [ae]     Represents calculation code,it can assume
                                  the following values:

                                  ae: All electrons  [default: ae]

  -m, --maximum_iterations INTEGER RANGE
                                  Maximum number of iterations performed by
                                  the atomic program  [default: 100]

  -f, --filename TEXT             Name of the created file  [default: INP]
  --quiet
  --help                          Show this message and exit.
```

3.8 minushalf correct-potfile

```
$ minushalf correct-potfile --help
Usage: minushalf correct-potfile [OPTIONS]

Generate the occupied atomic potential file used for ab initio calculations.

Requires:

    VTOTAL.ae: pseudopotential of the atom with all electrons

```

(continues on next page)

(continued from previous page)

```
    VTOTAL_OCC: pseudopotential of the occupied atom

    INP_OCC: Input file for the run-atomic command of the occupied atom

    The command also needs the potential files used by the chosen software:

        VASP: POTCAR (This name can't be changed)

Generates:

    POTFILEcut${CUT_VALUE} (If amplitude is equal to 1.0)

    POTFILEcut${CUT_VALUE}A${AMPLITUDE_VALUE} (If amplitude is different from 1.0)

Options:
--quiet
-b, --base_potfile_path PATH    Path to the folder containing the potential
                                file

-v, --vtotal_path PATH          Path to the pseudopotential file
                                generated by the atomic program for the atom
                                with all electrons.  [default: VTOTAL.ae]

-o, --vtotal_occupied_path PATH
                                Path to the pseudopotential file
                                generated by the atomic program for the
                                occupied atom.  [default: VTOTAL_OCC]

-s, --software [VASP]           Specifies the software used to make ab initio␣
↪calculations.
                                [default: VASP]

-c, --correction [VALENCE|CONDUCTION]
                                Indicates whether the correction should be
                                made in the valence band or the
                                conduction band.  [default: VALENCE]

-C, --cut TEXT                  distance value used to cut the potential
                                generated artificially by fractional atomic
                                occupation, it can be passed in two ways:

                                unique value : float or integer. Ex: 1.0

                                range: ␣
↪begin(float|integer):pass(float|integer):end(float|integer). Ex: 1.0:0.1:2.0
                                [default: 2.0]

-a, --amplitude FLOAT RANGE     Scaling factor to be used to correct the artificially␣
↪generated potential.
                                In the vast majority of cases, the amplitude value is 1.
↪0. However, there are some
                                special cases where this value needs to be adjusted.␣
```

(continues on next page)

(continued from previous page)

```
↪Therefore, we recommend that
                                you do not change this value unless you know exactly␣
↪what you are doing  [default: 1.0]

--help                          Show this message and exit.
```

To consult a case where changing the amplitude value is necessary, check the reference[1].

3.8.1 References

3.9 minushalf execute

This command automates the use of the DFT -1/2. It uses the Nelder-Mead algorithm[1] to find the optimal values of CUT(S) and generates a text file with all the respective CUTS and the final value of the gap.

```
$ minushalf execute --help
Usage: minushalf execute [OPTIONS]

Uses the Nelder-Mead method to find the optimal values for the CUT(S) and,
finally, find the corrected Gap value. This command uses external software
to perform ab initio calculations, so it must be installed in order to
perform the command. Check the docs for an list of the softwares supported
by the CLI.

    Requires:

        minushalf.yaml : Parameters file. Check the docs
        for a more detailed description.

        ab_initio_files: Files needed to perform the ab initio
                         calculations. They must be in the same
                         directory as the input file minushalf.yaml

        potential_folder: Folder with the potential files for each atom in
                          the crystal. The files must be named in the following pattern
                          ${POTENTIAL_FILE_NAME}.${LOWERCASE_CHEMICAL_SYMBOL}

    Returns:

        minushalf_results.dat : File that contains the optimal
                                values of the cuts and the final
                                value of the Gap.

        corrected_valence_potfiles: Potential files corrected with opti-mum valence cuts.

        corrected_conduction_potfiles: Potential files corrected with optimum conduction␣
```

(continues on next page)

[1] C. A. Ataide, R. R. Pelá, M. Marques, L. K. Teles, J. Furthmüller, and F. Bechstedt Phys. Rev. B 95, 045126 – Published 17 January 2017.

[1] Nelder, John A.; R. Mead (1965). A simplex method for function minimization. Computer Journal. 7 (4): 308–313.

(continued from previous page)

```
↪cuts.

Options:
--quiet
--help   Show this message and exit.
```

3.9.1 minushalf.yaml

`minushalf.yaml` is the input file for the command `execute`, each of its tags and default values are described below.

software tag

This tag specifies the software that to perform ab initio calculations. For now, the command supports the following values for the software tag:

- VASP (Default value)

Currently, minushalf only supports one software, but one hope to add more soon.

```
software: VASP
```

vasp tag

The vasp tag specifies the command needed to perform first principles calculations. This tag has the following fields:

- command: Command used to perform first principles calculations. (Default: ['mpirun','vasp'])

The `mpirun` command is used for convenience and can be overridden depending on the local settings of the user's machine. The example below shows an use of the vasp tag in the `minushalf.yaml file`:

```
vasp:
    command: ['mpirun','-np','6','vasp']
```

atomic_program tag

The atomic_program tag is a set of various informations that specifies the settings for the atomic program execution. The informations are:

- exchange_correlation_code: Functional of exchange and correlation (Default: pb)
- calculation_code: Calculation code for the atomic program (Default: ae)
- max_iterations: Maximum number of iterations performed by the atomic program (Default: 100)

The values that the exchange_correlation_code and calculation_code tags can assume are listed below:

exchange_correlation_code

- ca: Ceperley-Alder
- wi: Wigner
- hl: Hedin-Lundqvist
- gl: Gunnarson-Lundqvist
- bh: Von Barth-Hedin
- pb: PBE scheme by Perdew, Burke, and Ernzerhof
- rp: RPBE scheme by Hammer, Hansen, and Norskov
- rv: revPBE scheme by Zhang and Yang
- bl: BLYP (Becke-Lee-Yang-Parr) scheme

calculation_code

- ae: All electrons

Below follows an example of the atomic_program tag in the `minushalf.yaml` file:

```
atomic_program:
    exchange_correlation_code: wi
    calculation_code: ae
    max_iterations: 200
```

correction tag

The correction tag specifies how the DFT -1/2
method is executed. It contains the following parameters:

- correction_code: Code thar specifies the potential correction (Default: v)
- potfiles_folder: Path to folder that holds the potential files for each atom. The files must be named in the following pattern `${POTENTIAL_FILE_NAME}.${LOWERCASE_CHEMICAL_SYMBOL}` (Default: minushalf_potfiles)
- amplitude: Scaling factor to be used to correct the artificially generated potential. In the vast majority of cases, the amplitude value is 1.0. However, there are some special cases where this value needs to be adjusted[5]. Therefore, we recommend that you do not change this value unless you know exactly what you are doing (Default: 1.0)
- valence_cut_guess: Initial Guess for the Nelder-Mead algorithm for cut in valence correction. If not provided, the default value of $0.15 + 0.84d$[6] will be used for each optimization, where d is the distance of the nearest neighbor in the unit cell. (Default: $0.15 + 0.84d$)
- conduction_cut_guess: Initial Guess for the Nelder-Mead algorithm for cut in valence correction. If not provided, the default value of $0.15 + 0.84d$ will be used will be used for each optimization, where d is the distance of the nearest neighbor in the unit cell. (Default: $0.15 + 0.84d$)
- tolerance: Absolute tolerance for the result of the Nelder-Mead algorithm (Default: 0.01)

[5] C. A. Ataide, R. R. Pelá, M. Marques, L. K. Teles, J. Furthmüller, and F. Bechstedt Phys. Rev. B 95, 045126 – Published 17 January 2017.

[6] L. G. Ferreira, M. Marques, and L. K. Teles, Phys. Rev. B 78, 125116 (2008).

- fractional_valence_treshold: *Treshold* ϵ for fractional valence correction (Default: 10).
- fractional_conduction_treshold: *Treshold* ϵ for fractional conduction correction (Default: 10).
- overwrite_vbm: In some special cases[Page 29, 6], it is necessary to consider another band as the VBM. This tag is made for these situations. It is necessary to inform the kpoint and the band number that specifies the band location. The program immediately overwrites the old projection values and uses the new values for DFT -1/2 calculations (Default: No overwrite)
- overwrite_cbm: In some special cases[Page 29, 6], it is necessary to consider another band as the CBM. This tag is made for these situations. It is necessary to inform the kpoint and the band number that specifies the band location. The program immediately overwrites the old projection values and uses the new values for DFT -1/2 calculations (Default: No overwrite)
- inplace: This tag allows you to decide whether all calculations will be done in the root folder or not. It is recommended to pass it as `True` if non-self-consistent calculations are being performed for the Gap calculation, since the program only copies the input files, the output files needed for the non-self-consistent calculation will not be considered (Default: False)
- divide_character: Factor that divides the character of each atom. It is used in cases where all the bonds of an atom are with atoms of the same chemical element, as in crystals of germanium and silicon. This factor is automatically calculated by the program, however this tag will overwrite these values. So use with caution.
- vbm_characters: This tag allows the character values of the last valence band to be provided manually. It is recommended that this tag be used with caution as it can severely impact your results.
- cbm_characters: This tag allows the character values of the first conduction band to be provided manually. It is recommended that this tag be used with caution as it can severely impact your results.

The values that the correction_code tag can assume are listed below:

correction_code

- v: Simple valence correction
- vf: Fractional valence correction
- c: Simple conduction correction
- cf: Fractional conduction correction
- vc: Simple valence and simple conduction corrections
- vfc: Fractional valence and simple conduction corrections
- vcf: Simple valence and fractional conduction corrections
- vfcf: Fractional valence and fractional conduction corrections

The example below shows an use of correction tag in the `minushalf.yaml` file:

```
correction:
    correction_code: vf
    potfiles_folder: ../potcar
    amplitude: 3.0
    valence_cut_guess:  [["C","p",2.0],["C","s",1.5]]  # initial guesses for each␣
↪orbital that contributes to the valence band
    conduction_cut_guess:  [["Si","s",1.0],["Si","p",3.5]]
    tolerance: 0.01
    fractional_valence_treshold: 15
```

(continues on next page)

(continued from previous page)

```
    fractional_conduction_treshold: 23
    overwrite_vbm: [4,9] # Kpoint and band number, respectively
    overwrite_cbm: [1,3] # Kpoint and band number, respectively
    inplace: False
    divide_character: [["C",p,1]] # divide the p character of C with one more atom
    vbm_characters: [["C","s",34],["C","s",50]] # Overwrite the characters mannualy
    cbm_characters: [["C","s",34],["C","s",50]]
```

3.9.2 Examples

To demonstrate the command usage, one apply the simple valence and simple conduction correction on SiC-2d .

VASP

To execute the command, the files must be provided in the following structure:

```
.
├── INCAR
├── KPOINTS
├── minushalf.yaml
├── POSCAR
├── POTCAR
└── potcars
    ├── POTCAR.c
    └── POTCAR.si
```

For the input file, the following initial settings were chosen:

```
software: VASP
vasp:
    command: ['mpirun','-np','4','vasp']

correction:
    correction_code: vc
    potfiles_folder: ./potcars
    valence_cut_guess: [["C","p" 3.20]]
    conduction_cut_guess: [["Si","p", 3.0]]
```

After executing the command, one can view the result in the file `minushalf_results.dat`. he file contains information on the values obtained in the optimization of the CUT and the resulting band energy Gap (in eV).

```
Valence correction cuts:
        (C,p):3.13 a.u
---------------------------------------------------------------
Conduction correction cuts:
        (Si,p):2.77 a.u
---------------------------------------------------------------
GAP: 4.37eV
```

For comparison purposes, the table below shows the values obtained by the method compared with Pure GGA, functional hybrids and GW.

Table 1: SiC-2D band energy gap (in eV)

GGA	Hybrid	GW	DFT -1/2
2.54	3.35,3.46[2]	4.19[3],4.42[4]	4.37

3.9.3 References

[2] Y. Rao, S. Yu, and X.-M. Duan, Phys. Chem. Chem. Phys. 19, 17250 (2017).

[3] H. Sahin, S. Cahangirov, M. Topsakal, E. Bekaroglu, E. Akturk, R. T. Senger, and S. Ciraci, Phys. Rev. B 80, 155453 (2009).

[4] H. C. Hsueh, G. Y. Guo, and S. G. Louie, Phys. Rev. B 84, 085404 (2011).

CHAPTER

FOUR

API DOCUMENTATION

4.1 Subpackages

4.1.1 minushalf.commands package

Submodules

minushalf.commands.band_character module

Aims to show how the band in a specific k-point is composed by the orbitals of each atom.

minushalf.commands.band_gap module

Command to read band gap

minushalf.commands.cbm_character module

Aims to show how the last conduction band and the first valence band are composed by the orbitals of each atom.

minushalf.commands.correct_potfile module

Atomic Correct potential file

minushalf.commands.create_input module

Makes fractional occupation on INP file

minushalf.commands.execute module

Execute command

`minushalf.commands.execute.`**`get_atoms_list`**`(`*factory:* SoftwaresAbstractFactory`)` → list

Returns atoms_list

minushalf.commands.fractional_occupation module

Makes fractional occupation on INP file

minushalf.commands.run_atomic_program module

Run atomic program

minushalf.commands.vbm_character module

Aims to show how the last valence band are composed by the orbitals of each atom.

Module contents

Commands of the minhushalf cli

4.1.2 minushalf.corrections package

Submodules

minushalf.corrections.correction module

Implements the algorithm that automates the process of vasp correction and optimizes the necessary parameters.

class minushalf.corrections.correction.**DFTCorrection**(*root_folder: str*, *potential_filename: str*, *potential_folder: str*, *exchange_correlation_type: str*, *max_iterations: int*, *software_factory:* SoftwaresAbstractFactory, *runner:* Runner, *calculation_code: str*, *amplitude: float*, *cut_initial_guess: dict*, *tolerance: float*, *input_files: list*, *inplace: bool*, *corrected_potfiles_folder: str*, *correction_type: str*, *band_projection: DataFrame*, *atoms: list*, *is_conduction: bool*, *correction_indexes: dict*, *divide_character: list*)

Bases: *Correction*

An algorithm that realizes corrections for VASP software

execute() → tuple

Execute vasp correction algorithm

property potential_folder: str

Returns: Name of the folder that helds all the potential files initially not corrected.

Module contents

Init for corrections factory

4.1.3 minushalf.data package

Submodules

minushalf.data.calculation_code module

List calculation code options for the INP file

class minushalf.data.calculation_code.**CalculationCode**(*value*)

Bases: Enum

Enum type for the calculation code

ae = 'ae'

static get_default()

Returns the default value for this parameter

static to_list()

Generate list of available calculation codes

minushalf.data.constants module

Physical constants

class minushalf.data.constants.**Constants**

Bases: object

Class for physical constants used in the program. Contains:

pi: About 3.1415

trimming exponent: exponent used in the trimming function

bohr_radius: The Bohr radius is a physical constant, equal to
the most probable distance between the nucleus and the electron in a hydrogen atom in its ground state.

rydberg: In spectroscopy, the Rydberg constant, symbol for heavy
atoms or for hydrogen, named after the Swedish physicist Johannes Rydberg, is a physical constant relating to the electromagnetic spectra of an atom

property bohr_radius

Bohr radius

property pi_constant

Constant PI

property rydberg

Rydberg constant

property trimming_exponent

Expoent for trimming function

minushalf.data.correction_code module

Enum type for correction codes used in minushalf.yaml

class minushalf.data.correction_code.**CorrectionCode**(*value*)

Bases: Enum

Enum type for the correction codes

c = 'c'

cf = 'cf'

static get_default()

Returns the default value for this parameter

static to_list()

Generate list of available correction codes

v = 'v'

vc = 'vc'

vcf = 'vcf'

vf = 'vf'

vfc = 'vfc'

vfcf = 'vfcf'

minushalf.data.electronic_distribution module

List eletronic distribution for all chemical elements

class minushalf.data.electronic_distribution.**ElectronicDistribution**(*value*)

Bases: Enum

Enum type for the electronic distributions of atoms

Ac = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 1.000 0.000 \n', ' 5 3 0.000 0.000 \n']

Ag = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 10.000 0.000 \n', ' 4 3 0.000 0.000 \n']

Al = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 1.000 0.000 \n', ' 3 2 0.000 0.000 \n', ' 4 3 0.000 0.000 \n']

Am = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000 \n', ' 5 3 7.000 0.000 \n']

Ar = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 6.000 0.000 \n', ' 3 2 0.000 0.000 \n', ' 4 3 0.000 0.000 \n']

As = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 3.000 0.000 \n', ' 3 2 10.000 0.000 \n', ' 4 3 0.000 0.000 \n']

```
At = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 5.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Au = [' 11 4\n', ' 6 0 1.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

B = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 1.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Ba = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Be = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Bi = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 3.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Bk = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 9.000 0.000 \n']

Br = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 5.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

C = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 2.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Ca = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Cd = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Ce = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 1.000 0.000
\n', ' 4 3 1.000 0.000 \n']

Cf = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 10.000 0.000 \n']

Cl = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 5.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Cm = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 1.000 0.000
\n', ' 5 3 7.000 0.000 \n']

Co = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 7.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Cr = [' 5 4\n', ' 4 0 1.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 5.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Cs = [' 11 4\n', ' 6 0 1.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Cu = [' 5 4\n', ' 4 0 1.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']
```

```
Dy = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 10.000 0.000 \n']

Er = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 12.000 0.000 \n']

Es = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 11.000 0.000 \n']

Eu = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 7.000 0.000 \n']

F = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 5.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Fe = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 6.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Fm = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 12.000 0.000 \n']

Fr = [' 15 4\n', ' 7 0 1.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 0.000 0.000 \n']

Ga = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 1.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Gd = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 1.000 0.000
\n', ' 4 3 7.000 0.000 \n']

Ge = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 2.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

H = [' 0 4\n', ' 1 0 1.000 0.000 \n', ' 2 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

He = [' 0 4\n', ' 1 0 2.000 0.000 \n', ' 2 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Hf = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 2.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Hg = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Ho = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 11.000 0.000 \n']

I = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 5.000 0.000 \n', ' 4 2 10.000 0.000 \n',
' 4 3 0.000 0.000 \n']

In = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 1.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Ir = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 7.000 0.000
\n', ' 4 3 14.000 0.000 \n']
```

```
K = [' 5 4\n', ' 4 0 1.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Kr = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 6.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

La = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 1.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Li = [' 1 4\n', ' 2 0 1.000 0.000 \n', ' 2 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Lr = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 1.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 14.000 0.000 \n']

Lu = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 1.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Md = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 13.000 0.000 \n']

Mg = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Mn = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 5.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Mo = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 5.000 0.000 \n',
' 4 3 0.000 0.000 \n']

N = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 3.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Na = [' 3 4\n', ' 3 0 1.000 0.000 \n', ' 3 1 0.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Nb = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 4.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Nd = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 4.000 0.000 \n']

Ne = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 6.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Ni = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 8.000 0.000 \n',
' 4 3 0.000 0.000 \n']

No = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 14.000 0.000 \n']

Np = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 1.000 0.000
\n', ' 5 3 4.000 0.000 \n']

O = [' 1 4\n', ' 2 0 2.000 0.000 \n', ' 2 1 4.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']
```

```
Os = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 6.000 0.000
\n', ' 4 3 14.000 0.000 \n']

P = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 3.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Pa = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 1.000 0.000
\n', ' 5 3 2.000 0.000 \n']

Pb = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 2.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Pd = [' 8 4\n', ' 5 0 0.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Pm = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 5.000 0.000 \n']

Po = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 4.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Pr = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 3.000 0.000 \n']

Pt = [' 11 4\n', ' 6 0 1.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 9.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Pu = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 6.000 0.000 \n']

Ra = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 0.000 0.000
\n', ' 5 3 0.000 0.000 \n']

Rb = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Re = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 5.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Rh = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 8.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Rn = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 6.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Ru = [' 8 4\n', ' 5 0 1.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 7.000 0.000 \n',
' 4 3 0.000 0.000 \n']

S = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 4.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Sb = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 3.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Sc = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 1.000 0.000 \n',
' 4 3 0.000 0.000 \n']
```

```
Se = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 4.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Si = [' 3 4\n', ' 3 0 2.000 0.000 \n', ' 3 1 2.000 0.000 \n', ' 3 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Sm = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 6.000 0.000 \n']

Sn = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 2.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Sr = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 0.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Ta = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 3.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Tb = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 9.000 0.000 \n']

Tc = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 5.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Th = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 2.000 0.000
\n', ' 5 3 0.000 0.000 \n']

Ti = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 2.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Tl = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 1.000 0.000 \n', ' 5 2 10.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Tm = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 13.000 0.000 \n']

U = [' 15 4\n', ' 7 0 2.000 0.000 \n', ' 7 1 0.000 0.000 \n', ' 6 2 1.000 0.000 \n',
' 5 3 3.000 0.000 \n']

V = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 3.000 0.000 \n',
' 4 3 0.000 0.000 \n']

W = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 4.000 0.000 \n',
' 4 3 14.000 0.000 \n']

Xe = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 6.000 0.000 \n', ' 4 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']

Y = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 1.000 0.000 \n',
' 4 3 0.000 0.000 \n']

Yb = [' 11 4\n', ' 6 0 2.000 0.000 \n', ' 6 1 0.000 0.000 \n', ' 5 2 0.000 0.000
\n', ' 4 3 14.000 0.000 \n']

Zn = [' 5 4\n', ' 4 0 2.000 0.000 \n', ' 4 1 0.000 0.000 \n', ' 3 2 10.000 0.000
\n', ' 4 3 0.000 0.000 \n']
```

```
Zr = [' 8 4\n', ' 5 0 2.000 0.000 \n', ' 5 1 0.000 0.000 \n', ' 4 2 2.000 0.000 \n',
' 4 3 0.000 0.000 \n']
```

minushalf.data.exchange_correlation module

List exchange and correlation codes for the INP file

class minushalf.data.exchange_correlation.**ExchangeCorrelation**(*value*)

Bases: Enum

Enum type for exchange and correlation codes

bh = 'bh'

bl = 'bl'

ca = 'ca'

static get_default()

Returns the default value for this parameter

gl = 'gl'

hl = 'hl'

pb = 'pb'

rp = 'rp'

rv = 'rv'

static to_list()

Generate list of exchange and correlation codes

wi = 'wi'

minushalf.data.minushalf_yaml_default_configuration module

Lists minushalf.yml parameters and their default values

class minushalf.data.minushalf_yaml_default_configuration.**AtomicProgramDefaultParams**(*value*)

Bases: Enum

Default value of parameters in the atomic_program tag.

calculation_code = 'ae'

exchange_correlation_code = 'pb'

max_iterations = 100

static to_dict()

Returns a dictionary of default parameters.

static to_list()

Returns a list of default parameters.

class minushalf.data.minushalf_yaml_default_configuration.**CorrectionDefaultParams**(*value=<no_arg>*, *names=None*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Default value of parameters in the correction tag.

amplitude = 1.0

conduction_cut_guess = None

correction_code = 'v'

fractional_conduction_treshold = 9

fractional_valence_treshold = 10

inplace = False

overwrite_cbm = []

overwrite_vbm = []

potfiles_folder = 'minushalf_potfiles'

static to_dict()

Returns a dictionary of default parameters.

static to_list()

Returns a list of default parameters.

tolerance = 0.01

valence_cut_guess = None

class minushalf.data.minushalf_yaml_default_configuration.**MinushalfParams**(*value*)

Bases: Enum

minushalf.yaml parameters.

atomic_program = 'atomic_program'

correction = 'correction'

software = 'software'

static to_dict()

Returns a dictionary with the name of the parameters present in minushalf.yaml.

static to_list()

Returns a list with the name of the parameters present in minushalf.yaml.

class minushalf.data.minushalf_yaml_default_configuration.**VaspDefaultParams**(*value*)

Bases: Enum

Default value of parameters in the vasp tag.

command = ['mpirun', 'vasp']

static to_dict()

Returns a dictionary of default parameters.

static to_list()

Returns a list of default parameters.

minushalf.data.orbital module

List atomic orbitals and their respective groups

class minushalf.data.orbital.**Orbital**(*value*)

Bases: Enum

Enum type for orbitals. The indices are basically the order in which the orbitals are reported in VASP and has no special meaning.

dx2 = 8

dxy = 4

dxz = 7

dyz = 5

dz2 = 6

f0 = 12

f1 = 13

f2 = 14

f3 = 15

f_1 = 11

f_2 = 10

f_3 = 9

px = 3

py = 1

pz = 2

s = 0

class minushalf.data.orbital.**OrbitalType**(*value*)

Bases: Enum

Enum type for orbital type. Indices are basically the azimuthal quantum number, l.

d = 2

f = 3

p = 1

s = 0

minushalf.data.periodic_table module

Enum class for all elements of the periodic table

class minushalf.data.periodic_table.**PeriodicTable**(*value*)

Bases: Enum

Enum type for the elements of the periodic table.

Ac = 'Ac'

Ag = 'Ag'

Al = 'Al'

Am = 'Am'

Ar = 'Ar'

As = 'As'

At = 'At'

Au = 'Au'

B = 'B'

Ba = 'Ba'

Be = 'Be'

Bh = 'Bh'

Bi = 'Bi'

Bk = 'Bk'

Br = 'Br'

C = 'C'

Ca = 'Ca'

Cd = 'Cd'

Ce = 'Ce'

Cf = 'Cf'

Cl = 'Cl'

Cm = 'Cm'

```
Cn = 'Cn'

Co = 'Co'

Cr = 'Cr'

Cs = 'Cs'

Cu = 'Cu'

Db = 'Db'

Ds = 'Ds'

Dy = 'Dy'

Er = 'Er'

Es = 'Es'

Eu = 'Eu'

F = 'F'

Fe = 'Fe'

Fl = 'Fl'

Fm = 'Fm'

Fr = 'Fr'

Ga = 'Ga'

Gd = 'Gd'

Ge = 'Ge'

H = 'H'

He = 'He'

Hf = 'Hf'

Hg = 'Hg'

Ho = 'Ho'

Hs = 'Hs'

I = 'I'

In = 'In'

Ir = 'Ir'

K = 'K'

Kr = 'Kr'

La = 'La'
```

```
Li = 'Li'

Lr = 'Lr'

Lu = 'Lu'

Lv = 'Lv'

Mc = 'Mc'

Md = 'Md'

Mg = 'Mg'

Mn = 'Mn'

Mo = 'Mo'

Mt = 'Mt'

N = 'N'

Na = 'Na'

Nb = 'Nb'

Nd = 'Nd'

Ne = 'Ne'

Nh = 'Nh'

Ni = 'Ni'

No = 'No'

Np = 'Np'

O = 'O'

Og = 'Og'

Os = 'Os'

P = 'P'

Pa = 'Pa'

Pb = 'Pb'

Pd = 'Pd'

Pm = 'Pm'

Po = 'Po'

Pr = 'Pr'

Pt = 'Pt'

Pu = 'Pu'
```

```
Ra = 'Ra'
Rb = 'Rb'
Re = 'Re'
Rf = 'Rf'
Rg = 'Rg'
Rh = 'Rh'
Rn = 'Rn'
Ru = 'Ru'
S = 'S'
Sb = 'Sb'
Sc = 'Sc'
Se = 'Se'
Sg = 'Sg'
Si = 'Si'
Sm = 'Sm'
Sn = 'Sn'
Sr = 'Sr'
Ta = 'Ta'
Tb = 'Tb'
Tc = 'Tc'
Te = 'Te'
Th = 'Th'
Ti = 'Ti'
Tl = 'Tl'
Tm = 'Tm'
Ts = 'Ts'
U = 'U'
V = 'V'
W = 'W'
Xe = 'Xe'
Y = 'Y'
```

Yb = 'Yb'

Zn = 'Zn'

Zr = 'Zr'

minushalf.data.softwares module

List softwares supported by the CLI

class minushalf.data.softwares.**Softwares**(*value*)

Bases: Enum

Enum type for the softwares supported by the program

static get_default()

Returns the default value for this parameter

static to_list()

Generate list of available softwares

vasp = 'VASP'

Module contents

Init file for data module

4.1.4 minushalf.interfaces package

Submodules

minushalf.interfaces.band_projection_file module

Interfaces for files thar contain band projections

class minushalf.interfaces.band_projection_file.**BandProjectionFile**

Bases: ABC

Class to handle files containing informations about projections of the atoms in bands.

abstract get_band_projection(*kpoint: int*, *band_number: int*) → dict

Abstract method for return the contribution of each atom orbital in a specific band of an specific kpoint.

minushalf.interfaces.correction module

Correction interface

class minushalf.interfaces.correction.**Correction**

Bases: ABC

Interface to correction algorithms

abstract execute()

Execute the correction

minushalf.interfaces.potential_file module

Interface for classes that handles with fourier transforms of the potential

class minushalf.interfaces.potential_file.**PotentialFile**

Bases: ABC

Interface for defining common methods between classes that handles with potential files of different softwares

abstract get_maximum_module_wave_vector() → float

Abstract methods returns the maximum modulus of the wave vector in reciprocal space

abstract get_name() → float

Abstract methods returns the name of the potential file

abstract get_potential_fourier_transform() → list

Abstract methods returns the fourier transform of the potential

abstract to_file(*filename: str*) → None

Abstract methods that output a potential file with a specific filename

abstract to_stringlist() → list

Abstract methods that returns a list containing the lines of the potential file

minushalf.interfaces.runner module

Runner abstract class

class minushalf.interfaces.runner.**Runner**

Bases: ABC

Run softwares that realizes ab initio calculations

abstract run(*cwd: str*)

Create command and run the subprocess for it

minushalf.interfaces.software_abstract_factory module

Software abstract Factory

class minushalf.interfaces.software_abstract_factory.**SoftwaresAbstractFactory**

Bases: ABC

Abstract Factory for create instances for each supported software.

abstract get_atoms_map(*filename: str*, *base_path: Optional[str] = None*) → dict

Abstract method for returns a map of the atomic symbol to its index.

abstract get_band_projection_class(*filename: str*, *base_path: Optional[str] = None*) → *BandProjectionFile*

Abstract method for returns the class that handles with the projections of atoms orbitals in the bands.

abstract get_eigenvalues(*filename: str*, *base_path: Optional[str] = None*) → dict

Abstract method for returns eigenvalues for each band and each kpoint

abstract get_fermi_energy(*filename: str*, *base_path: Optional[str] = None*) → float

Abstract method for returna energy of the fermi level.

abstract get_nearest_neighbor_distance(*ion_index: str*, *filename: str*, *base_path: Optional[str] = None*) → float

Abstract method for returns the nearest neighbor distance for an ion in the solid.

abstract get_number_of_bands(*filename: str*, *base_path: Optional[str] = None*) → int

Abstract method for returns the number of bands used in the calculation

abstract get_number_of_equal_neighbors(*atoms_map: dict*, *symbol: str*, *filename: str = 'OUTCAR'*, *base_path: Optional[str] = None*) → float

Given an map that links atoms symbols with it's index this function returns the number of neighbors of the atom with equal symbol but different indexes.

Args:

atoms_map (dict): Map the atoms index to their symbol. symbom (str): The symbol of the target atom.

Returns:

number_equal_neighbors (int): Returns the number of neighbors with same symbol but different indexes.

abstract get_number_of_kpoints(*filename: str*, *base_path: Optional[str] = None*) → int

Abstract method for returns the number of kpoints used in the calculation

abstract get_potential_class(*filename: str*, *base_path: Optional[str] = None*) → *PotentialFile*

Abstract method for returns the potential class

abstract get_runner(*command: List[str]*)

Get a class that run the software for ab initio calculations

Module contents

Init file for interfaces

4.1.5 minushalf.softwares package

Subpackages

minushalf.softwares.vasp package

Submodules

minushalf.softwares.vasp.eigenval module

Reads Eigenval file, an output of VASP software

class minushalf.softwares.vasp.eigenval.**Eigenvalues**(*filename: str*)

Bases: object

Reads eigenvalues and store useful informations

minushalf.softwares.vasp.potcar module

Reads and analyze POTCAR file

class minushalf.softwares.vasp.potcar.**Potcar**(*filename: str = 'POTCAR'*)

Bases: *PotentialFile*

Parse the POTCAR file, a vasp input file. It store the fourier coefficients and the restant lines of the file

get_maximum_module_wave_vector() → float

Returns
maximum modulus of the wave vector in reciprocal space

Return type
k_max (float)

get_name() → str

Returns potential file name

get_potential_fourier_transform() → list

Returns
List of fourier transform of the potential

Return type
potential(list)

to_file(*filename: str*) → None

Write POTCAR file

Args:
filename (str): Name of the file

to_stringlist() → list

Returns
List of the POTCAR lines

Return type
potcar_lines (list)

minushalf.softwares.vasp.procar module

Reads procar file, an output of VASP software

class minushalf.softwares.vasp.procar.**Procar**(*filename: str*)

Bases: *BandProjectionFile*

Reads procar and store useful informations

get_band_projection(*kpoint: int*, *band_number: int*)

Get the band projection for an specific kpoint and number of band

Args:
kpoint (int): Number of kpoints band_number (int): Number of the band

minushalf.softwares.vasp.runner module

Implementation for vasp runner

class minushalf.softwares.vasp.runner.**VaspRunner**(*command: List[str]*)

Bases: *Runner*

Output terminal command that aims to run vasp

run(*cwd: str = '.'*)

Create a subproccess to run vasp

minushalf.softwares.vasp.vasprun module

Reads vasprun.xml file, an output of VASP software

class minushalf.softwares.vasp.vasprun.**Vasprun**(*filename: str*)

Bases: object

Reads vasprun.xml and store useful informations

Module contents

Vasp module

Submodules

minushalf.softwares.vasp_factory module

Factory to generate same modules for different softwares

Supports the following softwares: - VASP

class minushalf.softwares.vasp_factory.**Vasp**

Bases: *SoftwaresAbstractFactory*

Concrete Factory for create instances for each supported software.

get_atoms_map(*filename: str = 'vasprun.xml'*, *base_path: Optional[str] = None*)

Abstract method for returns a map of the atomic symbol to its index.

get_band_projection_class(*filename: str = 'PROCAR'*, *base_path: Optional[str] = None*)

Abstract method for returns the class that handles with the projections of atoms orbitals in the bands.

get_eigenvalues(*filename: str = 'EIGENVAL'*, *base_path: Optional[str] = None*)

Abstract method for returns eigenvalues for each band and each kpoint

get_fermi_energy(*filename: str = 'vasprun.xml'*, *base_path: Optional[str] = None*)

Abstract method for returna energy of the fermi level.

get_nearest_neighbor_distance(**args*, ***kwargs*)

Abstract method for returns the nearest neighbor distance for an ion in the solid.

get_number_of_bands(*filename: str = 'PROCAR'*, *base_path: Optional[str] = None*)

Abstract method for returns the number of bands used in the calculation

get_number_of_equal_neighbors(**args*, ***kwargs*)

Given an map that links atoms symbols with it's index this function returns the number of neighbors of the atom with equal symbol but different indexes.

Args:

atoms_map (dict): Map the atoms index to their symbol. symbom (str): The symbol of the target atom.

Returns:

number_equal_neighbors (int): Returns the number of neighbors with same symbol but different indexes.

get_number_of_kpoints(*filename: str = 'PROCAR'*, *base_path: Optional[str] = None*)

Abstract method for returns the number of kpoints used in the calculation

get_potential_class(*filename: str = 'POTCAR'*, *base_path: Optional[str] = None*)

Abstract method for returns the potential class

get_runner(*command: List[str]*)

Return the class that runs VASP

Module contents

Init file for software factory

4.1.6 minushalf.utils package

Submodules

minushalf.utils.atomic_potential module

Correct crystal potential to fractional occupations

class minushalf.utils.atomic_potential.**AtomicPotential**(*vtotal: Vtotal*, *vtotal_occupied: Vtotal*, *potential_file: PotentialFile*)

Bases: object

Correct atomic potential fourier tranform for fractional occupations in valence or conduction bands

correct_file(*potential: list*, *cut: float*, *amplitude: float*, *is_conduction: bool = False*) → None

Create the potential file corrected

Args:

potential (list): List of corrected potentials fourier transform

cut (float): Cutting parameter to cancel the potential

amplitude (float): Multiplicative factor of the potential function

correct_potential(*cut: float*, *amplitude: float*, *is_conduction: bool = False*) → list

Correct fourier transform of the potential (V(k)) present in POTCAR file.

Args:

cut (float): Cutting parameter to cancel the potential

amplitude (float): Multiplicative factor of the potential function

is_conduction (bool): Indicates whether the potential correction will be made in the valence or in the conduction

Returns:
List of corrected potentials fourier transform

`get_corrected_file_lines`(*potential: list*) → list

Create the potential file corrected

Args:
potential (list): List of corrected potentials fourier transform

Returns:
potential_lines(list): A List of potcar lines

`occupy_potential`(*cut: float*, *amplitude*) → list

Parameters

- **cut** (*float*) – Cutting parameter to cancel the potential
- **amplitude** (*float*) – Multiplicative factor of the potential function

Returns
A list that contains the potentials of fractional electron occupation at the exact level to be corrected.

minushalf.utils.band_structure module

Band structure informations

`class minushalf.utils.band_structure.BandStructure`(*eigenvalues: dict*, *fermi_energy: float*, *atoms_map: dict*, *num_bands: int*, *band_projection:* BandProjectionFile)

Bases: `object`

Extact band structure insights from VASP classes

`band_gap`() → dict

Find VBM and CBM, then returns band gap :returns: VBM index and its eigenvalue, CBM index and its eigenvalue and band gap

`band_projection`(*kpoint: int*, *band: int) -> defaultdict(<class 'list'>, {}*)

Find the projection of each atom for a specific band.

Args:
kpoint (int): Number of kpoints band_number (int): Number of the band

Returns:
band_projection (defaultdict(list)): Contains the projection of each orbital of each atom in the respective band

`cbm_index`() → tuple

Find the kpoint and the band for cbm

Returns:
vbm_index (tuple): Contains the kpoint number and the band number of th vbm

cbm_projection() -> *defaultdict(<class 'list'>, {})*

Find the projection of each atom for valence band minimum.

Returns:
vbm_projection (defaultdict(list)): Contains the projection of each orbital of each atom in the respective band

static create(*software_module:* SoftwaresAbstractFactory, *base_path: str = '.'*)

Create band structure class from ab inition results

Args:

software_module (SoftwaresAbstractFactory): Holds the results of first principles output calculations base_path (str): Path to first principles output files

Returns:

band_strucure (BandStructure): Class with band structure informations

is_metal(*tolerance: float = 0.0001*) → bool

Check if the band structure indicates a metal by looking if the fermi level crosses a band.

Returns:
True if a metal, False if not

vbm_index() → tuple

Find the kpoint and the band for vbm

Returns:
vbm_index (tuple): Contains the kpoint number and the band number of th vbm

vbm_projection() -> *defaultdict(<class 'list'>, {})*

Find the projection of each atom for valence band maximum.

Returns:
vbm_projection (defaultdict(list)): Contains the projection of each orbital of each atom in the respective band

minushalf.utils.check_file_exists module

Function to check if a file exists

minushalf.utils.check_file_exists.**check_eigenval_exists**(*func*)

Function decrator to check if a file exists

minushalf.utils.check_file_exists.**check_outcar_exists**(*func*)

Function decrator to check if a file exists

minushalf.utils.check_file_exists.**check_potcar_exists**(*func*)

Function decrator to check if a file exists

minushalf.utils.check_file_exists.**check_procar_exists**(*func*)

Function decrator to check if a file exists

minushalf.utils.check_file_exists.**check_vasprun_exists**(*func*)

Function decrator to check if a file exists

minushalf.utils.cli_messages module

Default cli messages

minushalf.utils.cli_messages.**end_message**() → None

Print end message

minushalf.utils.cli_messages.**welcome_message**(*text: str*) → None

Print welcome message

minushalf.utils.correct_potential_fourier_transform module

Fourier Transform

minushalf.utils.correct_potential_fourier_transform.**correct_potential_fourier_transform**(*coefficient: array*, *k: array*, *rays: array*, *occupation_potential: array*, *cut: float*) → array

The pseudopotential is given in terms of the radial distance, and is only defined for r >= 0, as expected. Since it is only evaluated inside an integral from 0 to infinity, it does not matter what values it assumes for r < 0. A natural choice is to define the function to be zero for negative values, but a more convenient choice is to choose v(-r)=-v(r) and n(-r)=-n(r), since purely real and odd functions have purely imaginary Fourier transforms. Let v' and n' be the odd extensions of the potential and the number density, respectively.

$$E_v = \int_0^\infty v(r)n(r)dr = \int_0^\infty v'(r)n'(r)dr = \frac{1}{2}\cdot\int_{-\infty}^\infty v'(r)n'(r)dr = -\frac{1}{2}\cdot\int_{-\infty}^\infty V(k)N(k)dk$$

On the third equalitty, we used the fact that the product of two odd functions is even, and in the last step we have applied Parseval's theorem, considering that the Fourier transforms are purely imaginary. Even though the function may not pass through the origin, we can still make an odd extension, by making it discontinuous.

The data stored on POTCAR corresponds to the Fourier transform of the odd extension of v. It can be approximated by the summation on the right, where the prefactors were ommited.

$$V(k) = i\cdot\sqrt{\frac{2}{\pi}}\cdot\int_0^\infty v(r)sin(b\cdot k\cdot r)dr \Rightarrow V(k) \sim \sum_{i=1}^{N_r}\frac{(v[i]\cdot sin(b\cdot k\cdot r[i]) + v[i-1]\cdot sin(b\cdot k\cdot r[i-1]))}{2\cdot(r[i]-r[i-1])}$$

Computes the opposite of the imaginary part of the j-th fourier transform coefficient through numerical integrationIndex zero stands for the r=DeltaR, and the function is assumed to be zero at the origin. Thus, the first trapezium of the numerical integration is degenerated to a triangulum, and its area must be calculated as so.

Args:

coefficient (np.array): Fourier transform of the potential for the atom in its ground state

k (np.array): The wave vector in reciprocal space

rays(np.array): List of rays on which pseudopotential calculations were made

occupation_potential (np.arraygit): Potential of fractional electron occupation at the exact level to be corrected

cut(float): Cutting parameter to cancel the potential

Returns:

Fourier transform of the potential for the state with fractional occupation of the crystal

minushalf.utils.cut_initial_guess module

Gives cut initial guess

class minushalf.utils.cut_initial_guess.**CutInitialGuess**

Bases: object

Estimate cut inital guess from the nearest neighbor distance.

guess(*distance: float*, *method: str*) → float

Given the nearest neighbor distance and the method, it returns the initial guess.

Args:

method (str): method of gessing distance (float): nearest neighbor distance

Returns:

cut_guess (float): An initial guess to cut.

minushalf.utils.drop_comments module

Function to check if the line a commentarie or not

minushalf.utils.drop_comments.**drop_comments**(*lines: list*) → list

Function to remove comments from lines in a file

Args:

lines(list): list of file lines

Returns:

lines_without_comments (list): lines of the file without comments

minushalf.utils.fractionary_correction_indexes module

Get atoms to be corrected for simple valence correction and simple conduction correction

minushalf.utils.fractionary_correction_indexes.**get_fractionary_correction_indexes**(*band_projection: DataFrame*, *treshold: int = 5*) → dict

Get dataframe index of the orbitals which contributes more than 5 percent to (VBM|CBM)

Returns:

correction_indexes (dict):A dict wherw the keys are the atoms
symbols and the value is a list with the orbitals type to be corrected. Ex: { 'Ga': ['p','s'], 'N' : ['d','f'], }

minushalf.utils.get_correction_params module

Extract the parameters for the correction

minushalf.utils.get_correction_params.**get_conduction_correction_params**(*minushalf_yaml: MinushalfYaml*, *software_factory:* SoftwaresAbstractFactory, ***kwargs*)

Returns the parameters for the conduction correction

minushalf.utils.get_correction_params.**get_valence_correction_params**(*minushalf_yaml: MinushalfYaml*, *software_factory:* SoftwaresAbstractFactory, ***kwargs*)

Returns the parameters for the valence correction

minushalf.utils.negative_band_gap module

Returns band-gap with the sinal changed, so one can use minimization algorithms to find the cut value that results in the maximum band_gap

minushalf.utils.negative_band_gap.**find_negative_band_gap**(*cuts: list*, **args: tuple*) → float

Run vasp and return the gap value multiplied by -1

Parameters

- **cuts** (*float*) – List of cuts
- **args** (*tuple*) – tuple containning a dictionary with the fields base_path (str): Path to mkpotcar{symbol}_{orbital} symbol (str): Atom symbol default_potential_filename (str): The default potential filename for each software potfiles_folder (str): Folder containing unmodified potfiles amplitude (float): scale factor to trimming function runner (Runner): runner for the software software_factory(SoftwaresAbstractFactory): Factory for each software atom_potential(AtomicPotential): Holds fourier transforms of the potential software_files (list): Aditional files besides potential file to make ab initio calculations

Returns
band gap multiplied for -1

Return type
negative_gap (float)

minushalf.utils.parse_cut module

Parse cut string used in correct potential file

minushalf.utils.parse_cut.**parse_cut**(*cut: str*) → list

Parse cut in a list of numbers.

Parameters
cut (*str*) – Cut energy to be used in the program, it can be passed in two ways:

unique value : float or integer range: begin(float|integer):pass(float|integer):end(float|integer)

Returns
Permited values of cut.

Return type
cut_numbers (list)

minushalf.utils.parse_valence_orbital_line module

Parse valence orbital line in INP file.

minushalf.utils.parse_valence_orbital_line.**parse_valence_orbitals**(*line: str*) → dict

Parse valence orbital line in principal quantum number, angular momentum quantum number and eletronic occupation

Args:
line (str): line of imp file that represents a valence orbital

Returns:
A dictionary with fields n, l and eletronic occupation

minushalf.utils.projection_to_df module

Transform informations about band_project generated by cbm_character, vbm_character or band_character in a normalized datafame Grouped by orbital types

minushalf.utils.projection_to_df.**projection_to_df**(*projection: defaultdict(<class 'list'>, {})*) → DataFrame

Transform received dictionaries into information with a higher degree of readability.

Args:
projection (defaultdict(list)): A dictionary containing the projections of each atom per orbital

Returns:
prolection_df (pd.DataFrame): A dataframe containing the projections per orbital type and normalized between 0 - 100

minushalf.utils.simple_correction_indexes module

Get atoms to be corrected for simple valence correction and simple condunction correction

minushalf.utils.simple_correction_indexes.**get_simple_correction_indexes**(*band_projection: DataFrame*) → dict

Get dataframe index of the orbital which contributes more to (VBM|CBM)

Returns:

correction_indexes (dict):A dict wherw the keys are the atoms
symbols and the value is a list with the orbitals type to be corrected. Ex: { 'Ga': ['p','s'], 'N' : ['d','f'], }

minushalf.utils.trimming_function module

Trimming function

minushalf.utils.trimming_function.**trimming_function**(*radius: array*, *ion_potential: array*, *atom_potential: array*, *cut: float*, *amplitude: float*) → array

Function that generate the potential for fractional occupation. The potential is cuted by a a function theta(r) to avoid divergence in calculations. The function of potential is defined as follows:

$$V_{1/2} = (V_{atom} - V_{ion}) \cdot \theta(r)$$

where theta is:

$$\theta(r) = A \cdot (1 - (\frac{r}{CUT})^n)^3, r \leq CUT$$

$$\theta(r) = 0, r > CUT$$

—

Args:

cut (float): cutting parameter to cancel the potential

amplitude (float): multiplicative factor of the potential function

radius (np.array): rays in which the potential was calculated

ion_potential (np.array): Atom pseudopotential with fractional occupation

atom_potential (np.array): Atom pseudopotential with all electrons

Returns:

potential of fractional electron occupation at the exact level to be corrected

Module contents

Init file for utils module

4.1.7 minushalf.io package

Submodules

minushalf.io.atomic_program module

Class for atomic program input parameters in minushalf.yaml

class minushalf.io.atomic_program.**AtomicProgram**(*exchange_correlation_code: str = 'pb'*, *calculation_code: str = 'ae'*, *max_iterations: int = 100*)

Bases: `MinushalfYamlTags`

Set parameters and their default values

to_dict()

Return dictionary with the class variables

to_list()

return list with the class variables

minushalf.io.correction module

Class for correction input parameters in minushalf.yaml

class minushalf.io.correction.**Correction**(*correction_code: str = 'v'*, *potfiles_folder: str = 'minushalf_potfiles'*, *amplitude: float = 1.0*, *valence_cut_guess: Optional[list] = None*, *conduction_cut_guess: Optional[list] = None*, *tolerance: float = 0.01*, *fractional_valence_treshold: float = 10*, *fractional_conduction_treshold: float = 9*, *cbm_characters: Optional[list] = None*, *vbm_characters: Optional[list] = None*, *overwrite_vbm: Optional[list] = None*, *overwrite_cbm: Optional[list] = None*, *inplace: bool = False*, *divide_character: Optional[list] = None*)

Bases: `MinushalfYamlTags`

Set parameters and their default values

property cbm_characters: list

Returns: Artificial character in cbm

property conduction_cut_guess: list

Returns: CUT guess for nelder mead algorithm

property correction_code: dict

Returns: Code for DFT -1/2 correction

property divide_character: list

Returns: Factor that divides the correction between atoms

property overwrite_cbm: dict

Returns: Tag to overwrite vbm character

property overwrite_vbm: dict

Returns: Tag to overwrite vbm character

to_dict()

Return dictionary with the class variables

to_list()

return list with the class variables

property valence_cut_guess: list

Returns: CUT guess for nelder mead algorithm

property vbm_characters: list

Returns: Artificial character in vbm

minushalf.io.input_file module

Leads with input file (INP.ae) read by atomic program.

class minushalf.io.input_file.**InputFile**(*exchange_correlation_code: str*, *calculation_code: str*, *chemical_symbol: str*, *esoteric_line: str*, *number_valence_orbitals: int*, *number_core_orbitals: int*, *valence_orbitals: list*, *description: str = ''*, *last_lines: Optional[list] = None*)

Bases: object

Parses input file.

property calculation_code: str

Returns: Calculation code for inp file (ae)

property chemical_symbol: str

Returns: Chemical symbol of the element (H, He, Li...)

electron_occupation(*electron_fraction: float*, *secondary_quantum_number: int*) → None

Corrects the input file of the atomic program, decreasing a fraction of the electron in a layer specified by the secondary quantum number

Args:

electron_fraction (float): Fraction of the electron that will be decreased in the INP file. Can vary between 0 and 0.5

secondary_quantum_number (int): Specifies the layer on which the occupation is to be made.

property exchange_correlation_code: str

Returns: Functional of exchange and correlation (ca, wi, hl, gl,bh, pb, rp, rv, bl

static from_file(*filename: str = './INP'*) → any

Parse INP.ae file.

Args:

filename: name of the INP file.

Returns:

input_file: instance of InputFile class.

static minimum_setup(*chemical_symbol: str*, *exchange_correlation_code: str*, *maximum_iterations: int = 100*, *calculation_code: str = 'ae'*) → any

Create INP file with minimum setup.

Args: chemical_symbol (str): Symbol of the chemical element (H, He, Li...).

exchange_correlation_code (str): functional of exchange and correlation (ca, wi, hl, gl, bh, pb, rp , rv, bl)

maximum_iterations (int): Maximum number of iterations for atomic program. The default is 100

Returns:
input_file: instance of InputFile class.

`to_file`(*filename: str = './INP'*) → None

Write INP file

Args:
filename (str): name of the output file

`to_stringlist`() → list

Returns
List with the lines of the INP file.

minushalf.io.make_minushalf_results module

function make minushalf_results.dat

`minushalf.io.make_minushalf_results.`**`make_minushalf_results`**(*gap: float*, *valence_cuts: Optional[dict] = None*, *conduction_cuts: Optional[dict] = None*, *name: str = 'minushalf_results.dat'*) → None

Make output file for execute command, minushalf_results.dat.

Args:
gap (float): final gap in the correction method

valence_cuts (dict):dictionary inform the atom symbol, orbital
and cut for valence correction in the following format: {(symbol,orbital):cut}

conduction_cuts (dict):dictionary inform the atom symbol, orbital
and cut for condsuction correction in the following format. {(symbol,orbital):cut}

name (str): name of the file

minushalf.io.minushalf_yaml module

Parser for minushalf.yaml

class `minushalf.io.minushalf_yaml.`**`MinushalfYaml`**(*software_configurations: MinushalfYamlTags*, *atomic_program: MinushalfYamlTags*, *correction: MinushalfYamlTags*)

Bases: `MinushalfYaml`

Class that parses the input for the execute command

static `from_file`(*filename: str = 'minushalf.yaml'*)

Receives a file and catch all the parameters presents in the documentation

get_amplitude() → float
Returns the amplitude

get_atomic_program_params() → dict
Get dictionary of atomic program parameters

get_calculation_code() → str
Returns the calculation code

get_cbm_characters() → list
Returns the parameter vbm_characters

get_command() → list
Returns the command that runs first principles calculations

get_conduction_cut_initial_guess() → str
Returns the conduction cut initial guess

get_correction_code() → list
Returns the code used to identify the correction

get_correction_params() → dict
Get dictionary of correction parameters

get_divide_character() → list
Returns the divide characters

get_exchange_corr_code() → str
Returns exchange correlation code

get_inplace() → bool
Returns the inplace

get_max_iterations() → int
Returns max iterations

get_overwrite_cbm() → list
Returns the parameter that overwrites cbm

get_overwrite_vbm() → list
Returns the parameter that overwrites vbm

get_potential_folder() → str
Returns the potential folder name

get_software_configurations_params() → dict
Get dictionary of software configurations parameters

get_software_name() → str
Returns the name of the software that runs first principles calculations

get_tolerance() → float
Returns the tolerance

get_valence_cut_initial_guess() → str
Returns the valence cut initial guess

get_vbm_characters() → list
Returns the parameter that cbm_characters

minushalf.io.software_configurations module

Class for atomic program software parameters in minushalf.yaml

class minushalf.io.software_configurations.**SoftwareConfigurations**(*command: Optional[list] = None*, *software_name: str = 'VASP'*)

Bases: MinushalfYamlTags

Set parameters and their default values

property command: dict

Returns: Command to perform first-principles calculations

property software_name: str

Returns: Name of the software used for ab initio calculations (VASP,...)

to_dict()

Return dictionary with the class variables

to_list()

return list with the class variables

minushalf.io.vtotal module

Analyze VTOTAL

class minushalf.io.vtotal.**Vtotal**(*radius: array*, *down_potential: array*)

Bases: object

Output for ATOM that contains the pseudopotential generated by an atom

static from_file(*filename: str = './VTOTAL.ae'*) → any

Parse VTOTAL and extract the following informations

static read_down_potential(*filename: str*) → array

Extracts the potentials related to the state of spin Down calculated for the main elements

Args:
filename (str): Name of the VTOTAL file

static read_radius(*filename: str*) → array

Extracts from the file information regarding the rays for which the potential calculations will be made made.

Args:
filename (str): Name of the VTOTAL file

Module contents

Python files to handle input and outputs

4.2 Submodules

4.3 minushalf.minushalf module

Definition of the minushalf CLI

4.4 Module contents

Init file for minushalf

CHAPTER

FIVE

SUPPORT AND FINANCING

This project was developed at Instituto Tecnológico de Aeronáutica (ITA) , in São José dos campos, with the collaboration of other researchers from the materials group semiconductors and nanotechnology (GMSN). The project was funded by the CNPq Institutional Scientific Initiation Scholarship Program (PIBIC) and by CAPES.

PYTHON MODULE INDEX

m

INDEX

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z