Documentation

Parametrized Tight-Binding Program: JuTiBi

von

Timo Schena

April 2011

Contents

1 Introduction

I want to start with a motivation and an answer to the question, why is computational physics important?

The computational performance will continue to increase obeying Moore's law until at least to the year 2020 and also the complexity of the systems able to investigate with the help of state-of-the-art computers will increase. Therefore the calculation of "real" systems becomes more and more important to gain a better understanding of the condensed matter. A lot of methods have been developed to investigate the electronic, magnetic, thermal, optical properties and more of the condensed matter. Beside the scientifically important issue of gaining a better understanding these calculations could also have the predictive power to lead to technologically advanced materials. Therefore no one should underestimate the role of computational physics in the physics.

In the following I want to focus on the calculation of the electronic and magnetic ground state properties. One of the most prominent ways to determine the electronic and magnetic ground state properties is the density functional theory (DFT) [1, 2]. Its rather accurate description has lead to many success stories and it has the big advantage to be parameter-free (i. e. *ab-initio*). This means that no external, empirical parameters have to be used for calculation and therefore DFT can be applied to general systems. However, there are systems which can not be explained within DFT as for example strongly correlated systems. But if we limit the systems to the cases where DFT yields a nice description one could ask, where is the advantage to use non-DFT based methods instead?

This is a legitimate question, because DFT-based methods yield the most accurate and reliable results of all methods constructed on an one-electron Schrödinger equation as far as I know. But keep in mind that this accuracy has its price. The required computational time for treating large and complex systems represents a challenge even on tomorrow's computers. In addition the more accurate results the more complex the DFT-based method, which makes it more difficult to obtain a physical interpretation of the results. Therefore it would be helpful to have a fast and simple method, which allows to investigate tendencies and the qualitative behaviour of the electronic and magnetic structure to gain a better understanding. The JuTiBi code explained in this documentation is suited to these demands. The JuTiBi code is based on a parametrized tight-binding scheme, which allows fast calculations combined with a large freedom of changing physically insightful parameters to perform "numerical experiments".

First I want to give you a short overview over the tight-binding Hamiltonian used in the JuTiBi code:

$$\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_{\mathrm{mag}} + \mathcal{H}_{\mathrm{LCN}} + \mathcal{H}_{\mathrm{SOC}} \,. \tag{1.1}$$

$\mathcal{H}_0$ contains the on-site energies and the hopping elements, which parametrization is based on the NRL-TB method [3, 4, 5, 6] and therefore ensures parameter sets with a high degree of transferability. This means that a parameter set suited to a special structure can be used also for a qualitatively satisfying description of a similar structure. For describing magnetism a Stoner model [7, 8] is included via $\mathcal{H}_{\mathrm{mag}}$, which can be straightforwardly modified for non-collinear magnetic systems. Tuning the Stoner parameters allows to change magnetic moments and the polarization and therefore is an example for the aforementioned "numerical experiments". The implementation of the Stoner model allows a self-consistent calculation of the charges. In this self-consistent cycle $\mathcal{H}_{\mathrm{LCN}}$ assures local charge neutrality. The implementation of the generalized Bloch theorem [9, 10, 11] makes sure that spin spiral calculations for each spin-spiral vector $\boldsymbol{q}$ can be performed on the same unit cell. Even under consideration of spin-orbit coupling (SOC) via the Hamiltonian $\mathcal{H}_{\mathrm{SOC}}$, the advantage of the generalized Bloch theorem can be utilized by treating SOC in 1st order perturbation theory [12, 13]. This tight-binding model has been successfully used for calculations of magnetic properties in Fe-systems [14, 15, 16], Pt-systems [17, 16] and Fe/Pt-systems [16].

So, what can be done with the JuTiBi code?

The JuTiBi code is suited to treat complex magnetic structures of moderate system-size, in particular spin-spirals. Therefore the code could be very useful to calculate the data needed to obtain the Heisenberg exchange-coupling parameters, the Dzyaloshinskii-Moriya constant and the magneto-crystalline anisotropy. The code can treat bulk-, surface-, chain- and cluster-systems and for almost all transition metals high-quality NRL-TB parameter sets do exist [4].

Due to the parametrization a lot of physical properties can be tuned to get a feeling for the most important physics in a these systems. For example the spin-polarization can be controlled via the Stoner parameters, the SOC-strength via the SOC-parameters and in principle there is even the possibility to control each hopping element of the system. Beside the investigation of magnetic structures the code can be also used to investigate pure SOC-related phenomena as for example the Rashba-splitting. Due to the simplicity of the SOC-description in the tight-binding code, one could again get a feeling for the main mechanisms of this phenomena.

For users who are interested in calculating transport properties the JuTiBi code could be also of interest. In the recent version no transport is included, however it should be not too difficult to exploit the tight-binding scheme for transport problems.

Every method has also disadvantages and a disadvantage of the here used parametrized tight-binding method is the quantitative description. The parameter sets in the NRL-TB parametrization are obtained from fits to *ab-initio* band structures of the corresponding bulk geometries. Therefore the description of a surface- or chain-system does not yield reliable quantitative results. The same problem regards the treatment of binary systems within the NRL-TB scheme. However, the qualitative description is in a rather nice agreement to *ab-initio* results [16]. Therefore let us conclude and determine if the code is useful for your purposes:

You should use the code if you want to perform fast calculations and you do not care about the exact numerical values, but rather the tendencies. If you want to analyse a

system by tuning particular properties as the polarization, the SOC-strength etc. to gain a better understanding, therefore to say it short - to conduct "numerical experiments" - this code could be also very useful. If you have the results of an *ab-initio* calculation, but it is quite too difficult to extract the important physical quantities for the behaviour the JuTiBi code could be also helpful.

Now when do you should not work with the JuTiBi code? If you want to have a numerical precise prediction of a physical quantity as for example the magneto-crystalline anisotropy of a system you should use other methods.

Additionally I want to remark that in the recent version of the JuTiBi code you should not use it to perform calculations for very large system (i. e. several hundert atoms). In principle one can (easily) extend it to treat such systems, but in this version the code is especially designed to investigate the magnetic interactions in systems of a size, which should not exceed 100 atoms. In particular users interested in spin-spiral calculations should take a deeper look into my code.

Finally I want to give a short outline, what you have to expect in this documentation.

First a detailed description of the theory behind the tight-binding scheme is presented in chapter 2. Then the `inputcard` of the JuTiBi code is explained in thematic sections in the chapter 3. In the next chapter 4 the output of the JuTiBi code is described in detail. First the output on the screen is presented in section 4.1, then all external files are explained in section 4.2. In the tutorial in chapter 5 the user can exercise with the JuTiBi code using well chosen examples in a reasonable order, which assures a well learning curve. I would recommend all possible users to work through this chapter!

The next chapter and the following appendices are suited to users, who want to modify the code or want to understand the code structure in detail. First a detailed description of the most important subroutines of the JuTiBi code together with an overview of the code structure is presented in chapter 6. The appendices contain some detailed calculations for the the theory (appendix A), the parameter sets for Fe and Pt (appendix B), an example of the full `inputcard` to get a feeling of its structure (appendix C; to read details better use magnifying glasses ;)), a full list of the appearing keywords in the `inputcard` with short description (appendix D), a full list of all important variables appearing in the JuTiBi code with short description (appendix E) and finally a list of all data sets contained in the JuTiBi package to check completeness.

I wish the users a lot of fun using the JuTiBi code and if you have some constructive critic or questions concerning the JuTiBi code please feel free to mail to t.schena@fz-juelich.de.

2 Theory

2.1 Motivation

The Holy Grail of condensed matter physics is the solution of the eigenvalue problem of the following Hamiltonian:

$$\mathcal{H} = T_e + T_c + V_{e-c} + V_{e-e} + V_{c-c}\,, \tag{2.1}$$

with $T_e = -\sum_i \frac{1}{2}\nabla_i^2$ the kinetic energy of the electrons, $T_c = -\sum_\alpha \frac{1}{2M_\alpha}\nabla_\alpha^2$ the kinetic energy of the nuclei, $V_{e-c} = -\sum_\alpha \sum_i \frac{Z_\alpha}{|\boldsymbol{r_i}-\boldsymbol{R_\alpha}|}$ the Coulomb interaction between the electrons and the nuclei, $V_{e-e} = \frac{1}{2}\sum_i \sum_{j\neq i} \frac{1}{|\boldsymbol{r_i}-\boldsymbol{r_j}|}$ the Coulomb interaction between the electrons and $V_{c-c} = \frac{1}{2}\sum_\alpha \sum_{\beta\neq\alpha} \frac{Z_\alpha \cdot Z_\beta}{|\boldsymbol{R_\alpha}-\boldsymbol{R_\beta}|}$ the Coulomb interaction between the nuclei.

It is impossible to solve this problem for many-electron systems without appropriate approximations. First, it is common to use the Born-Oppenheimer approximation, if the electron-phonon coupling does not critically affect the electronic structure. Then the electronic problem is described by the Hamiltonian

$$\mathcal{H}_e = T_e + V_{e-c} + V_{e-e} + V_{c-c}\,, \tag{2.2}$$

where the positions $\boldsymbol{R_\alpha}$ of the nuclei enter only as parameters, which are usually chosen at the equilibrium positions of the lattice. But also this problem is too difficult to solve without approximations due to the electron-electron interaction. Fortunately, there are plenty of more or less successful ways to reduce the N-particle problem to an effective one-(quasi)-particle problem. One of the most prominent ones is the density functional theory (DFT) of Hohenberg, Kohn and Sham [1, 2], which is implemented in the majority of *ab-initio* electronic structure methods. One problem of this highly accurate method is the required computational time, which remains a challenge even on state-of-the-art supercomputers.

To reduce this computational time one can use a parametrized tight-binding method, with the parameters obtained by *ab-initio* calculations. The tight-binding method described in this documentation is able to reproduce the qualitative behaviour of physical systems, including their magnetic properties.

A convenient feature of these method is that, by virtue of the Slater-Koster theory, the Slater-Koster parameters depend only on the type of elements in a material, but not on its geometry. For example there is a possibility to use the tight-binding parameters of bcc-Fe to describe a Fe monolayer. Incorporating a model for the description of magnetism (like a Stoner-like-model , section 2.6) a reasonable description of spin-spiral states in this Fe monolayer can be achieved. At least theoretically the method could

be as accurate as the best *ab-initio* methods, if very accurate parameters for a precise description of the electronic structure are used. In the next sections the theory of the tight-binding method is described in detail.

2.2 Basics

The solution of the one-electron Schrödinger equation in a special finite set of basis functions is searched:

$$\mathcal{H}\left|\Psi\right\rangle = (T+V)\left|\Psi\right\rangle = E\left|\Psi\right\rangle \,, \tag{2.3}$$

where T is the one-electron kinetic energy and V is the effective one-electron potential within a mean-field approximation. As basis functions atomic-orbital-like functions $|\boldsymbol{n}, i, \mu\rangle$ are used, where $\boldsymbol{n}$ denotes the Bravais vector $\boldsymbol{R}_n$, i is the i-th basis atom with its position $\boldsymbol{\tau}_i$ in the unit cell and μ stands for the type of orbital. In our code we use s-, p- and d-orbitals as basis functions, with their well-known angular dependence described by the spherical harmonics $Y_{lm}(\Theta, \phi)$.

Using Ritz's variational principle, one has to solve the following matrix-eigenvalue equation:

$$\underline{\boldsymbol{H}} \cdot \boldsymbol{c} = E\underline{\boldsymbol{S}} \cdot \boldsymbol{c}\,, \tag{2.4}$$

where $\boldsymbol{c}$ is an eigenvector containing the coefficients for the linear expansion of $|\Psi\rangle$:

$$\left|\Psi\right\rangle = \sum_{\boldsymbol{n},i,\mu} c_{\boldsymbol{n}i\mu} \cdot \left|\boldsymbol{n}, i, \mu\right\rangle \,, \tag{2.5}$$

and $\underline{\boldsymbol{H}}$ and $\underline{\boldsymbol{S}}$ are the Hamiltonian and overlap matrix in representation of the atomic orbitals:

$$H_{\boldsymbol{n}i\mu}^{\boldsymbol{m}j\nu} = \left\langle \boldsymbol{n}, i, \mu\right| \mathcal{H} \left|\boldsymbol{m}, j, \nu\right\rangle \,, \tag{2.6}$$

$$\text{and} \qquad S_{\boldsymbol{n}i\mu}^{\boldsymbol{m}j\nu} = \left\langle \boldsymbol{n}, i, \mu|\boldsymbol{m}, j, \nu\right\rangle . \tag{2.7}$$

In this work periodic structures are considered, therefore Bloch's theorem can be used. Thus, we can introduce a new basis of Bloch-waves of the following form:

$$\left|\Phi_{\boldsymbol{k},i,\mu}\right\rangle = \frac{1}{\sqrt{N}} \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \cdot \left|\boldsymbol{n}, i, \mu\right\rangle \,, \tag{2.8}$$

where N ist the number of unit-cells in the chosen super-cell for the periodic boundary conditions. In the representation of the Bloch-waves the Hamiltonian appears as follows:

$$H_{i\mu}^{j\nu}(\boldsymbol{k}) = \left\langle \Phi_{\boldsymbol{k},i,\mu}\right| \mathcal{H} \left|\Phi_{\boldsymbol{k},j,\nu}\right\rangle = \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \cdot H_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu} \,. \tag{2.9}$$

A completely analogous equation defines the matrix elements $S_{i\mu}^{j\nu}(\boldsymbol{k})$ of the overlap matrix:

$$S_{i\mu}^{j\nu}(\boldsymbol{k}) = \left\langle \Phi_{\boldsymbol{k},i,\mu}|\Phi_{\boldsymbol{k},j,\nu}\right\rangle = \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \cdot S_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu} \,. \tag{2.10}$$

The Bloch-waves are orthogonal for different $\boldsymbol{k}$-values in the Brillouin-zone:

$$\langle \Phi_{\boldsymbol{k},i,\mu} | \Phi_{\boldsymbol{k'},j,\nu} \rangle = 0 \quad , \quad \text{if} \quad \boldsymbol{k} \neq \boldsymbol{k'} \, . \tag{2.11}$$

Also the Hamiltonian in representation of the Bloch-waves is block-diagonal with respect to the $\boldsymbol{k}$-space, i.e.:

$$\langle \Phi_{\boldsymbol{k},i,\mu} | \, \mathcal{H} \, | \Phi_{\boldsymbol{k'},j,\nu} \rangle = 0 \quad , \quad \text{if} \quad \boldsymbol{k} \neq \boldsymbol{k'} \, . \tag{2.12}$$

In a parametrized tight-binding scheme the matrix elements $H_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu}$ and $S_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu}$ are described by a parametrization, which will be introduced in the next section.

To conclude, at the heart of the tight-binding theory lies the use of localized basis functions as basis representation, and in this work we use atomic orbitals for this purpose.

2.3 Parametrization of the matrix elements

In this section the parametrization for the matrix elements $H_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu}$ and $S_{\boldsymbol{0}i\mu}^{\boldsymbol{n}j\nu}$ is described. First, it is important to take a look at the different types of matrix elements which appear. For the sake of simplicity only the case of one basis atom is considered.

$$\begin{aligned} H_{\boldsymbol{0}\mu}^{\boldsymbol{n}\nu} &= \int d\boldsymbol{r} \, \phi_\mu^*(\boldsymbol{r}) \cdot (T + V) \cdot \phi_\nu(\boldsymbol{r} - \boldsymbol{R}_n) \\ &= \int d\boldsymbol{r} \, \phi_\mu^*(\boldsymbol{r}) \cdot (T + \sum_m v_{\text{at}}(\boldsymbol{r} - \boldsymbol{R}_m)) \cdot \phi_\nu(\boldsymbol{r} - \boldsymbol{R}_n) \, , \end{aligned} \tag{2.13}$$

where $v_{\text{at}}(\boldsymbol{r} - \boldsymbol{R}_m)$ is the atomic potential of the atom in the m-th unit-cell and $\phi_\mu(\boldsymbol{r}) = \langle \boldsymbol{r} | \boldsymbol{n}, \mu \rangle$. The following types of integrals can be distinguished.

1-center-integrals:

$$\int d\boldsymbol{r} \, \phi_\mu^*(\boldsymbol{r}) \cdot v_{\text{at}}(\boldsymbol{r}) \cdot \phi_\nu(\boldsymbol{r}) \tag{2.14}$$

$\mu = \nu$: typical on-site atomic orbital energies, ε_μ
$\mu \neq \nu$: small contribution if overlap $\langle \phi_\mu | \phi_\nu \rangle \neq 0$

2-center-integrals:

$$\text{(i)} \qquad \int d\boldsymbol{r} \, \phi_\mu^*(\boldsymbol{r}) \cdot \sum_{n \neq 0} v_{\text{at}}(\boldsymbol{r} - \boldsymbol{R}_n) \cdot \phi_\nu(\boldsymbol{r}) \tag{2.15}$$

$\mu = \nu$: contribution to the on-site energies due to existence of other atoms
$\mu \neq \nu$: "hopping-element" on-site from $\nu \rightarrow \mu$ due to existence of other atoms

$$\text{(ii)} \qquad \int d\boldsymbol{r} \, \phi_\mu^*(\boldsymbol{r}) \cdot v_{\text{at}}(\boldsymbol{r}) \cdot \phi_\nu(\boldsymbol{r} - \boldsymbol{R}) \, , \qquad \boldsymbol{R} \neq \boldsymbol{0} \tag{2.16}$$

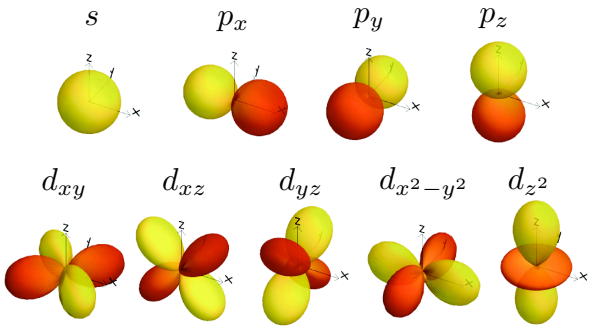


Figure 2.1: Atomic s,p,d orbitals | modified fig. from [15]

typical hopping elements, which describe the electronic transition from $\phi_\nu(\boldsymbol{r}-\boldsymbol{R})$ to $\phi_\mu(\boldsymbol{r})$; usually only the valence electrons are allowed to hop.

3-center-integrals:

$$\int d\boldsymbol{r}\, \phi_\mu^*(\boldsymbol{r}) \cdot v_{\text{at}}(\boldsymbol{r}-\boldsymbol{R}) \cdot \phi_\nu(\boldsymbol{r}-\boldsymbol{R'})\,, \qquad \boldsymbol{R} \neq \boldsymbol{0} \neq \boldsymbol{R'} \neq \boldsymbol{R} \tag{2.17}$$

These integrals are very small for atomic orbitals $\phi_\mu(\boldsymbol{r})$. Therefore they are usually neglected in theoretical tight-binding descriptions. Also in this work they will be neglected in the determination of the tight-binding parameters.

The matrix elements of the kinetic energy T with the atomic orbitals are of the same form as the 1-center-integrals and 2-center-integrals. Therefore the on-site energies and hopping elements contain also the kinetic energy contribution.

Summing up, one needs a parametrization for the on-site elements and hopping integrals. A common parametrization for the hopping elements is the so-called Slater-Koster parametrization [18]. The hopping elements of the type 2.16 are related by symmetry operations which can be exploited to parametrize the hopping elements with a minimal set of parameters, the Slater-Koster parameters, which are described in this section. Each hopping element depends on the distance and the direction of the bonding between the corresponding atoms. Therefore the parametrization of the hopping elements consists of a angular-dependent description via the Slater-Koster transformations and a distance-dependent parametrization by Mehl *et al.* [3, 4].

Slater-Koster parameters The angular dependence of the basis functions $\phi_\nu(\boldsymbol{r}-\boldsymbol{R})$ is described by s- p- and d-orbitals (see fig. 2.1). Essentially these orbitals are linear combinations of the (complex) spherical harmonics. Therefore, the hopping elements consist of the following matrix elements of $\mathcal{H}$ with the (complex) spherical harmonic functions $|l,m\rangle$:

$$\tilde{V}_{ll'm}^{(i\to j)} = \langle \boldsymbol{n}, i, l, m|\, \mathcal{H}\, |\boldsymbol{n'}, j, l', m\rangle \qquad (m = m')\,, \tag{2.18}$$

where l, l' label the angular-momentum quantum number of the orbitals and m the magnetic quantum number of the orbitals. The magnetic quantum numbers m and m' of the atomic orbitals have to be the same due to selection rules. Neglecting the 3-center-integrals, the hopping elements of the Hamiltonian can be described with only 10 types of Slater-Koster parameters, which are shown in fig. 2.2. The Slater-Koster parameters are linear-combinations of the matrix elements 2.18, e.g. the Slater-Koster parameter $V_{sp\sigma}$ is $\tilde{V}_{010}$ or $V_{pp\pi}$ is $\frac{1}{2}(\tilde{V}_{11-1} + \tilde{V}_{111})$ (for more information see [19]). From now on the Slater-Koster parameters are denoted as $V_{ll'm}$, where l,l' are the angular momentum quantum numbers in orbital notation (i.e. s,p,d) and m stands for a σ-, π- or δ-like symmetry of the bonding. As one can see in figure 2.2 these 10 parameters are sufficient to describe a system with one atom per unit-cell ($i = j$) and where only nearest-neighbour coupling along the z-axis as bonding direction is considered. Then the hopping elements of the Hamiltonian would take the following form in the $(s, p_x, p_y, p_z, d_{xy}, d_{xz}, d_{yz}, d_{x^2-y^2}, d_{z^2})$-representation:

$$\underline{\boldsymbol{H}}(R \cdot \boldsymbol{e}_z) = \begin{pmatrix} V_{ss\sigma} & 0 & 0 & V_{sp\sigma} & 0 & 0 & 0 & 0 & V_{sd\sigma} \\ 0 & V_{pp\pi} & 0 & 0 & 0 & 0 & V_{pd\pi} & 0 & 0 \\ 0 & 0 & V_{pp\pi} & 0 & 0 & V_{pd\pi} & 0 & 0 & 0 \\ -V_{sp\sigma} & 0 & 0 & V_{pp\sigma} & 0 & 0 & 0 & 0 & V_{pd\sigma} \\ 0 & 0 & 0 & 0 & V_{dd\delta} & 0 & 0 & 0 & 0 \\ 0 & 0 & -V_{pd\pi} & 0 & 0 & V_{dd\pi} & 0 & 0 & 0 \\ 0 & -V_{pd\pi} & 0 & 0 & 0 & 0 & V_{dd\pi} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_{dd\delta} & 0 \\ V_{sd\sigma} & 0 & 0 & -V_{pd\sigma} & 0 & 0 & 0 & 0 & V_{dd\sigma} \end{pmatrix}, \quad (2.19)$$

where $[\underline{\boldsymbol{H}}]^{\nu}_{\mu}(R \cdot \boldsymbol{e}_z) = H^{\boldsymbol{mj}\nu}_{\boldsymbol{ni}\mu}$ with $\boldsymbol{R}_n - \boldsymbol{R}_m = R \cdot \boldsymbol{e}_z$ ($\boldsymbol{R}_n \neq \boldsymbol{R}_m$) the nearest-neighbour bonding direction and $i = j$.

In general one needs the hopping elements for arbitrary distances and arbitrary directions of the bondings. Of course the hopping elements and also the Slater-Koster parameters depend on the chemical type of the basis atoms.

Angular-dependent parametrization Due to the angular dependence of spherical harmonic functions, all hopping elements along an arbitrary bonding direction $\boldsymbol{R}$ can be expressed as linear combinations of 10 Slater-Koster parameters. One has to rotate the matrix $\underline{\boldsymbol{H}}(R \cdot \boldsymbol{e}_z)$ containing the hopping elements for a bonding direction along the z-axis into the matrix with the hopping elements along the $\boldsymbol{R}$-direction:

$$\underline{\boldsymbol{H}}(\boldsymbol{R}) = \underline{\boldsymbol{U}}^{\dagger} \cdot \underline{\boldsymbol{H}}(R \cdot \boldsymbol{e}_z) \cdot \underline{\boldsymbol{U}}\,, \quad (2.20)$$

where $\underline{\boldsymbol{U}}$ is the unitary matrix, which changes the representation of the s-, p- and d-orbitals with the z-axis as quantization axis into the representation with the $\boldsymbol{R}$-axis as quantization axis. The matrix $\underline{\boldsymbol{U}}$ depends on the directional cosines of the bonding vector $\boldsymbol{R}$ [19]. These transformations are also called Slater-Koster transformations.

The explicit form of all these transformations can be found in the table A.1 in appendix A. One simple example for a hopping element between two atoms with a s- and p_z-orbital is shown in figure 2.3.

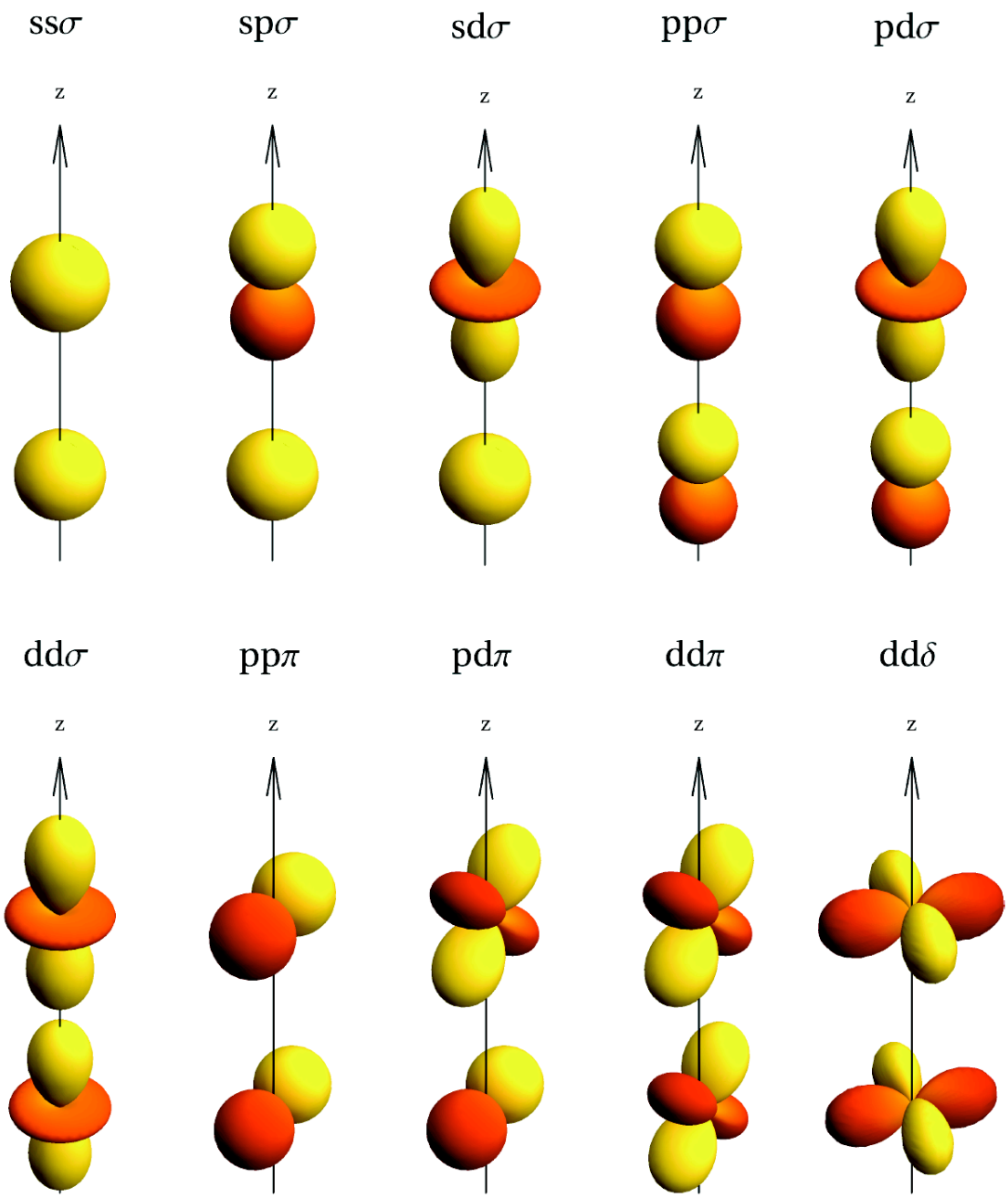


Figure 2.2: All 10 Slater-Koster parameters. σ, π and δ describe the symmetry with respect to the bonding axis and they correspond to magnetic quantum numbers m of the angular momentum. | fig. from [15]

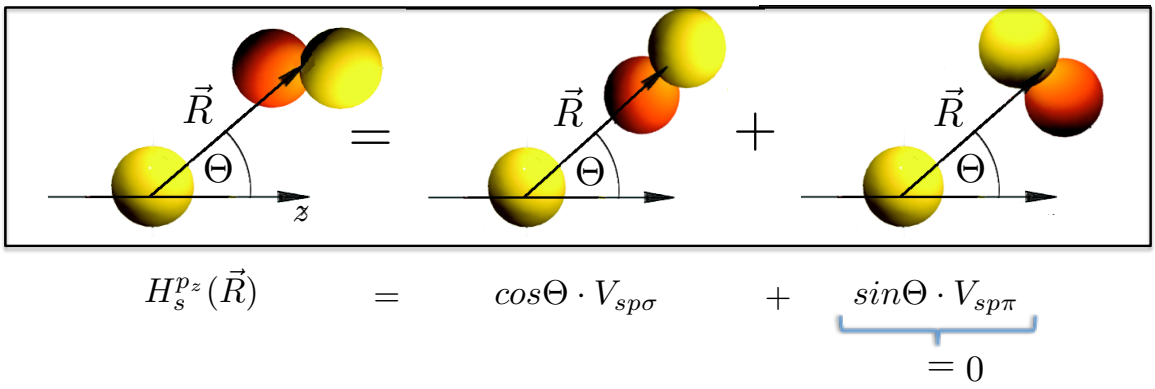


Figure 2.3: Slater-Koster transformation for s and p_z orbital. $H_s^{p_z}$ is a linear combination of the $V_{sp\sigma}$-case and the $V_{sp\pi}$-case, which is zero due to symmetry.

Making use of the Slater-Koster transformations hopping parameters along an arbitrary direction can be expressed by 10 Slater-Koster parameters. Of course the strength of the hopping depends very strongly on the distance between the orbitals. Hence, one needs also a distance-dependent parametrization for the Slater-Koster parameters.

Distance-dependent parametrization There are many distance parametrizations for the Slater-Koster parameters [20, 21, 22, 23]. In this work a distance parametrization by Mehl *et al.* is used [3]. In this parametrization the distance dependence of each Slater-Koster parameter $V_{ll'm}(R)$ is described by 4 other parameters $a_{ll'm}$, $b_{ll'm}$, $c_{ll'm}$ and $d_{ll'm}$:

$$V_{ll'm}^{(i\to j)} = \underbrace{(a_{ll'm}^{(i\to j)} + b_{ll'm}^{(i\to j)} \cdot R + c_{ll'm}^{(i\to j)} \cdot R^2)}_{\text{polynomial part}} \cdot \underbrace{e^{-(d_{ll'm}^{(i\to j)})^2 \cdot R}}_{\text{exp. part}} \cdot f_c(R)\,, \tag{2.21}$$

where

$$f_c(R) = \begin{cases} \frac{1}{1+\exp[(R-R_c+5L_c)/L_c]} & , R \le R_c \\ 0 & , R > R_c \end{cases} \tag{2.22}$$

is a Fermi-Dirac-like cutoff-function with R_c as the cutoff-radius and L_c as the broadening of the cutoff-function. The distances R are in atomic units and the Slater-Koster parameters are in units of Rydberg in the Mehl *et al.* parametrization.

In figure 2.4 one can see the distance-dependent behaviour of some of the Slater-Koster parameters for Fe-Fe hopping elements. The distance parametrization could lead to unphysical Slater-Koster parameters in the case of too small distances. An analogous parametrization is used for the overlap matrix elements.[1]

Besides the hopping elements also a parametrization for the on-site energies is needed, which is of the following form in the Mehl *et al.* parametrization:

$$\epsilon_{i\mu} = (\alpha_{i\mu} + \beta_{i\mu} \cdot \rho_i^{2/3} + \gamma_{i\mu} \cdot \rho_i^{4/3} + \chi_{i\mu} \cdot \rho_i^2) \tag{2.23}$$

[1] When adopting the parametrization scheme of Mehl *et al.* one should take care of the overlap matrix elements, because there are two slightly different parametrizations [5, 6].

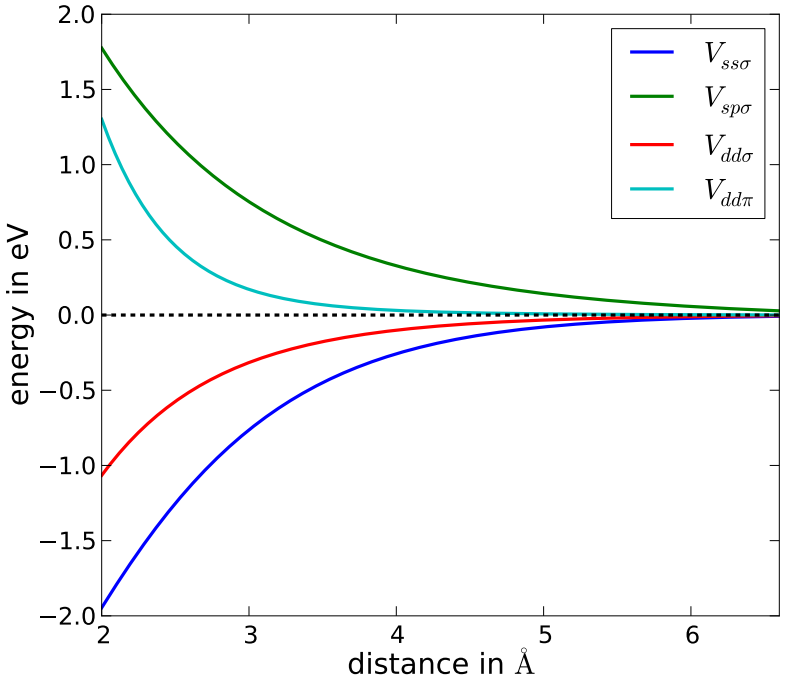


Figure 2.4: Distance-dependence of selected Fe-Fe hopping elements. While the distance parametrization provides reasonable Fe-Fe hopping elements for distances $\geq 2\,\text{Å}$, it is not able to reproduce the behaviour for smaller distances.

with

$$\rho_i = \sum_{j \neq i} e^{-\lambda_j^2 \cdot R_{ij}} \cdot f_c(R_{ij}) \tag{2.24}$$

as local atomic density at atom i with additional parameters λ_j , which depends on the chemical type of the atom j, and $R_{ij} = |\boldsymbol{R}_i - \boldsymbol{R}_j|$ the distance between atom i and j.

In total 80 parameters are needed to describe the hopping elements between chemically equivalent atoms and another 13 parameters for the on-site elements. To describe a binary system, consisting of two types of basis atoms A and B, 93 parameters for each bonding type $A-A$ and $B-B$ are needed. In principle also 112 parameters would be necessary to describe $A-B$-bondings (80 parameters for the Slater-Koster parameters + 4 additional for each $V_{ps\sigma}$, $V_{dp\sigma}$, $V_{dp\pi}$ and $V_{ds\sigma}$), but unfortunately there are almost no provided parameters for binary systems by Mehl *et al.* Therefore, the following ansatz is used to describe the Slater-Koster parameters for the $A-B$-hopping:

$$V_{ll'm}^{\text{A-B}} = \frac{1}{2}(V_{ll'm}^{\text{A-A}} + V_{ll'm}^{\text{B-B}})\,. \tag{2.25}$$

Although these parameters are a rough approximations, it can be shown that one can obtain reasonable results for Fe-Pt systems [16].

The parameter sets of Mehl *et al.* are fitted to results of *ab-initio* bandstructures of the corresponding equilibrium bulk-geometry. Therefore, these parameters provide a

proper description of the bulk structure, but are not as accurate for binary systems and especially systems, whose geometry is very different compared to the bulk geometry. This problem is known as transferability problem [24].

It should be mentioned that the hopping parameters are not spin-dependent, therefore an additional model Hamiltonian is needed to describe magnetism (see section 2.6). Also only valence electrons have the possibility to hop, whereas the energy of the core electrons is contained in the on-site elements.

2.4 Electronic structure properties

The parametrization by Mehl *et al.* provides a description of the Hamiltonian $H_{i\mu}^{j\nu}(\boldsymbol{k})$ and the overlap matrix $S_{i\mu}^{j\nu}(\boldsymbol{k})$. After solving the eigenvalue-equation

$$\underline{\boldsymbol{H}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_n(\boldsymbol{k}) = \varepsilon_n(\boldsymbol{k}) \underline{\boldsymbol{S}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_n(\boldsymbol{k}) \tag{2.26}$$

one can calculate the Fermi energy, the charges and the density of states (DOS) using the eigenenergies $\varepsilon_n(\boldsymbol{k})$ and the eigenvectors $\boldsymbol{\Psi}_n(\boldsymbol{k})$. The band index n numbers the different eigenenergies and eigenvectors.

2.4.1 Fermi energy

The Fermi energy ε_F is determined via the equation

$$N_{e^-} = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F)\,, \tag{2.27}$$

where

$$f(\varepsilon, \varepsilon_F) = \frac{1}{\exp[\beta \cdot (\varepsilon - \varepsilon_F)] + 1} \tag{2.28}$$

is the Fermi-Dirac distribution function for the electrons and N_{e^-} is the total number of valence electrons in the system. A thermal smearing $\beta^{-1} = k_B \cdot T$ is introduced to make the calculation of the Fermi energy numerically more stable.

2.4.2 Density of states (DOS)

The DOS is defined as

$$D(\varepsilon) = \sum_{\boldsymbol{k},n} \delta(\varepsilon - \varepsilon_{\boldsymbol{k},n})\,. \tag{2.29}$$

In this work a Lorentzian-broadening function is used instead of the δ-function:

$$D(\varepsilon) = \sum_{\boldsymbol{k},n} \frac{1}{\pi} \cdot \frac{\Gamma}{\Gamma^2 + (\varepsilon - \varepsilon_{\boldsymbol{k},n})^2}\,, \tag{2.30}$$

where Γ is the width of the Lorentzian-function. One has to be very careful in choosing a reasonable Γ: While a very large value smears out fine structure of the DOS, a too small width leads to many sharp peaks in the DOS.

Besides the (total) DOS, it is also enlightening to take a look at the partial DOS, such as atom- and orbital-resolved DOS. The partial DOS of the j-th basis atom and ν-th orbital is calculated as follows:

$$D_{j\nu}(\varepsilon) = \sum_{\boldsymbol{k},n} \delta(\varepsilon - \varepsilon_{\boldsymbol{k},n}) \cdot (\boldsymbol{\Psi}^{\dagger}_{\boldsymbol{k},n})_{j\nu} \cdot (\underline{\boldsymbol{S}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} \,. \tag{2.31}$$

If an orthogonal set of basis functions would be used, the eigenvectors would be orthonormal: $\boldsymbol{\Psi}^{\dagger}_n \cdot \boldsymbol{\Psi}_m = \delta_{nm}\, \forall n, m$. In the case of a non-orthogonal set of basis functions, it can be shown that

$$\boldsymbol{\Psi}^{\dagger}_n \cdot \underline{\boldsymbol{S}} \cdot \boldsymbol{\Psi}_m = \delta_{nm} \,, \qquad \forall n, m \,. \tag{2.32}$$

With eq. 2.32 follows:

$$D(\varepsilon) = \sum_{j,\nu} D_{j\nu}(\varepsilon) \,. \tag{2.33}$$

2.4.3 Band energy

The band energy is defined as follows:

$$E_{\text{band}} = \sum_{\boldsymbol{k},n} \varepsilon_{\boldsymbol{k},n} \cdot f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \,. \tag{2.34}$$

This is a contribution to the total energy expression (see section 2.5/2.6), where also double-counting terms enter.

2.4.4 Charges

In the following sections the self-consistency of the tight-binding scheme is explained in detail. One fundamental point of the self-consistent cycle is the determination of the atomic charges. The charge calculation for a non-magnetic system is shown here. The charge of the j-th atom in its μ-th orbital displays as follows:

$$n^{\text{Mull}}_{j\nu} = 2 \cdot \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot (\boldsymbol{\Psi}^{\dagger}_{\boldsymbol{k},n})_{j\nu} \cdot (\underline{\boldsymbol{S}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} \,, \tag{2.35}$$

where the factor 2 arises due to the spin-degeneracy in a non-magnetic system. To be precise, the above charges are the so-called Mulliken charges. They fulfil the normalization condition

$$\sum_{j,\nu} n^{\text{Mull}}_{j\nu} = N_{e^-} \,, \tag{2.36}$$

whereas the so-called net charges

$$n^{\text{net}}_{j\nu} = 2 \cdot \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot (\boldsymbol{\Psi}^{\dagger}_{\boldsymbol{k},n})_{j\nu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} \,, \tag{2.37}$$

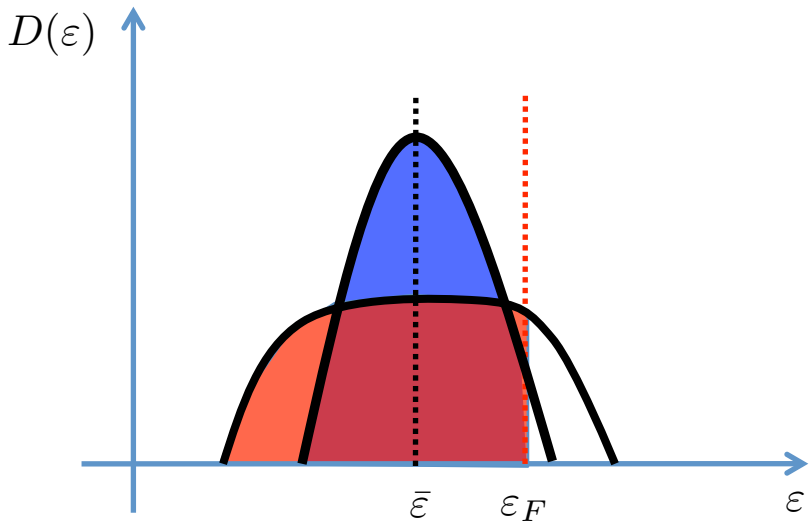


Figure 2.5: Charge transfer to the surface-atoms in a simple DOS model. Red and blue filling in the figure represents the filling of the electrons in the bulk DOS and surface DOS; the area under the two curves and the Fermi energy is the same, therefore there is more charge on the surface-atom

do not fulfil eq. 2.36. Net charges and Mulliken charges are identical for an orthogonal set of basis functions.

2.5 Local charge neutrality

In the parametrized tight-binding scheme described in section 2.3 the on-site energies are fixed. This could lead to problems in calculating the charges for systems of low symmetry such as slab systems or systems with non-equivalent basis atoms. One can obtain non-physical results due to large charge transfers and a constraint for the charge becomes necessary. The reason for the occurrence of these large charge transfers can be understood in a simple DOS model for a thick slab system. In the following the local (atom-specific) DOS of a surface-atom and of an atom in the bulk-like middle of the slab system are compared. For the sake of simplicity the on-site energy of the surface atom and bulk atom is fixed to the same value and the local DOS is symmetric with respect to its center value ε. The width of the local DOS depends on the hopping elements and the number of neighbours. Typically the width of the DOS for the surface atom is smaller than for the bulk-atom. In figure 2.5 the charge transfer to the surface atoms is clearly visible. To prevent very large charge transfers, the following ansatz for a constraint is used:

$$E_{\mathrm{LCN}} = \frac{U_{\mathrm{LCN}}}{2} \cdot \sum_i (n_i^{\mathrm{Mull}} - n_i^0)^2 \,, \tag{2.38}$$

where n_i^{Mull} is the Mulliken charge on the i-th atom, n_i^0 is the (desired) bulk-value of the Mulliken charge and U_{LCN} is the local charge neutrality constant. For simplicity the following derivation is done only for an orthogonal set of basis functions, therefore $n_i := n_i^{\text{Mull}} = n_i^{net}$. To determine the form of Hamiltonian, which corresponds to the above energy E_{LCN}, one should minimize the following Lagrange function:

$$F = E_{\text{tot}}^0 + \frac{U_{\text{LCN}}}{2} \cdot \sum_i (n_i - n_i^0)^2 + \sum_{\boldsymbol{k},n} \alpha_{\boldsymbol{k},n} \cdot \left[|\boldsymbol{\Psi}_{\boldsymbol{k},n}|^2 - 1 \right], \tag{2.39}$$

where

$$E_{\text{tot}}^0 = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\mu,\nu} \sum_{i,j} (\boldsymbol{\Psi}_{\boldsymbol{k},n}^\dagger)_{i\mu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} \cdot [H_0]_{i\mu}^{j\nu} \tag{2.40}$$

is the total energy without the constraints and the remaining two terms are the constraints. The first constraint is the local charge neutrality term (see eq. 2.38) and the other one takes into account the normalization of the wave functions.

$$\begin{aligned} F &= \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \sum_{i,j} \sum_{\mu,\nu} (\boldsymbol{\Psi}_{\boldsymbol{k},n}^\dagger)_{i\mu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} \cdot [H_0]_{i\mu}^{j\nu} + \\ &\quad \frac{U_{\text{LCN}}}{2} \cdot \sum_i \left[\sum_{\boldsymbol{k},n,\mu} |(\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu}|^2 \cdot f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) - n_i^0 \right]^2 + \sum_{\boldsymbol{k},n} \alpha_{\boldsymbol{k},n} \cdot \left[\sum_{i,\mu} |(\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu}|^2 - 1 \right] \\ &\Rightarrow \frac{1}{f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F)} \cdot \frac{\partial F}{\partial (\boldsymbol{\Psi}_{\boldsymbol{k},n}^\dagger)_{i\mu}} \\ &= \sum_{j\nu} [H_0]_{i\mu}^{j\nu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{j\nu} + U_{\text{LCN}} \cdot (n_i - n_i^0) \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu} + \tilde{\alpha}_{\boldsymbol{k},n} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu} = 0 \end{aligned}$$

The Lagrange parameters $-\tilde{\alpha}_{\boldsymbol{k},n} = -\frac{\alpha_{\boldsymbol{k},n}}{f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F)}$ can be identified as bandenergies $\varepsilon_{\boldsymbol{k},n}$ of an eigenvalue problem for the Hamiltonian:

$$H_{i\mu}^{j\nu} = [H_0]_{i\mu}^{j\nu} + U_{\text{LCN}} \cdot (n_i - n_i^0) \cdot \delta_{ij} \delta_{\mu\nu} \,. \tag{2.41}$$

What is the physics behind the local charge neutrality constraint?

The constraint for the charge neutrality shifts the on-site energies depending on $(n_i - n_i^0)$. If $(n_i - n_i^0) < 0$ the on-site energy of the corresponding atom is decreased, so that the charge on this atom increases (see figure 2.5). The parameter U_{LCN} in the charge constraint has to be chosen large enough to assure $n_i \approx n_i^0$. Theoretically n_i is equal n_i^0 for $U_{\text{LCN}} \to \infty$, but of course this can not be used numerically. A good value for preventing large charge transfers is $U_{\text{LCN}} = 5\,\text{eV}$.

With the local charge neutrality term the Hamiltonian depends on the charges and the charges itself are calculated from the eigenvectors and eigenenergies of the Hamiltonian. This leads to the possibility to calculate the charges of the system in a self-consistent scheme. In figure 2.6 the self-consistent scheme is displayed. Besides the above linear mixing, which converges very slowly, but is very stable, there is also the possibility to use Broyden mixing [25] for faster convergence.

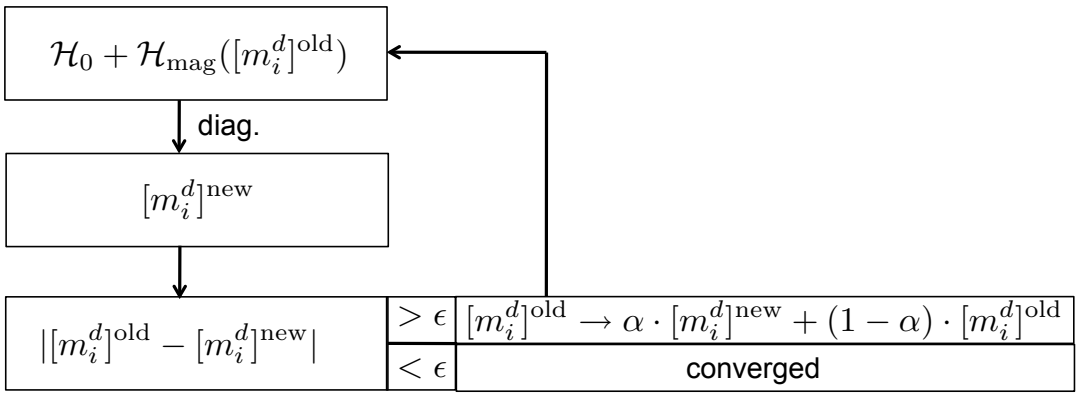


Figure 2.6: Self-consistent cycle via the local charge neutrality.

For a system with equivalent atoms the Hamiltonian is fixed to the parametrized Hamiltonian, which is described in section 2.3. Therefore the self-consistent charge calculation yields no advantages for homogeneous systems. But for systems with inequivalent atoms, the on-site energies are shifted due to the local charge neutrality constraint until the system reaches a stable equilibrium with respect to the charges.

In the self-consistent scheme with the local charge neutrality, the band energy 2.34 is not the whole part of the total energy and there is a double counting part, which has to be taken into account. This double counting term is not derived in the following, but in the section on the Stoner-model (see section 2.6) an additional double counting term is introduced, which derivation (eqs. 2.55-2.58) shows how one can derive eq. 2.43. The total energy has the following form:

$$E_{\mathrm{tot}} = E_{\mathrm{band}} - \frac{U_{\mathrm{LCN}}}{2} \cdot \sum_i (n_i^2 - (n_i^0)^2)\,. \tag{2.42}$$

For systems with identical basis atoms, $n_i^0 = n^0$, the double counting can be rewritten as

$$\frac{U_{\mathrm{LCN}}}{2} \cdot \sum_i (n_i - n_i^0)^2 \tag{2.43}$$

using charge conservation $\sum_i n_i = \sum_i n_0$.

All equations throughout this section are valid only for an orthogonal set of basis functions. For a non-orthogonal set of basis functions the charge n_i in the equations 2.41-2.43 has to be replaced by the Mulliken charge n_i^{Mull} and the additional charge constraint in the Hamiltonian has the following form:

$$\frac{U_{\mathrm{LCN}}}{2} \cdot \left((n_i^{\mathrm{Mull}} - n_i^0) + (n_j^{\mathrm{Mull}} - n_j^0)\right) \cdot S_{i\mu}^{j\nu}\,. \tag{2.44}$$

The structure of the double counting term remains unchanged.

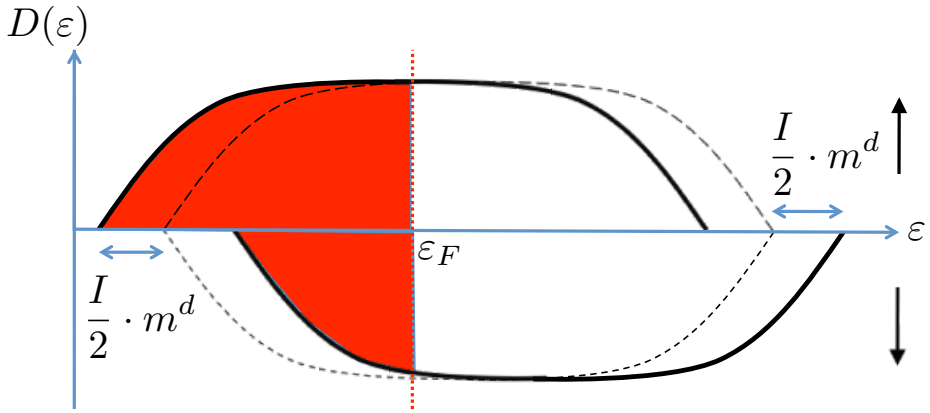


Figure 2.7: Stoner model in a simple DOS picture. The dashed lines display the DOS of the non-magnetic case. Due to an exchange splitting introduced by the Stoner model the DOS is shifted depending on its spin. The band filling, represented by the red colour, is different in the spin channels. Therefore the system develops a magnetic moment.

2.6 Stoner model

As mentioned in section 2.3 the parametrization for the hopping elements does not include magnetism, because the hopping elements are spin independent, i.e. $V_{ll'm}^{\uparrow\to\uparrow} = V_{ll'm}^{\downarrow\to\downarrow}$ and $V_{ll'm}^{\uparrow\to\downarrow} = 0$. In this work a simple Stoner model is applied to model magnetism. Here, an exchange splitting between the electronic majority-bands ($\uparrow$-bands) and minority-bands ($\downarrow$-bands) is introduced in the same way as in the original Stoner model [7, 8]. The exchange splitting depends on the magnetic d-moment of the corresponding atom and a Stoner parameter I:

$$\varepsilon_{i\mu}^{\text{exc}} = I_{i\mu} \cdot m_i^d \,. \tag{2.45}$$

The calculation of the magnetic moments in the tight-binding framework is explained later. At the moment it is enough to know, that $m_i^d = N_{i,d}^{\uparrow} - N_{i,d}^{\downarrow}$ for collinear magnets, where $N_{i,d}^{\sigma}\,(\sigma = \uparrow, \downarrow)$ is the number of electrons in the $\uparrow$- , $\downarrow$-d-bands. In figure 2.7 one can see the connection between the exchange splitting and the filling of $\uparrow$- and $\downarrow$-bands via a simple DOS consideration.

For the case of collinear magnetism the Stoner-part of the Hamiltonian in spin-space has the following form:

$$[H_{\text{mag}}]_{i\mu}^{j\nu} = \begin{pmatrix} -\frac{I_{i\mu}}{2} \cdot m_i^d & 0 \\ 0 & \frac{I_{i\mu}}{2} \cdot m_i^d \end{pmatrix} \cdot \delta_{ij}\delta_{\mu\nu} \,. \tag{2.46}$$

Therefore the Stoner-part modifies exclusively the on-site energies of the system. In the Stoner-part only the magnetic d-moments are used to determine the exchange splitting, because in $3d$-transition metals (like Fe) the magnetism is originated mainly from the $3d$-electrons. This can be shown again with the help of a simple DOS-model and the Stoner criterion: $I \cdot D(\varepsilon_F) > 1$ for ferromagnetic materials, where $D(\varepsilon_F)$ is the DOS at the Fermi energy.

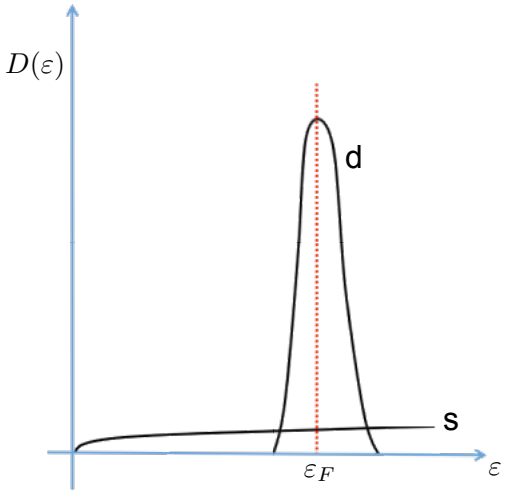


Figure 2.8: Simplified DOS of a $3d$-transition metal. The partial DOS in the s-states at the Fermi energy is much lower than the partial DOS in the d-states. Therefore the d-states dominate the magnetic behaviour.

Essential for the formation of ferromagnetism is the density of states at the Fermi energy. In figure 2.8 the simplified DOS of a $3d$-transition metal is shown. Typically for transition metals the DOS at the Fermi energy consists mainly of the partial DOS of the d-orbitals. As a consequence, a small exchange splitting in the d-states lead to large magnetic d-moments, so that $|m^d| \gg |m^s|,|m^p|$. Using only the d-moments to determine the exchange splitting simplifies the below described self-consistent tight-binding calculation without loosing much accuracy.

The Hamiltonian depends on the magnetic d-moments, which are calculated from the eigenenergies and eigenvectors of the Hamiltonian. Now one can calculate the magnetic moments in a self-consistent scheme until convergence is reached. In figure 2.9 the self-consistent scheme, together with the local charge neutrality described in section 2.5, is displayed. The local charge neutrality part shifts the on-site energies of the atoms until the charges are converged as described in detail in section 2.5. The Stoner-model modifies also the on-site energies of the system, but now an exchange splitting between the on-site energies of the $\uparrow$- and $\downarrow$-bands is introduced. This splitting changes during the self-consistent scheme until the magnetic moments are converged. In the converged case the system has reached a stable equilibrium with respect to the charges and the magnetic moments.

Now one should take a look, how these magnetic moments can be calculated. The magnetic moments and also the (net-)charges can be determined with the help of the density matrix $\underline{\boldsymbol{\rho}}$. The density matrix for the i-th atom and μ-th orbital has the following form in spin-space with the z-axis as quantization axis:

$$\underline{\boldsymbol{\rho}}_{i\mu} = \begin{pmatrix} \rho_{i\mu}^{\uparrow\uparrow} & \rho_{i\mu}^{\uparrow\downarrow} \\ \rho_{i\mu}^{\downarrow\uparrow} & \rho_{i\mu}^{\downarrow\downarrow} \end{pmatrix} , \tag{2.47}$$

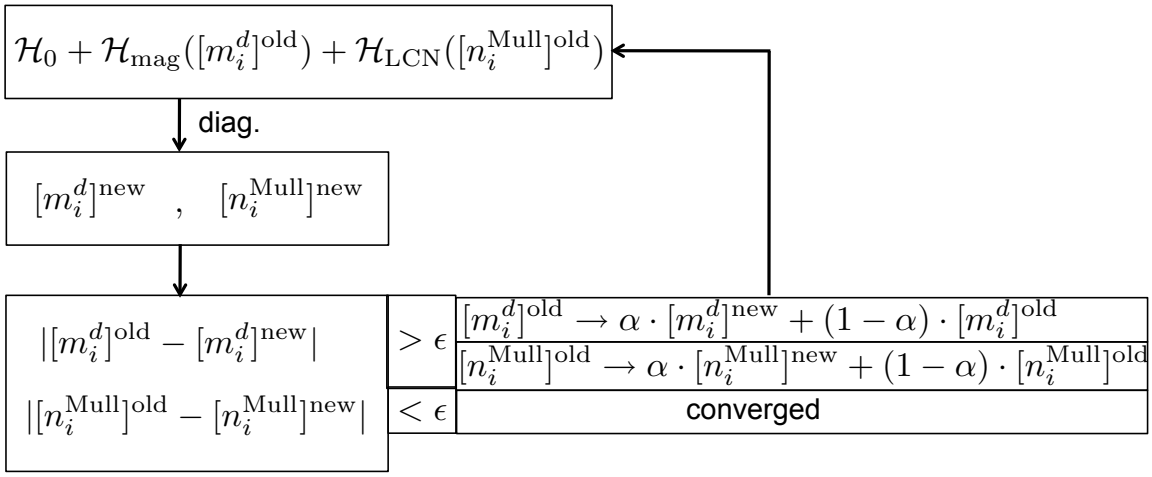


Figure 2.9: Self-consistent cycle via the Stoner part and the local charge neutrality

with

$$\rho_{i\mu}^{\sigma\sigma'} = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n}^{\dagger})_{i\mu}^{\sigma} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu}^{\sigma'} . \tag{2.48}$$

The (net-)charge of the i-th atom and μ-th orbital is the trace of the density matrix:

$$n_{i\mu}^{\mathrm{net}} = tr[\underline{\boldsymbol{\rho}}_{i\mu}] = \rho_{i\mu}^{\uparrow\uparrow} + \rho_{i\mu}^{\downarrow\downarrow} . \tag{2.49}$$

In the special case of a non-magnetic system one obtains the result 2.37. The magnetic moment of the i-th atom and μ-th orbital can be determined for all the three spatial directions as follows:

$$(\boldsymbol{m}_{i\mu})_x = 2\,\mathrm{Re}[\rho_{i\mu}^{\uparrow\downarrow}] \tag{2.50}$$

$$(\boldsymbol{m}_{i\mu})_y = 2\,\mathrm{Im}[\rho_{i\mu}^{\uparrow\downarrow}] \tag{2.51}$$

$$(\boldsymbol{m}_{i\mu})_z = \rho_{i\mu}^{\uparrow\uparrow} - \rho_{i\mu}^{\downarrow\downarrow} \tag{2.52}$$

These equations can be directly derived from the definition

$$\begin{aligned} [\boldsymbol{m}_{i\mu}]_\alpha &= \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\sigma\sigma'} (\boldsymbol{\Psi}_{\boldsymbol{k},n}^{\dagger})_{i\mu}^{\sigma} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu}^{\sigma'} \cdot [\underline{\boldsymbol{\sigma}}_\alpha]_{\sigma\sigma'} \\ &= tr[\underline{\boldsymbol{\rho}}_{i\mu}^{\dagger} \cdot \underline{\boldsymbol{\sigma}}_\alpha] \qquad \alpha = x, y, z , \end{aligned} \tag{2.53}$$

where $\underline{\boldsymbol{\sigma}}_\alpha$ are the well-known Pauli-matrices. Similarly to the charges there is a distinction between magnetic net moments and magnetic Mulliken moments. With eqs. 2.50-2.52 the magnetic net moments are calculated, whereas the magnetic Mulliken moments can be determined by

$$[\rho_{i\mu}^{\sigma\sigma'}]^{\mathrm{Mull}} = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n}^{\dagger})_{i\mu}^{\sigma} \cdot (\underline{\boldsymbol{S}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_{\boldsymbol{k},n})_{i\mu}^{\sigma'} . \tag{2.54}$$

Normally, one should use here the Mulliken moments. However, using the net moments instead of the Mulliken moments does not make a big difference in the converged results and one could stick to the simpler net moments.

We close this section with a derivation of an expression for the exchange energy contribution to the total energy as promised in section 2.5. The total energy E_{tot} is not any more only the band energy E_{band} (see eq. 2.34), there is a double counting term, which has to be considered. One can show [26], that the total energy of the Hamiltonian $\mathcal{H}_0 + \mathcal{H}_{\text{mag}}$ is

$$E_{\text{tot}} = E^0_{\text{tot}} - \frac{1}{4}\sum_{i,\mu} I_{i,\mu} \cdot m_i^\mu \cdot m_i^d \,, \tag{2.55}$$

where

$$E^0_{\text{tot}} = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\mu,\nu}\sum_{i,j}\sum_{\sigma,\sigma'} (\boldsymbol{\Psi}^\dagger_{\boldsymbol{k},n})^\sigma_{i\mu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})^{\sigma'}_{j\nu} \cdot [H_0]^{j\nu\sigma'}_{i\mu\sigma} \tag{2.56}$$

is the total energy of the non-magnetic system. Easily accessible is the band energy, which is calculated in the following:

$$\begin{aligned}
E_{\text{band}} &= \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \varepsilon_{\boldsymbol{k},n} \\
&= \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\mu,\nu}\sum_{i,j}\sum_{\sigma,\sigma'} (\boldsymbol{\Psi}^\dagger_{\boldsymbol{k},n})^\sigma_{i\mu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})^{\sigma'}_{j\nu} \cdot \\
&\quad \left([H_0]^{j\nu\sigma'}_{i\mu\sigma} - \frac{I_{i,\mu}}{2} \cdot \sigma \cdot m_i^d \cdot \delta_{\sigma\sigma'}\delta_{ij}\delta_{\mu\nu} \right) \\
&= E^0_{\text{tot}} - \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \sum_{i,\mu,\sigma} |(\boldsymbol{\Psi}_{\boldsymbol{k},\boldsymbol{n}})^\sigma_{i\mu}|^2 \cdot \frac{I_{i,\mu}}{2} \cdot m_i^d \cdot \sigma \\
&= E^0_{\text{tot}} - \sum_{i,\mu} \frac{I_{i,\mu}}{2} \cdot m_i^d \cdot m_i^\mu \,.
\end{aligned} \tag{2.57}$$

If one compares eq. 2.55 with eq. 2.57, one can determine the double counting:

$$E_{\text{tot}} = E_{\text{band}} + \sum_i \frac{I_{i,\mu}}{4} \cdot m_i^\mu \cdot m_i^d \,. \tag{2.58}$$

2.7 Spin-orbit coupling

The spin-orbit coupling (SOC) is a relativistic effect with the energy scale well below 0.5 eV. A very important aspect of SOC is the symmetry breaking it causes in a ferromagnet. Many interesting effects like the Rashba-splitting [27] or the Dzyaloshinskii-Moriya interaction (DMI) [28, 29] are based on SOC.

Due to the relativistic nature of SOC one has to examine the Dirac equation to understand its origin. In a non-relativistic expansion of the Dirac equation the following expression for SOC can be derived [30]:

$$H_{\text{SOC}} \propto (\nabla V(\boldsymbol{r}) \times \boldsymbol{p}) \cdot \boldsymbol{\sigma} \,, \tag{2.59}$$

where $\boldsymbol{\sigma}$ is the vector containing the Pauli-matrices, $V(\boldsymbol{r})$ is the electrostatic potential and $\boldsymbol{p}$ is the momentum of the electron. In solids the electric field, i.e. $\nabla V(\boldsymbol{r})$, is strongest close to the nucleus, where the potential is almost spherical. Therefore using a spherical potential $V(\boldsymbol{r}) = V(r)$ is a good approximation and the expression 2.59 can be rewritten:

$$H_{\mathrm{SOC}} = \xi(r) \cdot \boldsymbol{L} \cdot \boldsymbol{S}\,, \tag{2.60}$$

with $\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$ the angular momentum, $\boldsymbol{S} = \frac{\hbar}{2} \cdot \boldsymbol{\sigma}$ the spin of the electron and the radial-dependent function $\xi(r) = \frac{1}{2m^2c^2} \cdot \frac{1}{r} \cdot \frac{\partial V(r)}{\partial r}$.

In representation of the atomic orbitals, using $\langle \boldsymbol{r} | i\mu \rangle = R_{i\mu}(r) \cdot Y_{\mu}^{(i)}(\Theta, \phi)$ with $R_{i\mu}(r)$ a radial function and $Y_{\mu}^{(i)}(\Theta, \phi)$ a spherical harmonic function, the expression has the following form:

$$\begin{aligned}
(H_{\mathrm{SOC}})_{i\mu\sigma}^{j\nu\sigma'} &= \langle i, \mu, \sigma |\, \xi(r) \cdot \boldsymbol{L} \cdot \boldsymbol{S} \,| j, \nu, \sigma' \rangle \\
&= \underbrace{\frac{\hbar}{2} \left[\int dr \cdot r^2 \xi(r) R_{i\mu}^{*}(r) \cdot R_{j\nu}(r) \right]}_{\xi_{i\mu,j\nu}} \cdot \\
&\qquad \left[\int_0^{2\pi} \int_0^{\pi} d\Theta d\phi \, \sin\Theta \cdot [Y_{\mu}^{(i)}]^{*}(\Theta, \phi) \cdot Y_{\nu}^{(j)}(\Theta, \phi) \cdot \langle \sigma |\, \boldsymbol{L} \cdot \boldsymbol{\sigma} \,| \sigma' \rangle \right] \\
&= \xi_{i\mu,j\nu} \cdot \langle \mu\sigma |\, \boldsymbol{L} \cdot \boldsymbol{\sigma} \,| \nu\sigma' \rangle \ .
\end{aligned}$$

The function $\xi(r)$ is usually localized near $r = 0$, therefore $\xi_{i\mu,j\nu} \approx \delta_{ij}\, \xi_{\mu\nu}$ is a reasonable approximation. The representation of the angular moment $\boldsymbol{L}$ in atomic orbitals, i.e. $L_{\mu}^{\nu} = \langle \mu |\, \boldsymbol{L} \,| \nu \rangle$, can be found in the appendix in A.1. Important to notice is that $\langle \mu |\, \boldsymbol{L} \,| \nu \rangle$ is block-diagonal with respect to s-,p- and d-orbitals. Therefore only three SOC-parameters per basis atom are necessary for describing SOC in tight-binding:

- $\xi_{i,p}$, SOC-parameter of p-orbitals
- $\xi_{i,d}$, SOC-parameter of d-orbitals
- $\xi_{i,f}$, SOC-parameter of f-orbitals

There is no contribution to SOC from the s-orbital due to $l = 0$. In our code the f-orbitals are not implemented, but usually they have a large contribution to SOC for the rare-earth metals.

Combining these results leads to the following expression for SOC in representation of the atomic orbitals in spin-representation with the z-axis as quantization axis:

$$(H_{\mathrm{SOC}})_{i\mu}^{j\nu} = \xi_{i\mu} \cdot \begin{pmatrix} (L_z)_{\mu}^{\nu} & (L_x)_{\mu}^{\nu} - \mathrm{i} \cdot (L_y)_{\mu}^{\nu} \\ (L_x)_{\mu}^{\nu} + \mathrm{i} \cdot (L_y)_{\mu}^{\nu} & -(L_z)_{\mu}^{\nu} \end{pmatrix} \cdot \delta_{ij}\,, \tag{2.61}$$

where $(L_{\alpha})_{\mu}^{\nu}$ are the aforementioned angular momentum components in atomic orbital representation (see A.1).

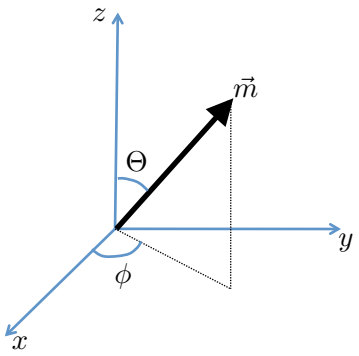


Figure 2.10: Magnetic moment in spherical coordinate representation.

Switching on SOC leads to non-vanishing expectation values for the orbital moments $\langle \boldsymbol{L_i} \rangle$. The following equation is used to determine the orbital moment of the i-th atom:

$$\langle \boldsymbol{L}_i \rangle = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\mu,\nu,\sigma} (\boldsymbol{\Psi}^{\dagger}_{\boldsymbol{k},n})^{\sigma}_{i\mu} \cdot (\boldsymbol{\Psi}_{\boldsymbol{k},n})^{\sigma}_{i\nu} \cdot \boldsymbol{L}^{\nu}_{\mu} \,. \tag{2.62}$$

The above moments are net moments. For a determination of the orbital Mulliken moments one has to include the overlap matrix as in eq. 2.54:

$$\langle \boldsymbol{L}^{\mathrm{Mull}}_i \rangle = \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \sum_{\mu,\nu,\sigma} (\boldsymbol{\Psi}^{\dagger}_{\boldsymbol{k},n})^{\sigma}_{i\mu} \cdot (\underline{\boldsymbol{S}}(\boldsymbol{k}) \cdot \boldsymbol{\Psi}_{\boldsymbol{k},n})^{\sigma}_{i\nu} \cdot \boldsymbol{L}^{\nu}_{\mu} \,. \tag{2.63}$$

2.8 Non-collinear magnetism

2.8.1 Non-collinear magnetism in the unit-cell

Many interesting phenomena appear due to non-collinear magnetism, for example the Néel-state in magnetic frustrated systems [31] or spin-spiral ground states in Cr/W(110) [32], 2Fe/W(110) [33] or even Fe(110)-monolayers [34]. Implementing non-collinear magnetism in the tight-binding scheme is rather straightforward. In figure 2.10 a magnetic moment is pointing in an arbitrary direction, which is defined by the angles Θ and ϕ in the spherical coordinates representation:

$$\boldsymbol{m} = |\boldsymbol{m}| \cdot \begin{pmatrix} \cos\phi \, \sin\Theta \\ \sin\phi \, \sin\Theta \\ \cos\Theta \end{pmatrix} \,. \tag{2.64}$$

We proceed by defining two frames of reference. The global frame is the representation of the spins with the z-axis as quantization axis, whereas the local frame is the representation with the direction of $\boldsymbol{m}$ as quantization axis. The following spin-rotation matrix $\mathcal{U}$ rotates the spin up state $|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ of the global frame into the spin up state $|\uparrow\rangle_{\boldsymbol{m}}$ of

the local frame [35]:

$$\mathcal{U}(\Theta,\phi) = \begin{pmatrix} \mathrm{e}^{-\mathrm{i}\frac{\phi}{2}} \cdot \cos(\frac{\Theta}{2}) & -\mathrm{e}^{-\mathrm{i}\frac{\phi}{2}} \cdot \sin(\frac{\Theta}{2}) \\ \mathrm{e}^{\mathrm{i}\frac{\phi}{2}} \cdot \sin(\frac{\Theta}{2}) & \mathrm{e}^{\mathrm{i}\frac{\phi}{2}} \cdot \cos(\frac{\Theta}{2}) \end{pmatrix} . \tag{2.65}$$

Therefore it follows:

$$|\uparrow\rangle_{\boldsymbol{m}} = \mathcal{U}(\Theta,\phi)\ |\uparrow\rangle = \mathrm{e}^{-\mathrm{i}\frac{\phi}{2}} \cdot \cos\left(\frac{\Theta}{2}\right)\ |\uparrow\rangle + \mathrm{e}^{\mathrm{i}\frac{\phi}{2}} \cdot \sin\left(\frac{\Theta}{2}\right)\ |\downarrow\rangle\ . \tag{2.66}$$

The magnetic part of the Hamiltonian has the following form in its local frame (see eq. 2.46):

$$\begin{aligned} [H_{\mathrm{mag}}]_{i\mu}^{j\nu} &= \begin{pmatrix} -\frac{I_{i\mu}}{2} \cdot m_i^d & 0 \\ 0 & \frac{I_{i\mu}}{2} \cdot m_i^d \end{pmatrix} \cdot \delta_{ij}\delta_{\mu\nu} \\ &= -\frac{I_{i\mu}}{2} \cdot \boldsymbol{m_i^d} \cdot \boldsymbol{\sigma} \cdot \delta_{ij}\delta_{\mu\nu}\,. \end{aligned} \tag{2.67}$$

We will show that this expression is, as expected, invariant under spin-rotations. In the special case of $\boldsymbol{m}_i^d = m_i^d \cdot \boldsymbol{e}_z$ the local frame and global frame are the same. $[H_{\mathrm{mag}}]_{i\mu}^{j\nu}$ can be transformed into the global frame using the transformation matrix $\mathcal{U}$:

$$[H_{\mathrm{mag}}^{\mathrm{global}}]_{i\mu}^{i\mu} = \mathcal{U}(\Theta_i,\phi_i) \cdot [H_{\mathrm{mag}}]_{i\mu}^{i\mu} \cdot \mathcal{U}^{\dagger}(\Theta_i,\phi_i)\,, \tag{2.68}$$

where Θ_i and ϕ_i are the angles defining the direction of the magnetic moment $\boldsymbol{m}_i^d$. This leads to the following expression:

$$\begin{aligned} [H_{\mathrm{mag}}^{\mathrm{global}}]_{i\mu}^{i\mu} &= \frac{I_{i\mu}}{2} \cdot \begin{pmatrix} -m_i^d \cdot \cos\Theta_i & -m_i^d \cdot \sin\Theta_i \cdot \mathrm{e}^{-\mathrm{i}\phi_i} \\ -m_i^d \cdot \sin\Theta_i \cdot \mathrm{e}^{\mathrm{i}\phi_i} & m_i^d \cdot \cos\Theta_i \end{pmatrix} \\ &= \frac{I_{i\mu}}{2} \cdot \begin{pmatrix} -(\boldsymbol{m}_i^d)_z & -[(\boldsymbol{m}_i^d)_x - \mathrm{i}\cdot(\boldsymbol{m}_i^d)_y] \\ -[(\boldsymbol{m}_i^d)_x + \mathrm{i}\cdot(\boldsymbol{m}_i^d)_y] & (\boldsymbol{m}_i^d)_z \end{pmatrix} \\ &= -\frac{I_{i\mu}}{2} \cdot \boldsymbol{m}_i^d \cdot \boldsymbol{\sigma}\,. \end{aligned} \tag{2.69}$$

The components of the magnetic d-moment $(m_i^d)_\alpha$ are determined via equations 2.50-2.52. Treating non-collinear magnetism using equation 2.68 increases the number of variables in the self-consistent scheme (figure 2.9) by a factor of two. For the collinear case one component of the d-moments $[\boldsymbol{m}_i^d]_\alpha$ is sufficient, whereas the complete vector $\boldsymbol{m}_i^d$ is necessary in the non-collinear case.

Performing self-consistent calculations for a non-collinear case makes it necessary to use another constraint, in addition to the local charge neutrality term, in order to pin the directions of the magnetic moments. Following a scheme of R. Gebauer [36], one should minimize the following generalized total energy:

$$E_{\mathrm{tot}}[n(\boldsymbol{r}),\boldsymbol{m}(\boldsymbol{r})] = E_{\mathrm{tot}}^0[n(\boldsymbol{r}),\boldsymbol{m}(\boldsymbol{r})] + \lambda \cdot \int d\boldsymbol{r}\,(\boldsymbol{m}(\boldsymbol{r}) - \bar{\boldsymbol{m}}(\boldsymbol{r}))^2\,, \tag{2.70}$$

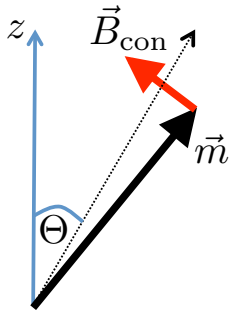


Figure 2.11: Magnetic field $\boldsymbol{B}_{\text{con}}$ in the constraint.

where $E^0_{\text{tot}}[n(\boldsymbol{r}), \boldsymbol{m}(\boldsymbol{r})]$ is the total energy without the constraint and $\lambda \cdot \int d\boldsymbol{r}\,(\boldsymbol{m}(\boldsymbol{r}) - \bar{\boldsymbol{m}}(\boldsymbol{r}))^2$, with $\bar{\boldsymbol{m}}(\boldsymbol{r})$ the predefined magnetic moment, a constraint for the direction of the magnetic moment. In [36] it is shown, that for our purposes the following constraint is able to pin the Θ-angle of the moments up to a small region around the predefined direction:

$$E_{\text{con}} = \lambda \cdot \sum_i \left(\arccos\left(\frac{[\boldsymbol{m}_i^d]_z}{|\boldsymbol{m}_i^d|} \right) - \Theta_i \right)^2 . \tag{2.71}$$

The Hamiltonian then has to be complemented with the following expression:

$$[H_{\text{con}}]^{j\nu}_{i\mu} = -\boldsymbol{\sigma} \cdot \boldsymbol{B}_{\text{con}}(\boldsymbol{m}_i^d) \cdot \delta_{ij}\delta_{\mu\nu} , \tag{2.72}$$

where $\boldsymbol{B}_{\text{con}}$ is a magnetic field, which is perpendicular to $\boldsymbol{m}_i^d$ (figure 2.11) pointing in the direction of the chosen angle. The strength of $\boldsymbol{B}_{\text{con}}$ depends on the difference between the chosen angle Θ_i and the current angle $\arccos\left(\frac{[\boldsymbol{m}_i^d]_z}{|\boldsymbol{m}_i^d|}\right)$:

$$\boldsymbol{B}_{\text{con}} = -\frac{2\lambda \cdot \left(\arccos\left(\frac{[\boldsymbol{m}_i^d]_z}{|\boldsymbol{m}_i^d|} \right) - \Theta_i \right)}{\left(1 - \left(\frac{[\boldsymbol{m}_i^d]_z}{|\boldsymbol{m}_i^d|} \right)^2 \right)^{1/2} \cdot |m_i^d|^3} \cdot \begin{pmatrix} [\boldsymbol{m}_i^d]_z \cdot [\boldsymbol{m}_i^d]_x \\ [\boldsymbol{m}_i^d]_z \cdot [\boldsymbol{m}_i^d]_y \\ -[\boldsymbol{m}_i^d]_x^2 - [\boldsymbol{m}_i^d]_y^2 \end{pmatrix} . \tag{2.73}$$

The constraint does not fix the absolute value of the magnetic moments due to the fact that $\boldsymbol{B}_{\text{con}}$ is perpendicular to $\boldsymbol{m}_i^d$.

2.8.2 Spin-spirals

Some materials like Cr/W(110) do not show a ferromagnetic ground state, but rather a spin-spiral structure [32]. A spin-spiral is a periodic magnetic structure, in which the directions of the spins are determined as follows:

$$\boldsymbol{S}_i = \begin{pmatrix} \cos(\boldsymbol{q} \cdot \boldsymbol{R}_n) \cdot \sin\Theta \\ \sin(\boldsymbol{q} \cdot \boldsymbol{R}_n) \cdot \sin\Theta \\ \cos\Theta \end{pmatrix} . \tag{2.74}$$

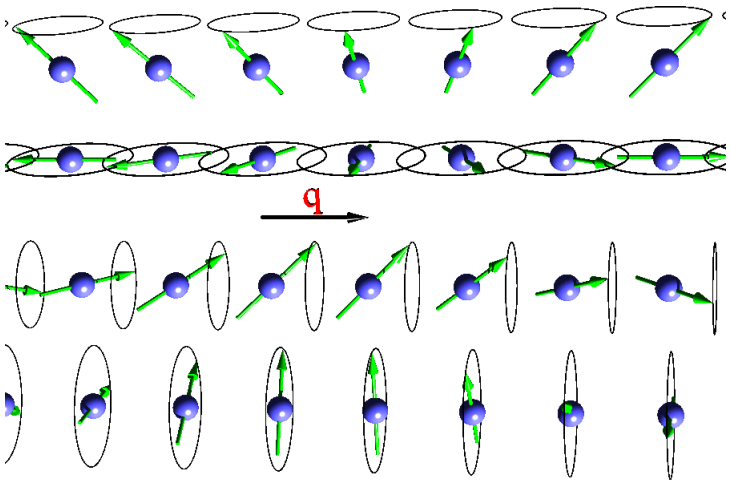


Figure 2.12: Four examples of spin-spirals with spin rotation axis perpendicular (upper two) and parallel (lower two) to the spin-spiral vector $\boldsymbol{q}$. For each case two spirals with cone-angles of $\Theta = \frac{\pi}{2}$ and $\Theta = \frac{\pi}{4}$ are shown.

The cone-angle Θ is the angle between the spins and the rotation axis of the spin-spiral. In equation 2.74 the rotation axis is defined along the z-axis. The rotation angle of the spin-spiral is defined via the spiral vector $\boldsymbol{q}$, which also defines the direction along which the spins are rotated. The rotation angle of the spin at position $\boldsymbol{R}_n$ is defined as $\phi_n = \boldsymbol{q} \cdot \boldsymbol{R}_n$. In figure 2.12 different spin-spirals with cone-angles Θ and spiral-vectors $\boldsymbol{q}$ are displayed.

There are two approaches how to treat spin-spiral systems computationally. One possibility is to describe spin-spirals with rational $|\boldsymbol{q}|$-values in a super-cell using the transformation 2.68 for H_{mag} (see picture 2.13a). A big disadvantage of this method is the huge amount of computational time needed in particular for small q-values. However, within this scheme the implementation of SOC can be done in a straightforward way (see section 2.6).

Another possibility to describe spin-spirals in periodic systems is via the generalized Bloch theorem (see figure 2.13b , [9, 10, 11]). The Bloch theorem according to eq. 2.9 is not any more valid for a spin-spiral due to the non-periodicity of the potential. But if one combines a lattice-translation with the corresponding spin-rotation, each unit-cell is equivalent. Therefore a generalized Bloch theorem remains valid and can be expressed as:

$$\mathcal{U}(\boldsymbol{q} \cdot \boldsymbol{R}_n) \cdot \mathcal{T}(\boldsymbol{R}_n) \cdot \begin{pmatrix} \Phi^{\uparrow}(\boldsymbol{k}, \boldsymbol{r}) \\ \Phi^{\downarrow}(\boldsymbol{k}, \boldsymbol{r}) \end{pmatrix} = \exp(\mathrm{i}\boldsymbol{k} \cdot \boldsymbol{r}) \cdot \begin{pmatrix} \exp(-\frac{\mathrm{i}}{2}\boldsymbol{q} \cdot \boldsymbol{R}_n) \cdot \Phi^{\uparrow}(\boldsymbol{k}, \boldsymbol{r}) \\ \exp(\frac{\mathrm{i}}{2}\boldsymbol{q} \cdot \boldsymbol{R}_n) \cdot \Phi^{\downarrow}(\boldsymbol{k}, \boldsymbol{r}) \end{pmatrix}, \quad (2.75)$$

where $\mathcal{T}(\boldsymbol{R}_n)$ is a lattice-translation operator, $\mathcal{U}(\boldsymbol{q} \cdot \boldsymbol{R}_n)$ is a spin-rotation matrix with a rotation angle $\phi_n = \boldsymbol{q} \cdot \boldsymbol{R}_n$ and the z-axis as rotation axis, and $\Phi^{\sigma}(\boldsymbol{k}, \boldsymbol{r})\,(\sigma = \uparrow, \downarrow)$ is the σ-component of the Bloch-wave $\Phi(\boldsymbol{k}, \boldsymbol{r})$.

As in eq. 2.8 one can construct Bloch-waves, which satisfy the generalized Bloch

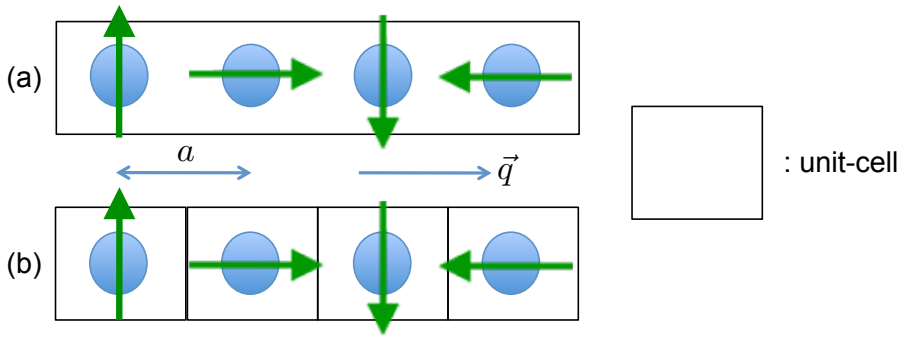


Figure 2.13: Different possibilities of treating spin-spirals: (a) large super-cell for commensurate q-values, (b) generalized Bloch theorem.

theorem:

$$|\Phi^{\uparrow}_{i\mu}(\boldsymbol{k})\rangle = \frac{1}{\sqrt{N}} \cdot \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \cdot |\boldsymbol{n}, i, \mu\rangle \cdot \begin{pmatrix} \mathrm{e}^{-\frac{\mathrm{i}}{2}\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \\ 0 \end{pmatrix} \tag{2.76}$$

$$|\Phi^{\downarrow}_{i\mu}(\boldsymbol{k})\rangle = \frac{1}{\sqrt{N}} \cdot \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \cdot |\boldsymbol{n}, i, \mu\rangle \cdot \begin{pmatrix} 0 \\ \mathrm{e}^{\frac{\mathrm{i}}{2}\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \end{pmatrix} . \tag{2.77}$$

In representation of these Bloch-waves one obtains the following Hamiltonian $H^{j\nu}_{i\mu}(\boldsymbol{k})$:

$$\langle \Phi^{\sigma}_{i\mu} | \mathcal{H} | \Phi^{\sigma'}_{j\nu} \rangle = [H^{j\nu}_{i\mu}(\boldsymbol{k}, \boldsymbol{q})]^{\sigma\sigma'} = \sum_n \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \cdot [s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n)]^{\sigma\sigma'} \cdot H^{\boldsymbol{n}j\nu}_{\boldsymbol{0}i\mu} , \tag{2.78}$$

where $[s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n)]^{\sigma\sigma'}$ are phase-factors for the $\sigma\sigma'$-components of the Hamiltonian:

$$(s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\uparrow} = \mathrm{e}^{-\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \tag{2.79}$$

$$(s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\downarrow\downarrow} = \mathrm{e}^{\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \tag{2.80}$$

$$(s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\downarrow} = 0 \tag{2.81}$$

In the appendix in A.2 one can find a detailed derivation of these expressions.

Important to notice is that the Hamiltonian $[H^{j\nu}_{i\mu}(\boldsymbol{k}, \boldsymbol{q})]^{\sigma\sigma'}$ of eq. 2.78 is expressed in representation of the global spin-frame. In my diploma thesis [16] I used the description in the local frame. In the local spin-frame the phase-factors $[s^j_i(\boldsymbol{q}\cdot\boldsymbol{R}_n)]^{\sigma\sigma'}$ are a bit more complicated than in the global frame (see eqs. 2.80-2.83 in [16]), whereas the magnetic Hamiltonian-part is of a simpler structure in the local frame.

The total energy of a spin-spiral system depends on the cone-angle Θ and the spiral-vector $\boldsymbol{q}$. A very insightful quantity is the magnon dispersion $E(\boldsymbol{q})$ of a spin-spiral with

Figure 2.14: Change in the Fermi energy after adding 1st-order SOC contribution. ε_F is the Fermi energy with SOC, whereas ε_F^0 is the Fermi energy without SOC.

fixed cone-angle. This curve $E(\boldsymbol{q})$ can be used to determine the Heisenberg exchange-coupling parameters J_{ij} or the Dzyaloshinskii-Moriya vectors $\boldsymbol{D}_{ij}$ (see section 2.10, equations 2.100 and 2.102). For DMI SOC is necessary, which brings up the question how to implement SOC in the framework of the generalized Bloch theorem. SOC breaks symmetries by introducing a preferred direction for the magnetization, therefore in its presence the generalized Bloch theorem is not valid. Within the generalized Bloch theorem each unit cell, independent of the direction of the magnetic moments, is equivalent. Therefore we add an additional energy-contribution induced by SOC after having calculated the spin-spiral energies without SOC with the generalized Bloch theorem. This is possible if one treats SOC in a perturbation-theoretical way [12, 13]:

$$\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) = \varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q}) + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})\,, \tag{2.82}$$

where $\varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q})$ is the eigenenergy of the spin-spiral without SOC and

$$\Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) = \langle \boldsymbol{\Psi}_{\boldsymbol{k},n}|\, H_{\mathrm{SOC}}\, |\boldsymbol{\Psi}_{\boldsymbol{k},n}\rangle \tag{2.83}$$

the 1st-order contribution due to SOC.

There is a possibility to calculate the new Fermi energy ε_F after adding $\Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})$ to the spin-spiral eigenenergies or to use the old Fermi energy ε_F^0 of the spin-spiral system without SOC. In general the Fermi energies can shift considerable as indicated in the figure 2.14. *Ab-initio* calculations for Fe/W-systems exhibit, that the new recalculated Fermi energy is nearly the same as the old Fermi energy [12, 13]. In Fe/Pt-systems, investigated in [16], there is a non-negligible difference between the old or new Fermi energy. Nevertheless for the determination of the DM-constant D both methods predict the same.

For the more detailed discussion of SOC it is intuitive, to take a look at the total 1st-order SOC contribution to the total energy using the new or old Fermi energy:

(a) unchanged Fermi energy:

$$\begin{aligned}\Delta E_{SOC}(\boldsymbol{q}) &= \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot (\varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q}) + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})) - \\ &\quad \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot \varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q}) \\ &= \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) \end{aligned} \tag{2.84}$$

It can be proven [12] that

$$\Delta\varepsilon_{\boldsymbol{k},n}(-\boldsymbol{q}) = -\Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})\,. \tag{2.85}$$

On the other hand, for the eigenenergies of the spin-spiral system without SOC we have

$$\varepsilon^0_{\boldsymbol{k},n}(-\boldsymbol{q}) = \varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q}) \tag{2.86}$$

due to symmetry. Therefore the SOC-contribution fulfils the following relation:

$$\Delta E_{\mathrm{SOC}}(-\boldsymbol{q}) = -\Delta E_{\mathrm{SOC}}(\boldsymbol{q})\,. \tag{2.87}$$

It is also obvious that $\Delta E_{\mathrm{SOC}} \propto \xi_{\mathrm{SOC}}$, which can be seen in expression 2.83.

(b) recalculated Fermi energy:

$$\begin{aligned}\Delta E_{\mathrm{SOC}}(\boldsymbol{q}) &= \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n} + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}), \varepsilon_F) \cdot (\varepsilon^0_{\boldsymbol{k},n} + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})) - \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot \varepsilon^0_{\boldsymbol{k},n} \\ &= \sum_{\boldsymbol{k},n} [f(\varepsilon^0_{\boldsymbol{k},n} + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}), \varepsilon_F) - f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F)] \cdot \varepsilon^0_{\boldsymbol{k},n} + \\ &\quad \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n} + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}), \varepsilon_F) \cdot \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) \\ &\approx \sum_{\boldsymbol{k},n} [f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) + (\varepsilon^0_{\boldsymbol{k},n}(\boldsymbol{q}) + \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q})) \cdot \\ &\quad (\frac{\partial f}{\partial \varepsilon^0_{\boldsymbol{k},n}} \cdot \Delta\varepsilon_{\boldsymbol{k},n}(\boldsymbol{q}) + \frac{\partial f}{\partial \varepsilon^0_F} \cdot (\varepsilon_F - \varepsilon^0_F))] \end{aligned} \tag{2.88}$$

It is obvious that the relation 2.87 is not any more valid. In the case of $\varepsilon_F \approx \varepsilon^0_F$ it is fulfilled to a good approximation and we can use the old Fermi energy without causing a significant error.

We should remark that treating SOC for spin-spirals in 1st-order perturbation theory does not allow to calculate the magneto-crystalline anisotropy energy (MCA) of the system. To treat also the MCA, one has to extend the mechanism to 2nd-order perturbation theory [12, 13], which is not implemented in our code yet. Instead collinear SOC calculations are used to determine the MCA.

2.9 Force theorem

The self-consistent scheme of the tight-binding method (see figure 2.9) can require a lot of computational time for systems with many basis atoms. Especially for magnon dispersions, calculations for a large set of $\boldsymbol{q}$-values have to be performed. Fortunately one can spare a lot of computing time using the so-called force theorem [37, 38, 39].

The force theorem can be used in systems, where a small perturbation $\delta\mathcal{H}$ is added to a system $\mathcal{H}_0$, whose self-consistent solution is known:

$$\mathcal{H} = \mathcal{H}_0 + \delta\mathcal{H} \qquad \text{with} \qquad \underline{\boldsymbol{H_0}} \cdot \boldsymbol{\Psi}^0_{\boldsymbol{k},n} = \varepsilon^0_{\boldsymbol{k},n} \cdot \boldsymbol{\Psi}^0_{\boldsymbol{k},n} \,. \tag{2.89}$$

For the perturbed system only one iteration is done, which is also known as "one-shot-calculation". Of course this one-shot-calculation depends delicately on the starting values for the magnetic moments and Mulliken charges, therefore reasonable starting values are necessary. The charges and moments of the converged unperturbed problem should provide good starting values if the perturbation $\delta\mathcal{H}$ is small.

The force theorem can be expressed by the following relation:

$$\begin{aligned} E_{\text{tot}} - E^0_{\text{tot}} &= E_{\text{band}} - E^0_{\text{band}} \\ &= \sum_{\boldsymbol{k},n} f(\varepsilon_{\boldsymbol{k},n}, \varepsilon_F) \cdot \varepsilon_{\boldsymbol{k},n} - \sum_{\boldsymbol{k},n} f(\varepsilon^0_{\boldsymbol{k},n}, \varepsilon^0_F) \cdot \varepsilon^0_{\boldsymbol{k},n} \,, \end{aligned} \tag{2.90}$$

where E_{band} is the band energy of the perturbed system, E^0_{band} is the band energy of the unperturbed system and E_{tot} is the total energy of the perturbed system for a self-consistent calculation (including double counting terms). Within the force theorem the difference of the single particle energies is taken as approximation for the difference of the total energies.

In the treatment of spin-spirals a self-consistent calculation for $\boldsymbol{q} = 0$ (ferromagnetic case) is performed and the converged charges and moments are used as starting-values for a one-shot-calculation for $\boldsymbol{q} \neq 0$.

To ensure that the force theorem is a proper approximation, in this case the spin-spiral for $\boldsymbol{q} \neq 0$ has to be only slightly different in comparison to the $\boldsymbol{q} = 0$-case. This is definitely true for small cone angles Θ and small q-values.

2.10 Extended Heisenberg model

It is common to use a model-Hamiltonian to describe the magnetic interactions in a (periodic) system. In the following sections the extended Heisenberg model is described in detail [40].

In the extended Heisenberg model the following assumptions are taken into account:

- The magnetic moment of each atom is described by a localized (classical) spin-vector $\boldsymbol{S}_i$ at the corresponding lattice position.

- The magnetic moment has a fixed absolute value, i.e. the absolute value of the magnetic moment does not change with the q-value, which is an approximation for larger cone-angles Θ in the spin-spiral case.

The extended Heisenberg model can be written as generalized scalar-product of spins:

$$\mathcal{H}_{\text{ex.Heisenberg}} = -\sum_{i,j} \boldsymbol{S}_i^T \cdot \underline{\boldsymbol{V}}_{ij} \cdot \boldsymbol{S}_j \,. \tag{2.91}$$

Here, $\underline{\boldsymbol{V}}_{ij}$ is a 3×3 matrix, which can be divided into a symmetric and an antisymmetric part, $\underline{\boldsymbol{V}}_{ij}^S$ and $\underline{\boldsymbol{V}}_{ij}^A$:

$$\underline{\boldsymbol{V}}_{ij} = \frac{1}{2}(\underline{\boldsymbol{V}}_{ij} + \underline{\boldsymbol{V}}_{ij}^T) + \frac{1}{2}(\underline{\boldsymbol{V}}_{ij} - \underline{\boldsymbol{V}}_{ij}^T) = \underline{\boldsymbol{V}}_{ij}^S + \underline{\boldsymbol{V}}_{ij}^A \,. \tag{2.92}$$

It is common to divide the symmetric matrix $\underline{\boldsymbol{V}}_{ij}^S$ into a traceless part and an isotropic exchange part:

$$\underline{\boldsymbol{V}}_{ij}^S = [\underline{\boldsymbol{V}}_{ij}^S - J_{ij} \cdot \underline{\boldsymbol{I}}] + J_{ij} \cdot \underline{\boldsymbol{I}} \,, \tag{2.93}$$

with

$$J_{ij} = \frac{1}{3} \cdot tr[\underline{\boldsymbol{V}}_{ij}^S] \,. \tag{2.94}$$

The antisymmetric part can be expressed using a cross-product:

$$\boldsymbol{S}_i^T \cdot \underline{\boldsymbol{V}}_{ij}^A \cdot \boldsymbol{S}_j = \boldsymbol{D}_{ij} \cdot (\boldsymbol{S}_i \times \boldsymbol{S}_j) \,, \tag{2.95}$$

where

$$[\underline{\boldsymbol{V}}_{ij}^A]_{nn'} = \sum_l [\boldsymbol{D}_{ij}]_l \cdot \epsilon_{lnn'} \,. \tag{2.96}$$

Here, $\epsilon_{lnn'}$ is the Levi-Civita tensor.

Summarizing the extended Heisenberg model consist of the following parts:

$$\begin{aligned} \mathcal{H}_{\text{ex.Heisenberg}} \;=\; -\sum_{i,j}[& \underbrace{J_{ij} \cdot \boldsymbol{S}_i^T \cdot \boldsymbol{S}_j}_{\text{symmetric isotropic exchange}} + \underbrace{\boldsymbol{S}_i^T \cdot (\underline{\boldsymbol{V}}_{ij}^S - J_{ij} \cdot \underline{\boldsymbol{I}}) \cdot \boldsymbol{S}_j}_{\text{symmetric anisotropic exchange}} + \\ & \underbrace{\boldsymbol{D}_{ij} \cdot (\boldsymbol{S}_i \times \boldsymbol{S}_j)}_{\text{Dzyaloshinskii}-\text{Moriya interaction}}] \,. \end{aligned} \tag{2.97}$$

In many cases the symmetric anisotropic exchange part is neglected except for the diagonal part

$$\boldsymbol{S}_i^T \cdot (\underline{\boldsymbol{V}}_{ij}^S - J_{ij} \cdot \underline{\boldsymbol{I}}) \cdot \boldsymbol{S}_j \approx \boldsymbol{S}_i^T \cdot \underline{\boldsymbol{K}}_i \cdot \boldsymbol{S}_i \cdot \delta_{ij} \,, \tag{2.98}$$

which describes the magneto-crystalline anisotropy (MCA) of the system with the MCA-matrix $\underline{\boldsymbol{K}}_i$.

2.10.1 Symmetric isotropic exchange

This part is the well-known Heisenberg exchange, which describes a ferromagnetic ($J_{ij} > 0$) or anti-ferromagnetic ($J_{ij} < 0$) coupling between the spins $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$. If one considers only next-neighbour coupling a collinear alignment of the spins would be the ground-state solution. In general, it can be proven that a solution of the Heisenberg Hamiltonian in a periodic system is a spin-spiral. For a spin-spiral with a cone-angle Θ, spiral-vector $\boldsymbol{q}$ and the z-axis as the rotation axis, the Heisenberg model can be rewritten in the following way using

$$\boldsymbol{S}_i = \begin{pmatrix} \cos(\boldsymbol{q}\cdot\boldsymbol{R}_n)\cdot\sin\Theta \\ \sin(\boldsymbol{q}\cdot\boldsymbol{R}_n)\cdot\sin\Theta \\ \cos\Theta \end{pmatrix} \quad :$$

$$-\sum_m \sum_{n\in\mathrm{shell}(m)} J_m \cdot \boldsymbol{S}_0 \cdot \boldsymbol{S}_n = -\sum_m \sum_{n\in\mathrm{shell}(m)} J_m \cdot \left[\cos^2\Theta + \sin^2\Theta\cdot\cos(\boldsymbol{q}\cdot\boldsymbol{R}_n)\right]. \quad (2.99)$$

Here a shell is a set of atoms, which have the same fixed distance from the reference atom. For example shell(1) includes all nearest neighbours of the reference atom. If one calculates the magnon dispersion $E(\boldsymbol{q})$, the Heisenberg exchange-coupling parameters J_n can be determined via the following equation:

$$\frac{1}{\sin^2\Theta}\cdot\left[E(\boldsymbol{q}) - E(0)\right] = -\sum_m \sum_{n\in\mathrm{shell}(m)} J_m \cdot \left[\cos(\boldsymbol{q}\cdot\boldsymbol{R}_n) - 1\right]. \quad (2.100)$$

The coupling parameters can be calculated via a fitting procedure to quantum-mechanical results of the energy $E(\boldsymbol{q})$.

An important quantity is the so-called spin-stiffness constant A, which describes the pre-factor of the quadratic q-dependence in the magnon dispersion for very long-wavelength spin-spirals, i.e. very small q-values. This quadratic q-dependence can be easily seen from eq. 2.100 due to $(1-\cos(\boldsymbol{q}\cdot\boldsymbol{R}_n)) \approx (\boldsymbol{q}\cdot\boldsymbol{R}^n)^2$ for small q.

2.10.2 Dzyaloshinskii-Moriya interaction

The antisymmetric exchange part in the extended Heisenberg model is also known as Dzyaloshinskii-Moriya interaction (DMI) [28, 29]. In contrast to the symmetric exchange part the energies of the antisymmetric exchange part are in general different for different rotational senses of the spin-spirals. This can be easily seen by rewriting the antisymmetric part of the Hamiltonian as follows:

$$\begin{aligned} E_{\mathrm{DM}} &= -\sum_m \sum_{n\in\mathrm{shell}(m)} \boldsymbol{D}_m \cdot (\boldsymbol{S}_0 \times \boldsymbol{S}_n) \\ &= -\sum_m \sum_{n\in\mathrm{shell}(m)} \boldsymbol{D}_m \cdot \begin{pmatrix} -\sin\Theta\cdot\cos\Theta\cdot\sin(\boldsymbol{q}\cdot\boldsymbol{R}_n) \\ \sin\Theta\cdot\cos\Theta\cdot(\cos(\boldsymbol{q}\cdot\boldsymbol{R}_n)-1) \\ \sin^2\Theta\cdot\sin(\boldsymbol{q}\cdot\boldsymbol{R}_n) \end{pmatrix}. \end{aligned} \quad (2.101)$$

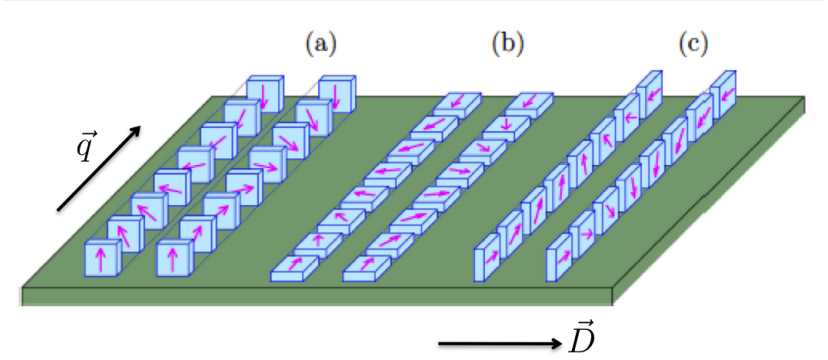


Figure 2.15: Different spin-spirals on an isotropic surface. Each case (a), (b) and (c) is explained in the text. | modified fig. from [33]

In the special case of flat-spirals ($\Theta = \frac{\pi}{2}$), the equation simplifies to

$$E_{\mathrm{DM}}(\boldsymbol{q}) = -\sum_{m} \sum_{n \in \mathrm{shell}(m)} (\boldsymbol{D}_m)_z \cdot \sin(\boldsymbol{q} \cdot \boldsymbol{R}_n)\,. \tag{2.102}$$

From this expression it becomes clear that $E_{\mathrm{DM}}(-\boldsymbol{q}) = -E_{\mathrm{DM}}(\boldsymbol{q})$. Of course the expression for the DMI would be of an analogous form for flat-spirals in the x-z or y-z plane.

In many cases only the first neighbour shell is considered for the DMI, therefore the DM-constant $D := D_1$ is determined via a linear fit to the magnon dispersion for small q-values for which $\sin(\boldsymbol{q} \cdot \boldsymbol{R}_n) \approx \boldsymbol{q} \cdot \boldsymbol{R}^n$ is valid.

Crucial for the appearance of the DMI is the presence of SOC and a non-inversion-symmetric environment. Both requirements are needed, otherwise, the energy of a spin-spiral is independent of the rotational sense due to symmetry arguments, i.e. $E_{\mathrm{DM}} = 0$. As a deep understanding of the symmetry arguments is crucial for further analyse, we provide a more detailed explanation.

If SOC is not considered, real-space and spin-space would be independent and a global spin-transformation would not change the energy of the system. Now there exists a mirror-transformation applied to a flat-spiral leads to the reversion of its rotation sense. This mirror-plane is the plane, which is spanned by the spiral vector $\boldsymbol{q}$ and the rotation axis of the spin-spiral. Since a global spin-transformation does not change the energy in the system without SOC, it holds $E_{\mathrm{DM}}(-\boldsymbol{q}) = E_{\mathrm{DM}}(\boldsymbol{q}) = -E_{\mathrm{DM}}(\boldsymbol{q}) = 0$.

The argument for a non-inversion-symmetric environment as requirement for non-vanishing DMI is a little bit more complicated to understand, because the type of spin-spiral plays also an important role. Figure 2.15 should help with the explanation. Here, flat spin-spirals on a surface are displayed. In each case (a), (b) and (c) two spin-spirals with equal $|\boldsymbol{q}|$ but opposite rotational sense are shown. The only difference among (a), (b) and (c) is the rotation axis. In the cases (a) and (b) there is a preserved mirror-plane, which leads to vanishing DMI. In the case (c) there is no mirror-plane due

to the symmetry-breaking effect of the surface, which leads to a non-vanishing DMI-contribution. These statements correspond to the case of an isotropic surface. For more details about the symmetry of the DMI for more general systems the reader is referred to reference [41].

2.10.3 Symmetric anisotropic exchange

As already mentioned in the extended Heisenberg model only the diagonal part of the symmetric anisotropic exchange is considered, which can be identified as MCA. The MCA is important to investigate the occurence of spin-spirals due to DMI, because eventually the MCA can inhibit the development of a spin-spiral. For example if the system in figure 2.15(c) has an easy axis perpendicular to the plane of the spin-spiral and the MCA is strong enough, the system could favour a collinear alignment along the easy axis over the spin-spiral structure even in the presence of DMI.

The stability of spin-spirals can be most easily investigated with a micromagnetic model [42, 43]. Without going into details, we note that relations between the spin-stiffness constant A, the DM-constant D and the MCA give a stability condition for the development of a spin-spiral in the system. The micromagnetic model is based on a continuum-theoretical approach for long-wavelength spin-spirals, which leads to the following equation for the energy E of the spin-spiral in dependence of the spin-spiral period length λ:

$$E(\lambda) = A \cdot \lambda^{-2} + D \cdot \lambda^{-1} + \bar{K} \,. \tag{2.103}$$

The derivation of the spin-stiffness constant A and the DM-constant D with the help of the magnon dispersion $E(\boldsymbol{q})$, is explained in section 2.10.1 and 2.10.2. The connection to eq. 2.103 can be easily seen by using $|\boldsymbol{q}| \propto \lambda^{-1}$ in eq. 2.100 and 2.102 for small q-values. The constant $\bar{K}$ describes the average MCA over an entire period of the spin-spiral.

3 Description of the `inputcard`

In this chapter the `inputcard` will be explained in detail. I strongly recommend to read this chapter, if you are planning to use the JuTiBi code. The `inputcard` is divided into thematic sub-parts, f. ex. the lattice-structure, magnetic properties etc. First I give a general explanation how to use the `inputcard` properly without causing errors in subroutines responsible for reading in the data (i.e. Ioinput and some "almost-copies" of these subroutine). Then I will discuss the `inputcard` in detail. Note that a list of all appearing keywords of the `inputcard` is presented in the appendix D.

3.1 Basic description of using the `inputcard`

The `inputcard` consists of several keywords (these should not be changed), which are all written in capital letters. A lot of comments should simplify the use of the `inputcard`. In this documentation the comments are removed in the quoted parts of the `inputcard` to restrict the focus on the keywords. The complete `inputcard` (including all comments) for the example of a self-consistent bcc-Fe calculation can be seen in the appendix C. First of all the `inputcard` is not written in a fixed format, this means parts can be shifted and one has not to take (too much) care in using the correct amount of indents behind a keyword to ensure a correct import of the input values into the code. But of course there are rules, which have to be followed:

- Do not change the keywords! This includes also the cases, where a "=" is set right behind a keyword. Leave it there!

- Do not change the position of the values in respect to the position of the keywords. This does not include any indents, but a value which stands in the same line than the keyword should not be set into the next line under the keyword. Values in a line under the keyword should not be written into another line in respect to the position of the keyword. More informations can be found in the description of the subroutine Ioinput. I would recommend to take a short look into the examples.

- Values, which should be read in from the `inputcard` should be located within the first 80 columns of the `inputcard`. The routine Ioinput will not read in the data further than the 80th column.

- If the code is behaving unexpected, there is possibly a mistake in the `inputcard`. The most common mistake is to set logical variables into wrong relations to each other. Sometimes the code would skip whole essential parts. For example if the user

wants to carry out a non-magnetic calculation (`spinlog=false`), but the logical `blo_wave` remains `true` the execution of the code could lead to some unexpected results. The code does not check all cross-relations between the logical variables, except of the most important ones, and as a consequence would give no error message.

3.2 `inputcard`: lattice structure

Let us take a look into the first part of the `inputcard`:

```
-------------------------------------------------------------------------
*** Structure Properties - Lattice ***

ALATBASIS=  2.87d0 1.0d0 1.0d0


BRAVAIS
-0.5d0 0.5d0 0.5d0
0.5d0 -0.5d0 0.5d0
0.5d0 0.5d0 -0.5d0

-------------------------------------------------------------------------
```

This part of the `inputcard` is necessary to define the Bravais lattice. Behind the keyword `ALATBASIS` the code demands the first lattice parameter a and the ratios of the lattice parameters $\frac{b}{a}$ and $\frac{c}{a}$. I recommend to give the lattice parameter a in units of Å. The three lines under the keyword `BRAVAIS` are the three Bravais vectors B_1, B_2 and B_3 in units of a, b and c. Therefore the Bravais vectors in the `inputcard` are read in row-wise![1] The above example of the `inputcard` is a bcc-lattice (in this case Fe with a lattice parameter of $a = 2.87\,\text{Å}$).

Note that the lattice parameters a, b and c will act separately on the x-, y- and z-components of the denoted Bravais vectors and not separately on the denoted Bravais vectors B_1, B_2 and B_3 themselves. To be more precise this means if you have a denoted first Bravais vector $(1, 1, 1)$ and lattice parameters a, b and c the code will convert it to the Bravais vector (a, b, c) and not to (a, a, a). In the case of cubic or tetragonal systems both methods would be the same of course.

[1] In difference to the array `abravais` in the code, where the Bravais vectors are stored column-wise.

If 1-dim. (or 2-dim. systems) are desired the third (and second) Bravais vector should be switched to $(0, 0, 0)$. Very important to mention is also that in these cases the surfaces have to be located in the $x-y$-plane and a chain should be located along the x-axis. Therefore the following structure should be used:

```
BRAVAIS                          BRAVAIS
1.0d0 0.0d0 0.0d0                 B1x    B1y 0.0d0
0.0d0 0.0d0 0.0d0       or        B2x    B2y 0.0d0
0.0d0 0.0d0 0.0d0                0.0d0 0.0d0 0.0d0
```

where `B1x`, `B1y` is the first 2-dimensional Bravais vector and `B2x`, `B2y` the second. Note that in the 1- and 2-dim. case the relation between the lattice parameters $\frac{b}{a}$ and $\frac{c}{a}$ should not be set to zero, but to some other arbitrary value! Otherwise the code will divide through zero in the k-mesh creation.

To complete the lattice structure the positions of the basis atoms have to be set.

```
-------------------------------------------------------------------
*** Basis atoms in unit cell ***

CARTESIAN= T

BASATOMS
  0.d0  0.d0    0.d0
  0.5d0 0.5d0   0.5d0

NUMBASIS=2

-------------------------------------------------------------------
```

The keyword `NUMBASIS` is used to read in the number of basis atoms. To enter the positions of the basis atoms, the user should insert them row-wise under the keyword `BASATOMS`. The code reads in all `numbasis` lines under the keyword, but not further. Therefore additional following lines are no problem. The vectors of the basis atoms should be given in units of a, b and c as the Bravais vectors.

If the user wants to enter the basis atoms in cartesian representation, he should set a `T` (for true) behind the keyword `CARTESIAN`, whereas a `F` (for false) should be used for a Bravais representation. Note that in the case of a 2- (or 1-dim.) system the third (and second) component of the basis atoms are always interpreted as given in cartesian representation.

3.3 `inputcard`: k-mesh properties

The following part of the `inputcard` is used to determine the k-mesh properties.

```
-----------------------------------------------------------------------
*** k-Mesh Properties ***

BZDIVIDE= 20 20 20
KPOIBZ= 8000
IrrBZ= T
HARD_DISK=F

-----------------------------------------------------------------------
```

In the section 6.2 within the description of the subroutine bzirr3d, the creation of the k-mesh within the code is briefly explained. To define the mesh the code needs to know the partitioning along the reciprocal Bravais vector directions, which has to be set behind the keyword `BZDIVIDE`. The first value corresponds to the partition along the first reciprocal Bravais vector, the second to the second and the third to the third.

Take care that the partition approximately fits to the (inverse) relations of the lattice constants, otherwise the k-mesh is not properly chosen. I also would recommend to use only even values, because in this case the k-mesh creation works best.

The total number of k-points has to be written behind the keyword `KPOIBZ`, which is the product of the 3 mesh partition values. One can also enter a larger number here, but this would be a waste of memory.

The keyword `IrrBZ` is used to decide, if the full k-mesh should be used in the diagonalization of the Hamiltonian (enter `F`) or if only the (full) irreducible part of the mesh should be used (enter `T`). For calculations with spin-orbit coupling or non-collinear magnetism I strongly recommend to use the full k-mesh, because there is no feature in the code to use only some of the lattice symmetries to obtain a k-mesh suited to the symmetries of the magnetic system.

If the eigenvectors for each k-point (and band index) shall be saved into the unformatted files `eigenvectors.unformatted` and `eigenvectors_mod.unformatted` instead into the memory fill in `T` behind the keyword `HARD_DISK`. For very large system with a lot of k-points one should avoid saving the eigenvectors onto the hard disk, due to a largely increased need of computational time.

3.4 inputcard: Shell generation

```
-----------------------------------------------------------------------
*** Properties of the Cluster Generation ***

MAXCLUST= 10000
RMAXIMAL= 4.5d0
RCUTOFF=  3.5d0
MAXBASIS= 50

-----------------------------------------------------------------------
```

In this part of the `inputcard` the necessary properties for the shell generation are read in. These values are used in the routines rrgen and clsgen99. In principle only the value behind `RCUTOFF` is really important, whereas the other values are not very significant as long as they are not too small.

First the maximum number of Bravais vectors within the created cluster should be assigned via the keyword `MAXCLUST`. For the majority of applications the value should be kept between 1000 and 10000. To be sure use a value of 10000, which should be enough for all applications. If the maximum number of Bravais vectors is not sufficient to create the cluster within the defined sphere, an error message from the subroutine rrgen appears:

```
 Dimension ERROR. Please, change the parameter NRD to         2001
```

A maximum number of basis atoms per unit cell has also to be defined behind the keyword `MAXBASIS`, which is needed for the allocation of some arrays in the subroutine clsgen99. If you want to spare a little bit memory space use the number of basis atoms instead of the default value of 50. Note that a maximal number of atoms within the cluster is calculated by multiplying the number of maximal Bravais vectors inside the cluster with the number of basis atoms. If this number is larger than 10000, the code sets it back to 10000 to prevent too large arrays and it outputs a warning message.

Additionally after defining the sizes of the arrays, the routines need two defined radii to construct the clusters within this spheres. The first radius behind the keyword `RMAXIMAL` is needed to roughly construct a first cluster containing only a set of the lattice vectors of the system (see subroutine rrgen). This radius is in units of the lattice parameter a and is of the magnitude of the radius, which is used for the cluster generation. Usually the used radius of this first cluster, which is calculated intrinsically within the routine rrgen, is much larger than the value behind `RMAXIMAL`. But as mentioned before this value is not important as long as it is not too small, therefore use at least the same value as behind the keyword `RCUTOFF`. The calculation of the shells are very fast and the arrays do not occupy too much memory space (the majority is occupied by the stored eigenvectors), therefore it is not necessary to think about optimized values in this case.

The really important value for the cluster generation is the cutoff-radius, which is written behind `RCUTOFF`. It is given in units of the lattice parameter a and defines the

sphere, inside which the neighbours are calculated. Neighbours lying outside this sphere are not considered. In the case of the NRL-TB parametrization an additional cutoff-parameter is introduced. The code checks automatically if this cutoff radius is smaller than the defined cutoff radius of the shell generation. If this is not the case an error message appears, telling the user to increase the value of `RCUTOFF`.

An additional feature provided by the JuTiBi code is the possibility to cut certain bondings in the crystal. This will not change the symmetries of the lattice, but the corresponding atoms of the removed bonding do not directly interact with each other. This can change the physics of the system drastically and allows a deeper analyse of the importance of some bondings. However, note that this feature is rather untested.

```
-------------------------------------------------------------------------
 *** Cut Bondings *** ! ALPHA STATUS

BONDCUT=F

NUMCUTBON=2
SPECBONDS
 1 0. 0. 1.5
 2 0. 0. -1.5

 -------------------------------------------------------------------------
```

To cut a specific bonding set `BONDCUT` to `T` and enter the type of bondings under the keyword `SPECBONDS`. The bondings are removed from the array `pos_cluster_atom`, which is also written out in the file `shells_neighbours.dat` and therefore this file should be used to obtain the components of the bonding vectors. First specify the cluster atom (i.e. the central basis atom of the cluster) in which the bonding is appearing and then enter the x-, y- and z-component of the bonding in units of the lattice constant a. Keep in mind that bondings should be removed symmetrically, otherwise the Hamiltonian could be not hermitian!

3.5 `inputcard`: Properties of the basis atoms

The next part of the `inputcard` contains specific information about each basis atom. The types of appearing atomic orbitals can be specified and the spin-orbit coupling

parameters ξ_p and ξ_d have to be put here.

```
-----------------------------------------------------------------------
*** Atominfo ***

LMAXIMAL=9
NUMELUC=18

-----------------|--------------------------------------
ZATOM       #ORB |  s_Orb |  px_Orb    py_Orb    pz_Orb |
26.0d0       9   |    1   |    1          1        1    |
-----------------|--------------------------------------|
SOC-Parameter:   |   ---  |           0.18d0            |
-------------------------------------------------------------
dxy_Orb    dxz_Orb    d_yz_Orb    d(x2-y2)_Orb    d(z2-r2)_Orb |
   1          1          1            1                1        |
-------------------------------------------------------------|
                                0.06d0                       |
-------------------------------------------------------------

-----------------|--------------------------------------
ZATOM       #ORB |  s_Orb |  px_Orb    py_Orb    pz_Orb |
78.0d0       9   |    1   |    1          1        1    |
-----------------|--------------------------------------|
SOC-Parameter:   |   ---  |           2.5d0             |
-------------------------------------------------------------
dxy_Orb    dxz_Orb    d_yz_Orb    d(x2-y2)_Orb    d(z2-r2)_Orb |
   1          1          1            1                1        |
-------------------------------------------------------------|
                                0.53d0                       |
-------------------------------------------------------------

-----------------------------------------------------------------------------
```

First the maximum number of appearing orbitals per atom has to be specified behind the keyword `LMAXIMAL`. Usually the TB scheme includes the s-, p- and d-orbitals, which leads to a value of 9 for `LMAXIMAL`. If all basis atoms are only treated within the s- and p orbitals one can reduce this number to 4, but it is not necessary to change the value. I recommend to keep the value always at 9, because it is a well tested case.

The number of electrons per unit cell enters behind the keyword `NUMELUC`.

The following table-like structure describes the atomic number, the type of used (or non-used) orbitals and the SOC-parameters for each basis atom. Therefore if a system consists of 10 basisatoms, one needs to specify 10 of these tables (use copy and paste!). Do not change the structure of these tables and between two of these tables, one line

has to be kept free, otherwise the code stops with an error message within the Ioinput routine.

Now the structure of the table will be explained. The value under the keyword `ZATOM` is the atomic number. This atomic number, unlike in *ab-initio* codes, is completely unimportant in the TB scheme. It can act as classification of the basis atom, to distinguish the different types of basis atoms. But this distinction is only "on paper", because the code distinguishes the atom types by the SKP-parameters, the mulliken charges and exchange energies, which are defined in the `inputcard`.

The next adjustable property are the used (or non-used) atomic orbitals per basis atom. For the NRL-TB scheme all 9 orbitals (s, p and d) should be activated, but if arbitrary Slater-Koster parameters are used, one can switch off orbital types of the basis atoms (see for example pure s- and p-treatment in [44]). To switch orbitals on the specific basis atoms on (or off) set the values under the keywords `s_Orb`, `px_Orb`, `py_Orb`, `pz_Orb`, `dxy_Orb`, `dxz_Orb`, `dyz_Orb`, `d(x2-y2)_Orb`, `d(z2-r2)_Orb` to `1` or `0`.[2] By switching orbitals on or off, one can change the basis representation of the system. Therefore a system with pure d-orbitals has 4 basis functions less per basis atom than a system with all 9 orbitals.

The last adjustable property of the basis atoms in this part of the `inputcard` are the SOC-parameters of the p- and d-orbitals. Put the values under the specification-chart of the p-orbitals and the d-orbitals (in above example: 0.18 eV for p and 0.06 eV for d for the first basis atom). I recommend to enter this parameters in eV. The magnitude of these parameters for several elements can be found in [45] and for Fe and Pt the parameters are presented in the appendix B of this documentation.

[2] In the code only `s_Orb`, `px_Orb` and `dxy_Orb` act as keyword.

3.6 inputcard: Slater-Koster parameters

In the following chapter of the inputcard specifications about the Slater-Koster parametrization (see section 2.3) can be done.

```
-------------------------------------------------------------------------------
*** Input of the Slater-Koster parameters ***


IDENTATOM=T
ONLYNEXTN=F
RNEIGH=3.5d0
SKPVARY=T
OVLAP=T
PAPA_BINA=T
TYP_BASAT
 1   1
 2   2
 3   1

-------------------------------------------------------------------------------
```

The code mainly consists of two types of parametrizations. The first one is the NRL-TB parametrization, which is explained in detail in the section 2.3. The second one is based on a manual definition of the SKPs for each bonding. For users who only want to use the code for calculations, I would strongly recommend to work with the NRL-TB parametrization, because the second parametrization type is a not well-tested case, yet. However it seems to work for very simple systems.

3.6.1 NRL-TB parametrization

This parametrization is less flexible and therefore I recommend to use the following setup. To activate this parametrization enter a T behind the keyword SKPVARY. Then choose ONLYNEXTN as F, because choosing it as T limits the hopping parameters to next-neighbour hoppings only. Behind the keyword RNEIGH the code demands the distance in units of the lattice parameter a in which the hopping parameters will be considered as non-zero. In the case of the NRL-TB parametrization this value should be at least as large as the cutoff radius in the parametrization. The cutoff radii of the parameter sets can be found on the webpage [4]. The value behind RCUTOFF of the shell generation (see 3.4) should be at least as large as the value RNEIGH. Otherwise the size of the created cluster is not sufficient to support all hopping parameters! Activate the treatment of the overlap matrix by setting a T behind the keyword OVLAP. If a system consisting of two different atom-types should be treated, then switch PAPA_BINA to T and specify the atom type for each basis atom in the lines under the keyword TYP_BASAT. In above example the first and third basis atom (for the positions see keyword BASATOMS in 3.2) are of the first

atom-type, whereas the second basis atom is of the second type. Now the question arises, where the NRL-TB parameter sets are read in? To have a clear arranged `inputcard` the parameter sets with their parameters $a_{ll'm}$, $b_{ll'm}$, $c_{ll'm}$, etc. (see 2.3) are imported from another file with the name `input_SKP_papakonst`. In this file the parameter sets of the specific element should be copied (unchanged!) from the webpage [4]. To distinguish between the different atom-types the code needs two additional keywords, `STA_READ` for the first atom-type and `STA2_READ` for the second atom-type, which should be set directly above the parameter set. Therefore it should look as follows:

```
STA_READ
NN00000                              (Old style Overlap Parameters - Not polarized)
Iron (Fe) -- Spin restricted -- 18 Jan 00 -- long range -- t2g=eg
1                                    (One atom type in this file)
16.5   0.5                           (RCUT and SCREENL for 1-1 interactions)
9                                    (Orbitals for atom 1)
55.85                                (Atomic Weight of Atom 1)
 2.0  0.0  6.0                       (formal spd valence occupancy for atom 1)
   .167876018007E+01    0   1    Fe-para-001 lambda
   .111237776825E+00    0   2    Fe-para-002 a_s
   .157472037798E+03    0   3    Fe-para-003 b_s
   .279493598875E+06    0   4    Fe-para-004 c_s
  -.886322891071E+08    0   5    Fe-para-005 d_s
   .512530808853E+00    0   6    Fe-para-006 a_p
   ..... etc.
```

Some additional notes concerning the NRL-TB parametrization:

- The keyword `IDENTATOM` is insignificant, because it is not used within the NRL-TB parametrization, and the keyword `SKPSPIN` of 3.7 has to be `F`. The NRL-TB parametrization implemented in this code has to be the spin-independent one.
- The NRL-TB parametrization for binary systems (i.e. systems consisting of two atom-types) is not using the binary parameters of the page [4], because unfortunately there is only a small amount of binary parameter sets. Therefore a simple ansatz is used (see eq. 2.25) to model the hopping parameters between the different elements. The used parameter sets are the sets for the pure systems (f.ex. an Fe- and a Pt- set to describe Fe/Pt systems). More details about the quality of this description can be found in [16].
- Using the NRL-TB parametrization the structures can be chosen within a wide variety, but too small inter-atomic distances lead to unphysical descriptions of the SKPs. An error message of the type

  ```
  Problems with Diagonalization (S pos. definit?)
  ```

  is probably based on this fact.

- More than two different atom-types can not be treated yet within the NRL-TB parametrization implemented in the JuTiBi code. To treat more types of atoms the alternative approach to define the SKPs by hand has to be used.

3.6.2 Setting SKPs by hand

The second type of parametrization for the SKPs within the code demands a manual input of each Slater-Koster parameter. Therefore this parametrization is only suited for smaller systems with nearest- (and 2nd nearest-) neighbour coupling. In return this parametrization has the advantage to allow simple interpretations of the results and it exhibits a large parameter-freedom.

The simplest possible parametrization based on next neighbour coupling and identical basis atoms can be exclusively imported from the `inputcard`. In this case the keywords `IDENTATOM` and `ONLYNEXTN` have to be set to `T` and the NRL-TB parametrization has to be switched off by setting `SKPVARY` to `F`. If the overlap matrix should be included set `OVLAP` to `T`. The Slater-Koster parameters for the next-neighbour distance can be entered into the first columns of the following scheme:

```
-----------------------------------------------------------------------

ONSITES=0.0d0     0.0d0
ONSITEP=0.0d0     0.0d0
ONSITED=0.0d0     0.0d0

s-couplings:
VSS_SIGMA=0.0d0   0.0d0       OSS_SIGMA=0.0d0   0.0d0
VSP_SIGMA=0.0d0   0.0d0       OSP_SIGMA=0.0d0   0.0d0
VSD_SIGMA=0.0d0   0.0d0       OSD_SIGMA=0.0d0   0.0d0

p-couplings:
VPP_SIGMA=0.0d0   0.0d0       OPP_SIGMA= 0.0d0  0.0d0
VPP_PI=   0.0d0   0.0d0       OPP_PI=    0.0d0  0.0d0
VPD_SIGMA=0.0d0   0.0d0       OPD_SIGMA= 0.0d0  0.0d0
VPD_PI=   0.0d0   0.0d0       OPD_PI=    0.0d0  0.0d0

d-couplings:
VDD_SIGMA=0.0d0   0.0d0       ODD_SIGMA=0.0d0  0.0d0
VDD_PI=   0.0d0   0.0d0       ODD_PI=   0.0d0  0.0d0
VDD_DELTA=0.0d0   0.0d0       ODD_DELTA=0.0d0  0.0d0

-----------------------------------------------------------------------
```

The assignment of the Slater-Koster parameters to the keywords should be self-explaining. Note that the keywords of the form `V`$\mu\nu$`_m` are the SKPs for the Hamiltonian, whereas `O`$\mu\nu$`_m` are the SKPs for the overlap matrix, which have to be filled out only in the case

`OVLAP=T`. The two columns of values behind each keyword can be used to define spin-dependent SKPs. The first column represents the SKPs for the majority spin-channel, whereas the second column contains the SKPs for the minority spin-channel. If the keyword `SKPSPIN` (see section 3.7) is set to `F` the second column is not read in, therefore the SKPs are treated spin-independent, whereas in the case `SKPSPIN=T` both columns are imported.

Now there is also the possibility to generalize this scheme by considering not only next-neighbour coupling, but also the 2nd nearest neighbours, and using two different types of basis atoms, but then the SKPs have to be specified in the file `SKPinput`. The parametrization is based on the type of parametrization found in [46]. The following insight into the file should demonstrate how to work with this parametrization:

```
BOND   | VSS_SIGMA |  VSP_SIGMA | VSD_SIGMA | VPP_SIGMA
-------------------------------------------------------------------------------
1 1    |  0.0032       -0.0437      0.0        -0.0009
2 2    |  -0.1038      -0.0919      0.0         0.4828
1 2    |  1.4895       -1.5445      0.0         2.1517
2 1    |  1.4895        2.5864      0.0         2.1517
-------------------------------------------------------------------------------
1 1    |  0.0032       -0.0437      0.0        -0.0009
2 2    |  -0.1038      -0.0919      0.0         0.4828
1 2    |  1.4895       -1.5445      0.0         2.1517
2 1    |  1.4895        2.5864      0.0         2.1517


ONSITES
 -7.7859d0      -0.2676d0
 -7.7859d0      -0.2676d0
```

First the atom-types involved in the bonding have to be specified under the keyword `BOND` and then the value for the Slater-Koster parameter has to be set under the corresponding keyword matching to the SKP. For example the value in the row `2 2` under the keyword `VSP_SIGMA` is the Slater-Koster parameter $V_{sp\sigma}$ for the bonding between the two atoms of the same atom-type 2. Note that the SKPs for the bonding-types `1 2` and `2 1` have to be the same for symmetric SKPs, but are generally different for non-symmetric SKPs (as $V_{sp\sigma}$). The first 4 rows in above example are the SKPs for the majority spin-channel, whereas the following 4 rows (which have the same values in above example) are the SKPs for the minority spin channel. Under the keyword `ONSITES` the on-site energies for the s-orbital have to be specified. The first row stands for the majority spin, whereas the second row stands for the minority spin. The columns fit to the type of basis atom.

In principle this scheme can be extended to an arbitrary number of (different) basis atoms. However writing the input-file `SKPinput` would be a very unpleasant work.

Additionally the code was only tested for the case of of two different types of basis atoms, therefore debugging would probably be necessary.

3.7 inputcard: Magnetic properties and SOC

The next part of the inputcard is used to define the magnetic properties of the system. The keyword SKPSPIN is already discussed in the section 3.6, but due to its strong relationship to the magnetic properties it is located in the magnetic "chapter" of the inputcard.

```
-----------------------------------------------------------------------
*** Magnetic Properties and SOC ***

SPINLOG= T
SKP_SPIN=F

XC_ENERGY
    1   0.12   0.12   1.2
    2   0.01   0.01   0.1

STON_PARA
1 0.096d0 0.096d0 0.96d0
2 0.058d0 0.058d0 0.58d0

SET_GAXIS=F
ROT_SPIN= 0.0d0 0.0d0 0.0d0

SOCLOG= T

-----------------------------------------------------------------------
```

The keyword SPINLOG is used to include the spin-dependence into the Hamiltonian. Magnetism can only be calculated, if SPINLOG=T. Non-magnetic systems can be treated either by setting SPINLOG=F or by switching the exchange energies to zero for the case SPINLOG=T. In comparison the first method takes only approximately half of the computational time.

The exchange energies of the system for each basis atom enter under the keyword XC_ENERGY. For each basis atom the exchange energies (in eV) of the *s*-, *p*- and *d*-orbitals have to be put into one row. For transition metals the exchange splitting in the *d*-orbitals is the most important one. The self-consistency of the JuTiBi code is based on the *d*-moments and the Mulliken charges (see chapter 2), therefore I recommend to treat the *s*- and *p*-splitting by setting it 10-times smaller than the *d*-splitting. Note that the entered exchange energies act as starting values in the case of a self-consistent calculation (if RESTA_SC=F, see 3.8).

The magnetism in the JuTiBi code is based on a Stoner model, therefore an additional Stoner parameter is needed, which enters under the keyword `STON_PARA` for each atom-type.[3] Each row contains the *s*-, *p*- and *d*-Stoner parameter in eV, and I recommend again to use a 10-times smaller Stoner parameter for the *s*- and *p*-orbitals than for the *d*-orbitals. The Stoner parameters of a couple of elements can be found in [47, 48] and the parameters for Fe and Pt can be found in the appendix B of this documentation.

If only `SPINLOG=T` and `SOCLOG=F` the Hamiltonian will be diagonalized separately for each spin-channel by the code. Incorporating spin-orbit coupling or non-collinear magnetism leads to a Hamiltonian, which can not be diagonalized separately in the spin-channels any more. To take this into account switch `SOCLOG` to `T`. It is important to remember that the keyword `SOCLOG` does not only stand for the treatment of spin-orbit coupling. Always in the case where the full Hamiltonian in spin-space has to be diagonalized, use `SOCLOG=T`, therefore also for the non-collinear case as `nc_in_uc=T` or `blo_wave=T` (see later in this section)! Note that this leads to an approximately 4-times increased need of computational time. If no SOC is wished, but a non-collinear magnetic system should be treated use SOC-parameters of zero (see 3.5).

[3] Note that it is each atom-type and not each basis atom!

In the case SOCLOG=T one can decide to perform a global spin rotation to the system. To do this the program yields two possibilities. The first one is using the non-collinear magnetism scheme of the inputcard explained later in this chapter. I recommend to use this first method, because it is well-tested compared to the 2nd possibility, which is based on the keywords SET_GAXIS and ROT_SPIN. But nevertheless I will briefly explain the second method. In this case switch SET_GAXIS to T and define the new direction of the magnetic moments (in cartesian coordinates) by setting it behind the keyword ROT_SPIN. The direction does not have to be normalized and keep in mind that the rotation bases on the assumption that all spins were pointing collinear along the z-axis before. Therefore these method can not be combined with a non-collinear treatment of the spins. However, as mentioned before I recommend to use the non-collinear scheme, which first part is presented in the following:

```
-------------------------------------------------------------------------
*** Non-collinear-Magnetism (including anti-ferromagnetism) ***

NCMAG=T

NC_IN_UC=T

NC_ANGLES
1  0.0d0 0.0d0
2  180.0d0 0.0d0

-------Constraint for magnetic moments:
CONST_ANG=F
U_CONST= 0.d0


-------Spin-spirals:
BLO_WAVE=T
BLO_THETA= 0.0d0
BLO_Q= 0.5d0 0.0d0 0.d0

FLAT_SPIR=T

-------------------------------------------------------------------------
```

First of all the keyword NCMAG has to be switched to T to treat non-collinear magnetism. Then there are two main approaches to describe non-collinear magnetism (in particular spin-spirals) in the JuTiBi code. These two approaches are adumbrated in figure 2.13. If a treatment with manually fixed magnetic moments as in fig. 2.13a is wished, switch NC_IN_UC to T and enter the angles Θ and ϕ (see fig. 2.10) of each magnetic moment for each basis atom under the keyword NC_ANGLES. After specifying the basis atom via a number, enter first Θ and then ϕ in degrees and note that angles above 180°

are not allowed. This method of working with non-collinear magnetic systems has the advantage to allow a treatment of magnetic structures of arbitrary complexity. However, for treating spin-spirals there is a more efficient method using the generalized Bloch theorem (see section 2.8.2) incorporated into the code. To make advantage of this gen. Bloch theorem set the keyword `BLO_WAVE` to `T` and specify the type of spin-spiral including the cone-angle Θ and the spin-spiral vector $\boldsymbol{q}$ (see fig. 2.12). The cone-angle between 0° and 180° has to be entered behind `BLO_THETA` and the components of the spin-spiral vector q in cartesian coordinates and in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$ enter behind the keyword `BLO_Q`. Working with these coned spin-spirals does not allow to investigate the so-called Dzyaloshinskii-Moriya interaction in surface-systems (see [16]), because a spin-spiral rotating in the x-z-plane is needed to obtain the symmetry-breaking in the surface system. The rotation of a coned spin-spiral takes place inside the x-y-plane (i.e. the z-component of the magnetic moments is fixed). The JuTiBi code allows to treat so-called flat spirals (or planar spirals), in which the magnetic moments are lying completely inside the x-z-plane, by switching `FLAT_SPIRAL` to `T`. Note that in this case the cone-angle has to be set to zero!

The code allows also to incorporate phase-shifts to the angles Θ and $\phi_n = \boldsymbol{q} \cdot \boldsymbol{R_n}$ in the spin-spiral case. For example two anti-ferromagnetic coupled spin-spirals can be realized within this method as in the above example of the `inputcard`.[4] To treat such systems switch on `NC_IN_UC` and `BLO_WAVE` (and not only `BLO_WAVE`) and specify the additional phase-factors under the keyword `NC_ANGLES`. Note that in the flat-spiral case only additional phase-factor for the angle Θ can be used! For more details take a look into the description of the subroutines moments_local_to_global and create_Hmagnetic. It should be mentioned that the possibility to include phase-shifts for the spin-spiral calculations is not extensively tested yet, therefore be aware of bugs.

In addition the code allows to fix the Θ-angles of the magnetic moments (for the spin-spiral the cone-angle) with the help of a constraint (see 2.8). If the energy of a non-stable magnetic configurations should be calculated within the self-consistent TB-scheme (see fig. 2.9), this constraint has to be used. Otherwise the magnetic system converges against a stable (ground-state) solution. To activate this constraint switch `CONST_ANG` to `T` and enter a rather large energy-value of about 5 eV for the Lagrange parameter of the constraint behind the keyword `U_CONST`.

The two possibilities to describe non-collinear magnetic systems are described in much more detail in 2.8 and in particular also in the description of the subroutines (see section 6.2). For interested users I would also recommend to take a look into the appendix A.2, which describes rather detailed how the gen. Bloch theorem is implemented.

[4] Flat-spiral with anti-ferromagnetic coupled spins between the first and second basis atom of the unit cell, realized by incorporating a phase shift of $\Theta = 180°$ between the two basis atoms.

The JuTiBi code is in particular configured to treat spin-spiral systems. The second part of the non-collinear scheme in the `inputcard` can only be activated if `BLO_WAVE=T`. One can decide to calculate not only the spin-spiral for one $\boldsymbol{q}$, but rather all $\boldsymbol{q}$-values along a defined way (`LOG_Q_WAY=T`) or in a q-mesh (`LOG_QMESH=T`). Calculating both together within one calculation is not possible!

```
-------------------------------------------------------------------------

-------Treatment of SOC in spin-spirals:
SOC_PERTU=F
CHG_FE=F

-------q-mesh:
LOG_QMESH=F
BZQDIVIDE= 10 10 10
QPOIBZ= 1000
IRRQBZ=T

-------q-way:
LOG_Q_WAY=F
NUMQWAYS=3

NUM_QWAY   CREA_QWAY
 10              0.0d0 0.d0 0.d0        0.5d0 0.5d0 0.d0
 10              0.5d0 0.5d0 0.d0       0.5d0 0.5d0 0.5d0
 10              0.5d0 0.5d0 0.5d0      0.d0 0.d0 0.d0

-------------------------------------------------------------------------
```

If a calculation with $\boldsymbol{q}$-points in a q-mesh is wished specify the partitioning along the reciprocal Bravais vectors (`BZQDIVIDE`), the total number of q-points (`QBOIBZ`) and the usage of the irreducible part of the q-mesh (`IRRQBZ`) as in the case of the k-mesh creation (see section 3.3 for more information!). A calculation using a q-mesh allows to calculate the Heisenberg exchange-coupling values later on with the help of an external program. The file `jenerg.dat`, which is necessary for this determination is only created in the case `LOG_QMESH=T`.

To calculate the magnon dispersion of the magnetic system use a q-way by defining all start- and end-points of the separate paths, which sum up together to the q-way. The paths are the linear connections between the start- and endpoints given in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$, which should be specified under the keyword `CREA_QWAY`. The number of used paths has to be defined behind the keyword `NUMQWAYS`. For each path the number of used q-points enters under the keyword `NUM_QWAY`. In above example the q-way consists of three paths each containing 10 q-points. The first path is along $(0, 0, 0) \rightarrow (0.5, 0.5, 0)$, the second path is going along $(0.5, 0.5, 0) \rightarrow (0.5, 0.5, 0.5)$ and the third path along $(0.5, 0.5, 0.5) \rightarrow (0, 0, 0)$. Note that the maximum number of adjustable paths is 10.

The generalized Bloch theorem can not be used any more if spin-orbit coupling is considered. However, a 1st order perturbation theoretical treatment of spin-orbit coupling is possible (see 2.8.2). For example this becomes necessary in the investigation of the Dzyaloshinskii-Moriya interaction as done in [16]. To switch on the perturbation theoretical treatment of SOC set `SOC_PERTU` to `T`. If the Fermi energy should remain unchanged after adding the 1st order contributions use `CHG_FE=F` (see eq. 2.84), whereas a recalculation of the Fermi energy is done in the case `CHG_FE=T` (see eq. 2.88).

3.8 inputcard: Self-consistency

The following part of the inputcard contains all properties for the self-consistent scheme of the JuTiBi code (see fig. 2.9).

```
-----------------------------------------------------------------------
*** Properties for the self-consistent calculations ***

TB_SELFC=T

SELF_U
1 5.0d0
2 5.0d0

CHARGE_0
    1   8.0d0   8.0d0
    2  10.0d0  10.0d0

Method of Mixing:

MIX_LIN=F
MIX_ALPHA=0.1d0


MIX_BROY=T
N_IN_LIN=3

MAX_ITER=400
SELF_COND=0.00001d0

RESTA_SC=F
RESTA_IT=0

-----------------------------------------------------------------------
```

To activate the self-consistent scheme set TB_SELFC to T and specify the type of mixing. The code allows to use a simple linear mixing, which is very stable but also very slow. To use this type of mixing enter a T behind the keyword MIX_LIN. The linear mixing parameter is specified behind the keyword MIX_ALPHA. The mixing parameter has to be a value between 0 and 1.

The second possible mixing incorporated into the code is an extended Broyden mixing scheme [25, 49], which can be activated by setting MIX_BROY to T. Broyden mixing allows a very fast convergence, but could be unstable for some systems. If the JuTiBi code outputs non-physical magnetic moments or charges during the self-consistent cycles (f.ex. larger magnetic moments than the atomic ones) or diverges to infinity, try the

self-consistent scheme again with a better converged start-density or use linear mixing instead! If Broyden mixing is activated deactivate linear mixing by setting `MIX_LIN=F`, but note that also the Broyden mixing scheme needs a linear mixing parameter for the first `N_INIT_LIN` steps, which will be linear mixing steps. I would recommend to use at least `N_INIT_LIN=1` linear mixing steps before going to Broyden mixing. Even better are about 3-10 pre-linear mixing steps, to avoid divergence problems.

The code needs a maximum number of iteration-steps, after which it would stop automatically if the system is not converged yet. I recommend to leave this value behind the keyword `MAX_ITER` to 400. The value ϵ behind the keyword `SELF_COND` determines the digit up to which the self-consistency variables have to converge (see value ϵ in fig. 2.9). A value of 10^{-5} is reasonable, but for more precise calculations one should decrease it. However, keep in mind that the TB-scheme implemented in the JuTiBi code is not really suited for very precise and accurate calculations.

The "start-densities" in the self-consistent scheme of the JuTiBi code are the Mulliken charges and the magnetic d-moments of the basis atoms. The start-values for the magnetic moments are determined in the form of the exchange-energies[5] in the part of the `inputcard` about magnetism and SOC (see section 3.7). The Mulliken charges (in units of e) are defined in this part of the `inputcard` under the keyword `CHARGE_0`. After specifying the number of the basis atom two additional values have to be entered per row. The first value is the start-value of the Mulliken charge, whereas the second value is the desired reference charge within the local charge neutrality scheme (see section 2.5). I recommend to use the same values for the start-charges as long as one does not have converged or pre-converged charges for the system. Additionally the local charge neutrality part of the Hamiltonian needs a LCN-parameter U_{LCN} for each atom-type (not each basis atom!) in the unit cell. Specify them under the keyword `SELF_U`! I recommend to use a value of 5 eV for each type of atom.

Sometimes it is necessary to pre-converge the system and use the pre-converged values as start-values for the "real" self-consistent calculation. For this purposes there is the possibility to read in the start-values for the charges and the exchange-energies not from the `inputcard`, but rather from the file `TB_SC_values.dat`, which contains all converged values of the calculation before for each iteration step.[6] First one should switch `RESTA_SC=T` to activate this possibility. The user does not have to copy the values into the `inputcard` (but of course this is also working). One can specify the iteration-step inside this file, denoted by `ITER_`, from which the values are used as new start-values of the self-consistent cycle. This iteration-step enters behind the keyword `RESTA_IT`.

[5] Note that $\varepsilon_{i\mu}^{\mathrm{exc}} = I_{i\mu} \cdot m_i^d$.

[6] Note that this file is overwritten in each calculation, therefore make a safety-copy before.

3.9 inputcard: Density of states

In this part of the inputcard all properties for the calculation of the density of states (DOS) are specified. Additionally the properties of the Fermi-Dirac smearing function can be entered.

```
-----------------------------------------------------------------------
*** Properties for DOS calculation ***

NUMENDOS=1000
LORWIDTH=0.1d0
FACLOR= 30

SMEAR_MAG=T
FERM_BROD=0.001d0

DOSORBAT=T

LOGSPEC=F
SPECORBAT
 1 5

DOS_LOCAL=F

LOG_F_SUR=F

-----------------------------------------------------------------------
```

First I would suggest to take a look into the description of the subroutine calc_DOS, because a lot of useful details about the calculation of the DOS are presented there. As one can read there, the DOS-calculation needs an energy-mesh and the specifications about the Lorentzian functions entering the eq. 2.30. The number of energy points inside the energy-mesh is read in behind the keyword NUMENDOS. The width of the Lorentzians in eV is specified behind LORWIDTH and the factor FACLOR determines the range $2\alpha \cdot w$ (see subroutine calc_DOS) in which the DOS-calculation will be carried out. Note that all three properties NUMENDOS, LORWIDTH and FACLOR are closely related to each other in the DOS-calculation. Therefore changing only one quantity to improve the quality of the DOS is not enough in the most cases. To get a feeling how the DOS is changing with respect to this values work out the example 5.4 in chapter 5.

The total DOS is calculated by default, but the partial DOSs are not calculated by default. To activate the calculation of these atom- and orbital-resolved DOSs set DOSORBAT to T. If only the partial DOS to a special basis atom and orbital is needed, one can spare computational time by using LOGSPEC=T and specify the basis atom and

orbital under the keyword `SPECORBAT` for which the partial DOS should be calculated.[7] If all partial DOSs should be calculated leave `SPECORBAT=F`.

For a non-collinear magnetic systems it is useful to display the DOS in the local spin-frame of each basis atom instead of the global one. To activate a rotation of the eigenvectors into the local spin-frame before the calculation of the DOS, switch `DOS_LOCAL` to `T`. It should be mentioned that this property is not well-tested, therefore debugging could be necessary.

For the DOS calculation (and also for the calculation of the energies and charges etc.) usually a Fermi-Dirac smearing function is used to simplify the convergence and obtain "nicer" results. In the case of the DOS calculation this smearing function can be deactivated by setting `SMEAR_MAG` to `F`. However, I strongly advise against the deactivation of the smearing function, because it could lead to a lot of computational problems. The thermal smearing (or broadening) of the Fermi-Dirac functions $k_B \cdot T$ enters behind the keyword `FERM_BROD`. The smearing is given in eV and I suggest to use a smearing of about 1-10 meV to obtain accurate results combined with a stable convergence.

In this part of the `inputcard` the keyword `LOG_F_SUR` can be used to activate the creation of a file `fermi_surface.bxsf`, which can be imported into the program XCrysDen [50] to display the Fermi surface of the system. To do this set `LOG_F_SUR` to `T`.

[7] The assignment of the numbers to the orbitals is as follows: $1 : s$, $2 : p_x$, $3 : p_y$, $4 : p_z$, $5 : d_{xy}$, $6 : d_{xz}$, $7 : d_{yz}$, $8 : d_{x^2-y^2}$, $9 : d_{z^2}$.

3.10 inputcard: k-way

In this part of the inputcard the properties of the k-way can be entered. A k-way is necessary to investigate the band structure of the system.

```
 ---------------------------------------------------------------------
*** Properties for the band structure ***

NUMWAYS=4

NUMPTSWAY          CREATEWAY
 100                  0.5 0. 0.           0.5 0.5 0.
 100                  0.5 0.5 0.          0.5  0.5  0.5
 100                  0.5  0.5  0.5       0.0 0.0 0.0
 100                  0. 0. 0.            0.5 0.0 0.0

FAT_BANDS=T

---------------------------------------------------------------------
```

The structure of this part of the inputcard is almost the same as in the definition of the q-way to calculate the magnon dispersion of spin-spirals in the system (see section 3.7). The k-way is defined in separate straight k-paths, which start- and endpoints should be entered under the keyword CREATEWAY. This start- and endpoints of the k-paths have to be in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$. Set the number of separate k-paths behind the keyword NUMWAYS, which commands the code to read in only the first NUMWAYS rows under CREATEWAY. The number of k-points for each k-path are specified under the keyword NUMPTSWAY in front of the corresponding start- and endpoint of the k-path. In above example the k-way consists of four paths each containing 100 k-points. The first path is along $(0.5, 0, 0) \rightarrow (0.5, 0.5, 0)$, the second path is going along $(0.5, 0.5, 0) \rightarrow (0.5, 0.5, 0.5)$ and the third path along $(0.5, 0.5, 0.5) \rightarrow (0, 0, 0)$. The last path starts at $(0, 0, 0)$ and ends at (0.5,0,0). Note that the maximum number of adjustable k-paths is 10 in the JuTiBi code!

In addition to the corresponding eigenenergies for each k-point, there is the possibility to also write out the squares of the absolute values of the components of the corresponding eigenvectors into the external files (see chapter 4). This can be for example used to plot "fat bands" of the system (see f. ex. [22]). To activate this feature set FAT_BANDS to T.

4 Output of the JuTiBi code

In this chapter the output of the JuTiBi code is explained in detail. First the output on the screen during the calculations is explained and after that the structure of the created external files is shown in detail.

4.1 Output on the screen

The output on the screen should give the user the possibility to follow the calculation steps of the JuTiBi code in real-time. This output can also help to locate errors in the `inputcard` and I used it extensively for debugging. In the following the structure of this output is explained in detail, but note that I will not mention each aspect. The output can also vary in its structure depending on the type of calculation the user is performing (f.ex. settings of `spinlog`, `soclog`, `tb_sc` etc.). However, these explanations should help to introduce the user to the output, so that he should have no big problems to understand it for all possible settings.

The first sections display the most important entries of the `inputcard`, which are read in by the subroutine readdim (see section 6.2). These sections can be used to control the correctness of the `inputcard`. This output is structured within the thematic sections of the `inputcard`. For example the section about magnetism and SOC has the following form:

```
################################
Magnetism ,SOC:
READ IN : Consider electron spin
Dimension of the Hamiltonoperator  WITH   SPIN:           36
READ IN: Stoner-Parameters (s,p,d): atom-type            1 with
 9.600000000000000E-002  9.600000000000000E-002  0.960000000000000
READ IN: Stoner-Parameters (s,p,d): atom-type            2 with
 5.800000000000000E-002  5.800000000000000E-002  0.580000000000000
READ IN : Use NRL-TB parametrization
READ IN : Consider SOC
READ IN : Consider non-collinear magnetism
READ IN : Treat spin-spirals with gen. Bloch theorem
################################
```

Other thematic sections are the structure properties, mesh properties, the definition

of the k- and q-ways, the properties of the clusters, the properties of the DOS, and properties of the self-consistent TB-scheme. They are not explained in any detail here.

The next part of the output shows the entered lattice constants a, b and c, the Bravais vectors and the basis atoms of the system[1]. The subroutine lattix_TB calculates the reciprocal Bravais vectors, which are displayed in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$. Therefore the product between the matrix containing the Bravais vectors and the matrix containing the reciprocal Bravais vectors has to be the unity matrix! Additionally the volume of the unit cell (in units of $a \cdot b \cdot c$) and the basis atoms in cartesian representation are displayed.

```
################################
READ IN: Bravaisvectors and Lattice constants
READ IN : Lattice parameters   2.87000000000000        2.87000000000000
  2.87000000000000

READ IN : Bravais vectors
-0.500000000000000       0.500000000000000       0.500000000000000
 0.500000000000000      -0.500000000000000       0.500000000000000
 0.500000000000000       0.500000000000000      -0.500000000000000

READ IN : Basis atoms
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000
READ IN: Basisatoms in  CARTESIAN representation

lattix_TB: Reciprocal Bravais vectors:
 0.000000000000000E+000   1.00000000000000        1.00000000000000
  1.00000000000000       0.000000000000000E+000   1.00000000000000
  1.00000000000000        1.00000000000000       0.000000000000000E+000

lattix_TB: Volume of unit cell:  0.500000000000000
lattix_TB: Basis atoms in cartesian coord.:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000
################################
```

Now the symmetries of the lattice are determined in the subroutine findgrp (see section 6.2). First the number of symmetries is displayed and then a detailed analysis of the type of these symmetries is given.[2] In the lower example the symmetries of a bcc-lattice are presented. Afterwards the properties of the constructed k-mesh within the subroutine bzirr3d are shown, which are the number of k-points in the (irreducible) k-mesh and the volume of the full reciprocal unit cell in units of a, b, c. The product between the volume

[1] Row-wise and in units of a, b, c.

[2] Here the convention by Cornwell [51] is used.

of the reciprocal unit cell and the volume of the (real-space) unit cell has to be one! Note that an external file containing all k-points of the k-mesh (including the weights) is created, which is discussed in section 4.2 of this chapter.

```
################################
findgroup: Calculating Bravais- and Basisvectors into a.u.
findgroup: Number of symmetries:           48
findgroup: Symmetry-operations: E         C3alfa    C3beta    C3gamma
C3delta   C3alfa-1  C3beta-1  C3gamma-1 C3delta-1 C2x       C2y
C2z       C4x       C4y       C4z       C4x-1     C4y-1     C4z-1
C2a       C2b       C2c       C2d       C2e       C2f       IE
IC3alfa   IC3beta   IC3gamma  IC3delta  IC3alfa-1 IC3beta-1 IC3gamma-1
IC3delta-1IC2x      IC2y      IC2z      IC4x      IC4y      IC4z
IC4x-1    IC4y-1    IC4z-1    IC2a      IC2b      IC2c      IC2d
IC2e      IC2f
BZIRR3D: Irreducibel BZ will be computed:
BZIRR3D: Reciprocal lattice has   48 symmetries
BZIRR3D: The real lattice   48  symmetries will be used
BZIRR3D: Number k points in k-Mesh:         256
BZIRR3D: Volume of BZ in units of a*b*c:   2.00000000000000
############################
```

The next part of the output presents all interesting information about the construction of the neighbour shells. The imported radius `RMAXIMAL` of the `inputcard` is shown in units of a and as mentioned in section 3.4 this radius is used to define a rough first cluster containing only the Bravais vectors (see subroutine rrgen). The radius of this cluster, chosen automatically within the code, is displayed as cluster-radius `R` and it is usually much larger than the radius defined behind the keyword `RMAXIMAL` of the `inputcard`. Additionally the number of created Bravais vectors inside this cluster is displayed.

This created pre-cluster is used in the subroutine clsgen99 to obtain the neighbour shells of each basis atom. The number of neighbour atoms inside the cutoff radius[3] is shown for each basis atom. In addition a coupling matrix of the cluster is presented, which indicates whether the bondings between the basis atoms of the unit cell are inside the cutoff-radius, which would be indicated by a `1`. In the lower example a bcc-lattice with one basis atom was used, therefore the coupling matrix is rather uninteresting in this case. However, for large unit cells it could be interesting to take a look, which basis atoms are decoupled from each other in the cluster generation due to their large bonding

[3]This radius has to be smaller than the cluster radius and at least as large as the cutoff radius of the NRL-TB parametrization (if this parametrization is used)!

distance.

```
############################
RRGEN: Generation of real space cluster:
READ IN : Min. Radius in units of a:        4.500000
RRGEN: Cluster-Radius R: 8.250010 (units of a)
RRGEN: mesh divisions  :    10   10   10
RRGEN: vectors created :            4513
CSLGEN99: Clustergeneration for basis atoms:
READ IN : Cutoff radius for Cluster-Generation (in units of a)        3.500000
CSLGEN99: Atom            1 has cluster with          339 atoms
CSLGEN99: Coupling-Matrix of the Cluster:
  1 1
############################
```

Note that an external file containing the positions of all neighbours inside the cutoff-radius is created. This file will be discussed in detail in section 4.2 of this chapter.

The output of the last preparation steps before the diagonalization of the Hamiltonian are presented in the following. First the number of considered neighbours inside the cutoff-radius for each basis atom is presented again. Then the atom types in order of the basis atoms are shown. Afterwards the subroutine protohamiltonian reads in the start Mulliken-charges (in e) and start magnetic moments (in μ_B) in the d-orbitals of each basis atom. These values are displayed within following structure:

Number of basis atom, Mulliken-charge, cartesian x-, y- and z-component of the magnetic (net-)moments in the d-orbitals.

I recommend to take a look into these outputs during each calculation to check the `inputcard` for errors in the definition of the magnetic structure.

```
#########################
Starting protohamiltonian:
READ IN: Overlap included!
Number of (nearest) neighbours for 1.,2.,... atom:
        338
Atomtype of the 1.,2.,... atom:
          1
Starting values for the Charges
 and magnetic d-moment (x,y,z) for each basis atom:
 1   8.00000  0.00000  0.00000  2.50000
#########################
```

If the NRL-TB parametrization is used some additional information is given in the

output. First the type of parameter set of each atom type[4] imported from the file `SKP_input_papakonst` is specified with respect to the material and the overlap parametrization, which can be the old one or the new scheme (see [6]). Finally the number of neighbours inside the defined NRL-TB cutoff radius is presented. This number has to be smaller than the number of considered neighbours within the shell generation, otherwise an error message appears.

```
 ##########################
 READ IN : NRL-TB parameters
 READ IN : Iron       - old
 Considered neighbours within cutoff:         258
 ####################################################
```

After all these preparations the diagonalization of the Hamiltonian is starting. The Fermi energy, the charges and the total energy will be calculated and displayed on the screen. In the case of a spin-spiral calculation these values are determined for each q-point. In the case of a self-consistent calculation the charges and the Fermi energy are displayed for each iteration step. Therefore the output has the following structure:

```
q-Vector:    1   0.00000   0.00000   0.00000


 Iterationstep: **    1  **
     ... etc.

 SC-calculation converged!

 Now: Calculation of the energies.
 ################################
 ###### Energy analysis:
 Band energy in eV:  -1.56535
 Double counting corrections in eV:
 LCN:    0.00000
 Stoner model:    1.67002
 Constraint for mag. moments:   0.00000

 **Total energy**:    0.10467
 ################################


Finished q-vector: 0.000000  0.000000  0.000000  || Step    1 of   50
```

After specifying the q-vector in units $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$ the results for each iteration step

[4]Remember: Within the NRL-TB parametrization only up to two atom-types can be treated!

are presented, which structure will be discussed later in this section. If convergence is reached the band energy, the double countings and the total energy of the system (all in eV) are displayed (for more theoretical details see sections 2.4, 2.5 and 2.6). Finally the calculation of this q-vectors is finished and the number of remaining q-vectors for the calculation can be extracted from the output.

Now let us take a more detailed look into the structure of the output for each iteration step[5]. After specifying the iteration step the computational times for the two most time consuming calculation steps are displayed. These steps are the creation of $\mathcal{H}_0(\boldsymbol{k})$ within the Slater-Koster scheme by exploiting the (gen.) Bloch theorem and the diagonalization of the full Hamiltonian for one k-point. Usually the diagonalization step will be the most time-consuming part of the code, in particular for larger systems. In the case of a spin-spiral calculation the rotation due to the q-vector has to be excluded before constructing the magnetic part (for more details see appendix A.2). In this case the cartesian components of the magnetic (net-)moment in the d-orbitals without this q-rotation is displayed for each basis atom. This output is very useful to analyse occurring problems within a spin-spiral calculation and helped me a lot during debugging.

```
################################################
 Iterationstep: **    1  **
  Magnetic d-moment (x,y,z) without consideration
 of spiral-rotation for each basis atom:
  1   0.00000  0.00000  2.50000
 create_Hmagnetic: H_mag is fully occupied!
 ############################
 Time for creating H0:       0.00030000 s
 Time for one diagonalization:      0.00040000 s
 ############################
|||||||||| Diagonalization for all k-points done!
 Fermi energy (rough) :   1.98211891eV
 Fermi energy within Newton method:   1.98287868 eV after   5 iterations
 Determine the new mixed charges and moments:
  ** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
  1   8.00000   0.00000   0.00000   2.50939     0.0000     0.0000
 RMS in SC:  0.00939
 # Hits in SC:    3 of    4
 Time for iteration   1 :      16.87670000 s
################################################
```

The diagonalization of the Hamiltonian for all k-points of the k-mesh can take some time. Therefore a bar (||||||||||) consisting of single stripes is used to display the

[5]The output in the case `tb_sc=F` is a little bit different. It will not be explained here.

progress in this diagonalization. Each of this stripes corresponds to a 10%-progress, therefore the diagonalization for all k-points is finished after 10 stripes.

With the help of all calculated band energies the Fermi energy can be calculated. The first displayed value is only a roughly determined value, which is used as start value within a Newton-method to determine a precise value of the Fermi energy. The number of iterations required by the Newton method are also shown. This number should not exceed 200 iteration steps, otherwise I recommend to increase the Fermi smearing (see keyword `FERM_BROD` in `inputcard`) to simplify the convergence. The self-consistency is based on the Mulliken-charges (LCN!) and the magnetic d-(net-)moments of each basis atom. These values are displayed for each iteration step in following order: After specifying the basis atom, the Mulliken-charge and the cartesian components of the magnetic moments are given. At last the angles Θ and ϕ are displayed (see fig. 2.10). In the special case of `BLO_WAVE=T` and `NC_IN_UC=T` the angles only represent the phase shifts, as discussed in section 3.7!

After each iteration step the root-mean-square value of the vector containing the difference between the old and new self-consistent variables (i.e. the Mulliken-charges and d-moments of each basis atom) is displayed. The number of self-consistent variables fulfilling the convergence condition and the necessary number to obtain convergence are displayed as `Hits in SC`. Finally the computational time of each iteration step is shown.

Up to now the main part of the JuTiBi code is executed and only some optional calculations as the DOSs or the band structure can be performed. For the DOS calculation the output on the screen is limited to the following:

```
 ##########################################
 # Start calculating DOS properties       #
 ##########################################
### Fully integrated DOS:   17.79295 | theoretical:    18
 WARNING: Integrated DOS not very accurate!
 To be sure you can increase variable faclor!
 ####### Calculation of the partial DOSs (majority spin channel):
|||||||||
 ####### Calculation of the partial DOSs (minority spin channel):
||||||||| DOS-calculation finished!
```

Within the calculation of the Total DOS the integrated DOS will be also determined. The integrated DOS should be approximately the total number of bands. However, using the Lorentzian functions leads to some small mistake in the integrated DOS. If the difference between the theoretical and the calculated values is larger than 0.1 a warning message appears as in the above example. To improve the calculated values one can play with the properties of the DOS in the `inputcard`. However, note that these properties are very strongly linked to each other, therefore finding an optimal input could be difficult (see section 5.4).

In the case `SPINLOG=T` and `DOSORBAT=T` the DOS in each spin-channel for each orbital and atom will be calculated. To follow the progress of these calculations progress bars as in the case of the diagonalization are used. Note that the DOS are saved into external files, which are explained in section 4.2 of this chapter.

Finally the band structure of the system along a defined high-symmetry way in the reciprocal unit cell can be calculated. Note that this band structure can be only calculated for the last converged values of Mulliken-charges and magnetic moments! The defined k-paths are displayed and an additional progress bar is used to follow the progress of the diagonalization routine.

```
 ################################
 # Calculation of band structure #
 ################################
 READ IN : k-way (in units of 2*pi (1/a,1/b,1/c))
  0.500000000000000       0.000000000000000E+000  0.000000000000000E+000  ->
  0.500000000000000       0.500000000000000       0.000000000000000E+000
  0.500000000000000       0.500000000000000       0.000000000000000E+000  ->
  0.500000000000000       0.500000000000000       0.500000000000000
  0.500000000000000       0.500000000000000       0.500000000000000       ->
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000  ->
  0.500000000000000       0.000000000000000E+000  0.000000000000000E+000
|||||||||| Calculation of band structure finished!
```

If everything worked out, the code quits without any error message and the following lines should be visible in the output:

```
####################
# JuTiBi finished! #
####################
```

4.2 External files

This section discusses the structure of the external files created by the JuTiBi code so far. Each file contains a header explaining briefly the structure of the file. Not all data calculated within the JuTiBi code is written out into files, therefore for some user purposes the code has to be modified.

4.2.1 Properties of the lattice structure

The constructed k-mesh and the weights of the k-points can be found in the file `kmesh.dat`. If the irreducible part of the k-mesh is calculated (`IrrBZ=T`), only the k-points inside this irreducible part are stored in this file. Otherwise all k-points are stored. I show some small section of this file:

```
0.13937282    0.13937282    0.00000000    0.00300000          12
0.15679443    0.15679443    0.00000000    0.00300000          12
0.17421603    0.17421603    0.00000000    0.00150000           6
0.03484321    0.01742160    0.01742160    0.00600000          24
0.05226481    0.03484321    0.01742160    0.01200000          48
0.06968641    0.05226481    0.01742160    0.01200000          48
```

The first 3 columns are the cartesian components k_x, k_y and k_z of the k-points in units of 2π. The lattice constants are contained in the k-points in the form $\frac{1}{a}, \frac{1}{b}, \frac{1}{c}$. The 4th column is the weight of the k-points, which is not directly used in the code. Instead the degeneracy of the k-point displayed in the 5th column is used in integrations over the k-mesh. Note that the degeneracies are all `1` in the case of `IrrBZ=F`!

The file `shells_neighbours.dat` contains the created neighbour-shells for all basis atoms of the lattice. Therefore this file can be used to plot the real-space structure of the lattice. The structure of the file is as follows:

```
############## Basis atom:          1
  1    0.00000000    0.00000000    0.00000000    0.00000000
  2   -0.50000000   -0.50000000   -0.50000000    0.86602540
  3    0.50000000   -0.50000000   -0.50000000    0.86602540
  4   -0.50000000    0.50000000   -0.50000000    0.86602540
```

For each basis atom (specified by `Basisatom:...`) the neighbour shells are presented. The number in the first column denotes the neighbour of the basis atom. The next three columns are the cartesian components of the bonding vector in units of the lattice constant a (therefore the fraction $\frac{b}{a}$ and $\frac{c}{a}$ is considered). The last column contains the distances in units of a from the central atom of the cluster. Note that each cluster is sorted in ascending order by the distance (and then the z-, y- and at last the x-component)!

4.2.2 Charges

The charges of the system are stored in the two files `charges.dat` and `conv_charges.dat`. The first file contains all elements of the density matrix (see eqs. 2.47, 2.48 and

2.54) for each iteration step and each q-point (if `BLO_WAVE=F` $\boldsymbol{q} = 0$ is used in the files). However, usually the file `conv_charges.dat` contains enough informations for the user. In this file all converged charges and magnetic moments (both Mulliken- and net-charges and moments) are stored for all q-vectors. Let us first take a look into the file `charges.dat`:

```
 #q-vector (if any):   0.000000    0.000000    0.000000
 #Iterationstep:            1
 #(net-) charges:
1  1     1    0.135803
1  2     1    0.047178
1  3     1    0.060714
1  4     1    0.050306
...
 #Mulliken-charges:
1  1     1    0.245466
...
 #cross-charges:
1     1    0.000000  0.000000
...
```

The elements of the density matrix are stored in following order for each q-vector and each iteration step. First the spin-, orbital and atom-resolved net-charges are saved into the file, then the Mulliken-charges are stored. Afterwards the orbital- and atom-resolved cross-charges (both the Mulliken- and net- ones) are listed. In the case of the net- and Mulliken-charges the first column represents the spin-channel with $\uparrow$:1 and $\downarrow$:2, the second column stands for the kind of orbital[6] and the third column contains the basis atom. Finally the value of the charge (in e) is saved into the last column. In the case of the cross-charges there is no column containing the spin-channel, but all other columns show the same structure as explained above.

```
###### Atom            1
 ### Charges:
# Net-charges (total, s, p ,d) : 6.94401   0.26788   0.31403   6.36210
# Mulliken-charges (total, s, p ,d) : 8.00000   0.48691   0.78831   6.72477
 ### Net-Magnetic Moments (total, x, y, z) :
# s-moment:  0.02378   0.00000   0.00000  -0.02378
# p-moment:  0.04397   0.00000   0.00000  -0.04397
# d-moment:  2.64127   0.00000   0.00000   2.64127
 ---------------------------------------------
```

[6] $1: s$, $2: p_x$, $3: p_y$, $4: p_z$, $5: d_{xy}$, $6: d_{xz}$, $7: d_{yz}$, $8: d_{x^2-y^2}$, $9: d_{z^2}$

```
# total moment:  2.57352   0.00000   0.00000   2.57352
 ### Mulliken-Magnetic Moments (total, x, y, z) :
# s-moment:  0.02955   0.00000   0.00000  -0.02955
# p-moment:  0.06823   0.00000   0.00000  -0.06823
# d-moment:  2.43361   0.00000   0.00000   2.43361
 ------------------------------------------------
# total moment:  2.33582   0.00000   0.00000   2.33582
```

The file `conv_charges.dat` contains the converged charges and magnetic moments of each q-vector. First all charges and moments are sorted by the basis atoms. The net-charges and Mulliken-charges are given in following order: first the total charge, then the charge in the s-, p- and d-orbitals is displayed. The magnetic moments are located under these charges. First the magnetic net-moments in the s-, p- and d-orbitals are given in units of μ_B, then the total magnetic net-moments are displayed. For all these moments the first column represents the absolute value, whereas the second to the fourth column contain the corresponding magnetic moments in cartesian coordinates (i.e. the x-, y- and z-component). Finally the magnetic Mulliken-moments are displayed in the same structure as the net-moments.

In addition the Mulliken-charges (LCN!) and the exchange energies (Stoner model) needed for the self-consistent TB-scheme of the JuTiBi code are saved into the file `TB_SC_values.dat` in each iteration step in order to use them as start values in following calculations (see section 3.8).

```
 SCVALUES

ITER_   2
    1   0.1204506014   0.1204506014   1.2045060135
    1   8.0000000000   8.0000000000

ITER_   3
    1   0.1208823931   0.1208823931   1.2088239309
    1   8.0000000000   8.0000000000

ITER_   4
    1   0.1212880679   0.1212880679   1.2128806791
    1   8.0000000000   8.0000000000
```

It is very important to leave the structure of this file unchanged, otherwise the data can not be imported for another self-consistent calculation as new start values! `SCVALUES` works as keyword to read in the data and the `N`th-iteration step is denoted by `ITER_N`. For each iteration step the exchange energies and Mulliken-charges of each basis atom

are displayed. The structure is exactly the same as in the corresponding parts of the `inputcard` (see sections 3.7 and 3.8). Therefore these values can be also copied into the `inputcard` to start a new self-consistent calculation with pre-converged results.

4.2.3 Energies of a spin-spiral

By performing a spin-spiral calculation (`BLO_WAVE=T`) along a high-symmetry q-way the file `Eq.dat` containing the q-dependent energies is created. This file consists of 9 columns, containing all relevant energies of the system. In the first column one can find the length-parametrized value of the q-way and the next three columns describe the cartesian components of the q-vector in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$. In the 5th column the band energy of the system (see eq. 2.34) for the corresponding q-vector is saved, whereas the total energy is located in the 6th column. The double-counting contributions are listed in the next three columns in following order: first the double counting coming from the local charge neutrality (eq. 2.42), then the double counting of the Stoner model (eq. 2.58) and finally the contribution arising from the constraint of the Θ-angle of the magnetic moments. Note that all energies are given in eV!

If no spin-spiral is taken into account the energies can be found in the output on the screen as explained in section 4.1. If a spin-spiral is considered and the q-points are not located along a defined high-symmetry way but rather inside the reciprocal unit cell the relevant data is saved into the file `jenerg.dat`. This file will be explained later in this section.

During my work of my diploma thesis [16] it was necessary to exploit a 1st order perturbation theoretical treatment of SOC for a spin-spiral system to calculate the Dzyaloshinskii-Moriya interaction. In total three files are created if the user decides to treat SOC in 1st order perturbation theory (see keyword `SOC_PERTU` in chapter 3). These three files are named as `Eq_1storder_SOC.dat`, `Eq_1storder_SOC_layer_resolved.dat` and `Eq_1storder_SOC_layer_orbital_resolved.dat`. I am going to explain only the first file more detailed, whereas the other files will be only briefly discussed.

In the file `Eq_1storder_SOC.dat` the SOC-contribution to the energy calculated within 1st order perturbation theory (see eqs. 2.84 or 2.88) is saved for all q-points along a defined q-way. The first column is again the length-parametrized value for the q-way and the three following columns display the q-vector in cartesian coordinates. The last column contains the energy-contribution in eV coming from the 1st order SOC treatment. The files `Eq_1storder_SOC_layer_resolved.dat` and `Eq_1storder_SOC_layer_orbital_resolved.dat` contain the layer-resolved (respectively the layer- and orbital-resolved) analysis of the 1st order SOC-contribution to the energy. Interested users should take a look into the header of the file to understand its structure.

4.2.4 Density of states

I recommend to take a look into the description of the subroutine calc_DOS in the section 6.2. There the way of calculating the DOS is described in detail. The total DOS and the partial DOS are saved into following files depending on the settings for `SPINLOG` and `SOCLOG`:

`spinlog=false`: Total DOS saved in `Total_DOS_up.dat` and the partial DOS in `DOS_atom_orbital_resolved.dat`

`spinlog=true`, `soclog=false`: Total DOS saved depending on the spin-channel in `Total_DOS_up.dat` and `Total_DOS_down.dat`, the partial DOS in `DOS_atom_orbital_resolved_up.dat` and `DOS_atom_orbital_resolved_down.dat`

`soclog=true`: Total DOS saved in `SOC_Total_DOS.dat`, and the partial DOS depending on the spin-channel in `SOC_DOS_atom_orbital_resolved_up.dat` and `SOC_DOS_atom_orbital_resolved_down.dat`

```
...
11.2446904888  0.1047001324
11.3293397442  0.0825157774
11.4139889997  0.0581123860
11.4986382551  0.0515929673
...
```

First I will explain the files containing the Total DOS of the system. The structure of all these files is the same: in the first column the energies[7] of the constructed energy mesh for the DOS calculation are saved and in the second column the DOS is listed in arbitrary units. In the case of `soclog=false` the file `Total_DOS_up.dat` contains the DOS in the majority spin-channel, whereas `Total_DOS_down.dat` contains the values for the minority spin-channel. If a non-magnetic calculation with `spinlog=false` is performed only the file `Total_DOS_up.dat` is created. Note that in the case of `soclog=true` the file `SOC_Total_DOS.dat` does not separate between the spin-channels.

The partial DOS is calculated in the case `DOSORBAT=T` and saved into the above listed files depending on `soclog` and `spinlog`. The structure of these files is in all cases the same. In the first column the energies of the energy-mesh are listed and then the partial DOS for each orbital and each atom is displayed. The next 9 columns display the densities of states of the first basis atom in the used orbitals[8], and behind that the partial DOS of the second basis atom etc. is presented. As in the above explained case of the total DOS the files with the suffix `_up` and `_down` contain the partial DOS of the

[7] in eV

[8] $1: s$, $2: p_x$, $3: p_y$, $4: p_z$, $5: d_{xy}$, $6: d_{xz}$, $7: d_{yz}$, $8: d_{x^2-y^2}$, $9: d_{z^2}$

majority and minority spin-channel. The prefix `SOC_` denotes the files containing the DOS in the case of `soclog=T`.

Note that the above explained structure of the files containing the partial DOS is different in the case of `LOGSPEC=T`. In this case the partial DOS of only one specific basis atom and orbital is calculated. Therefore only one additional column containing this specific partial DOS is displayed behind the column listing the energies of the energy-mesh.

4.2.5 Band structure

The band structure of the system is saved in following files depending on the settings for `SPINLOG` and `SOCLOG`. If `SOCLOG=T` the file `SOC_bandenergies.dat` is used to save the energies along the high-symmetry k-way. Note that in this case the energies can not be strictly separated in the majority and minority spin-channel. In the case `SOCLOG=F` the bands in the majority spin-channel are saved into the file `bandenergies_up.dat` and the minority spin-bands are saved into `bandenergies_down.dat`. In the case of `SPINLOG=F` only the file `bandenergies_up.dat` is used to store the band energies. All files have the same structure: the first column contains the length-parametrized k-way, whereas the next 3 columns show the corresponding k-point of the k-way in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$. The last column displays the energy of this k-point. Note that the different bands are stored one after another in rows.

```
...
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2974698615
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2974698615
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2400692452
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2400692452
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2390131055
0.0000000000    0.5000000000    0.0000000000    0.0000000000    -2.2390131055
...
```

If a fat-band representation is switched on (see keyword `FAT_BANDS` in chapter 3), the weights of the corresponding eigenvectors are saved in additional columns. Interested users should take a look into the header of the file to gain specific information.

4.2.6 Miscellaneous

The last two files `jenerg.dat` and `fermi_surface.bxsf` can be used as input in external programs to maintain the Heisenberg exchange-coupling parameters or the Fermi surface of the system. I will only give a brief description of the structure of the files.

The file `jenerg.dat` contains all information to obtain the Heisenberg exchange-coupling parameters within a linear least square method. This file can be used as input

for the Python program `plot_Jij.py` written by D. Bauer.

```
# This file contains the total energies in eV
# in dependence of the q-vectors: (q_x, q_y, q_z, E(q))
# The q-vectors are in units of 2*pi, but the
# the lattice constants a, b and c (in A) are contained.
# The cartesian representation is used.
# The Bravais vectors of the system in # cartesian representation:
#   -1.4350000000      1.4350000000      1.4350000000
#    1.4350000000     -1.4350000000      1.4350000000
#    1.4350000000      1.4350000000     -1.4350000000
# Cone angle:    1.57079632679490
# Magnetic moments:   2.50000000
0.0000000000   0.0000000000   0.0000000000   0.1118937017
0.0348432056   0.0348432056   0.0000000000   0.0835111066
0.0696864111   0.0696864111   0.0000000000   0.0102684273
```

Beside the information about the Bravais vectors (in units of a, b and c), the cone angle of the spin-spiral and the absolute value of the magnetic moment, the q-dependent band energies are displayed in the 4th column for each q-point of the q-mesh. The q-points are listed in cartesian components in units of 2π in the first three columns.

The file `fermi_surface.bxsf` can be exported into the program XCrysDen to visualize the Fermi surface of a 3-dimensional system[9]. The structure of this file is given on the webpage of the XCrysDen code. The users interested in more details of this file should take a look into this reference [50]. Additionally in the newest version three files `basics.berrycurv`, `hopping.berrycurv` and `charges.berrycurv` are written out, if a Berry curvature analysis with the external code by Hongbin Zhang should be carried out. The correspondin code of H. Zhang and a short documentation can be also found in the JuTiBi package.

[9]The JuTiBi code does not provide any files to calculate the Fermi surface of a 1- or 2-dim. system!

5 Tutorial for the JuTiBi code

The tutorial for the JuTiBi code should help the user to get used to the `inputcard` and learn how to perform the basic calculations. Most of the exercises and examples are chosen to be as simple as possible to allow a smooth introduction. However, some of the examples are a little bit more tricky and can show numerical difficulties. These examples are mostly called "exercises for experts". These cases should give the user an idea, what one has to take care of.

The user should learn how to calculate band structures of non-magnetic and magnetic systems, including the procedure of self-consistency. Calculating the DOS, the magneto-crystalline anisotropy energy (MCA) and performing non-collinear magnetic calculation with the code.

I am assuming that the user proceeds linearly through this chapter. Therefore in later exercises some requirements from the first exercises are needed.

5.1 Get JuTiBi to run

First you should get the JuTiBi code running. Copy the folders containing the JuTiBi code into a folder of your choice. A complete list of all files contained in the folder of the JuTiBi code can be found in the appendix F. Then use a shell and change into the JuTiBi-directory. To compile the code type in `make`[1]. To compile the code you need a `Fortran90` compiler, as f. ex. `ifc` or `ifort`. I have tested the code only for `ifc` and `ifort` so far, therefore I would recommend to use them if possible. In addition a `Lapack`/`BLAS` library (preferably the MKL one; see webpage [52]) has to be installed on your computer.

To get more details about the used Compilers, paths etc. take a look into the `makefile`. If the code is not being compiled on your computer, probably you have to change the path of the `Lapack` libraries etc. in the `makefile`. Congratulations if you were successful in compiling the JuTiBi code. You have passed the possibly largest obstacle of working with the code ;)

[1] Additional flags as "`make mac`" or "`make iff`" can be used to use alternative paths for the MacOSX or the `IFF` Desktop PCs. The flag "`make clean`" can be used to erase all . o and . mod files in the directory. For more details take a look into the `makefile`.

5.2 General remarks

The `inputcard` has to be used to assign the system you want to compute. I will not explain every detail of setting the `inputcard` for the particular exercises, but only the most important points. Therefore I strongly recommend to read the chapter 3 before and when working through the tutorial. However, for users who only want to calculate the systems of the tutorial the directory `inputcards_for_Tutorial` contains "ready-to-compute" inputcards for each exercise in the tutorial.

Additionally the file `SKP_input_papakonst` has to be used occasionally to assign the correct Slater-Koster parameter sets from the webpage [4] to the system. For more details see section 3.6. Again the necessary files to compute the exercises of the tutorial can be found in the directory `inputcards_for_Tutorial`.

5.3 Band structure calculation

In this section you should learn how to calculate the band structure of a simple system. First we should focus on a non-magnetic system because no self-consistency is needed there to obtain the band structure. Afterwards you should perform a calculation for a simple magnetic system to get used to the self-consistent scheme.

5.3.1 Non-magnetic system: fcc Cu

Goals:

- defining lattice structures
- working with parameter sets of the NRL-TB parametrization
- performing band structure calculations along a defined high-symmetry k-way

files: `inputcard_Cu.dat`, `parameter_set_Cu`

As non-magnetic material we will choose copper in a fcc-lattice structure with a lattice parameter of $a = 3.61$ Å. First you should import the parameter set of Cu from the webpage [4] and paste it unchanged into the file `SKP_input_papakonst`. Remember that the parameter set has to be put under the keyword `STA_READ`[2] to define the first atom type of the system. The second atom type, which would be defined with a parameter set under the keyword `STA2_READ` can be an arbitrary set, because it will be not used in this example.

[2]Note that this keyword has to be unique in the file `SKP_input_papakonst`! For more details see section 3.6.

Then you should enter the lattice structure in the `inputcard`, which should look like this:

```
ALATBASIS=  3.61d0 1.0d0 1.0d0

BRAVAIS
0.0d0 0.5d0 0.5d0
0.5d0 0.0d0 0.5d0
0.5d0 0.5d0 0.0d0
```

Specify the number of k-points to 8000 in the full BZ, which is enough for Cu. You can use the irreducible part of the k-mesh.

```
BZDIVIDE= 20 20 20
KPOIBZ= 8000
IrrBZ= T
```

The number of valence electrons in Cu is 11, therefore enter `NUMELUC=11` and set (the start-values of) the Mulliken charge and the reference charges to 11.0:

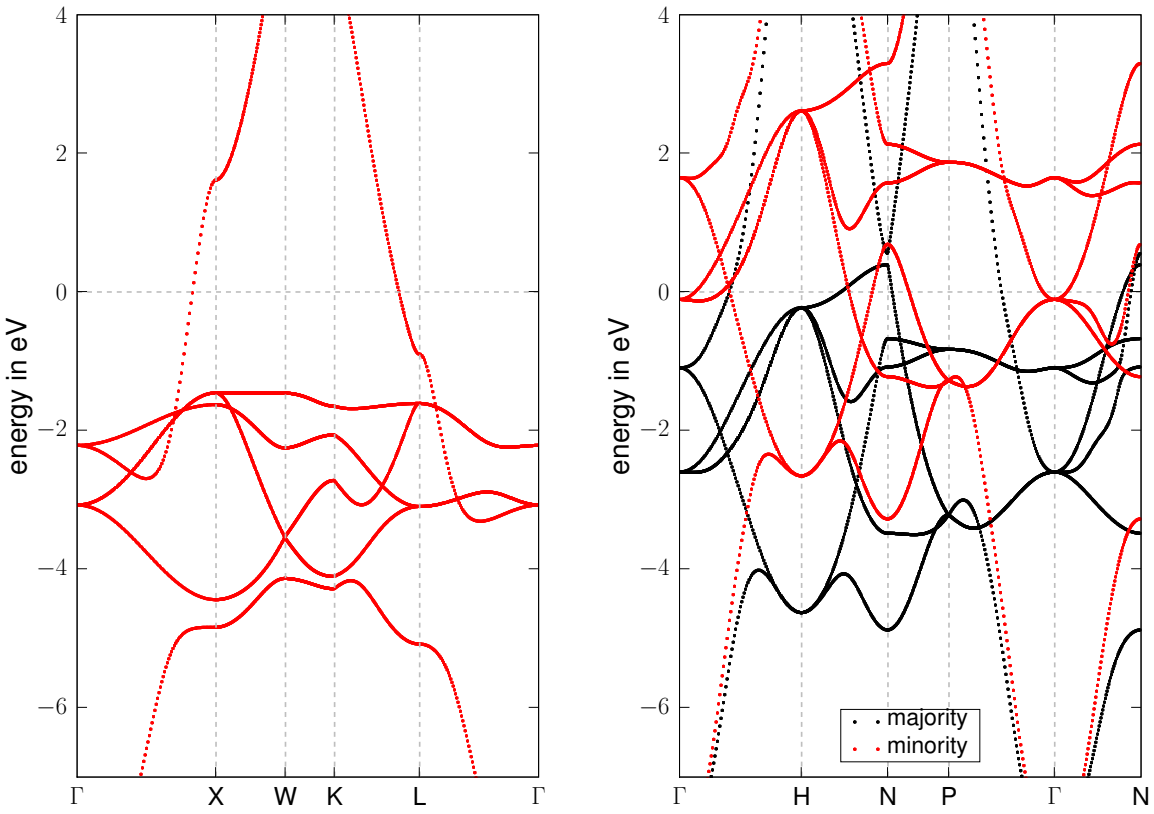
```
CHARGE_0
    1   11.0d0   11.0d0
```

To perform a non-magnetic calculation without self-consistency switch `SPINLOG=F` and `TB_SELFC=F`. To obtain the correct double countings you should also set the exchange energies to zero to be consistent. Finally you can specify the k-way by defining the start- and end-points of the separate k-paths. For a way along $\Gamma \to X \to W \to K \to L \to \Gamma$ it should be as follows:

```
NUMWAYS=5

NUMPTSWAY          CREATEWAY
 100                  0.0 0.0 0.0          1.0 0.0 0.0
 100                  1.0 0.0 0.0          1.0 0.5 0.0
 100                  1.0 0.5 0.0          0.75 0.75 0.0
 100                  0.75 0.75 0.0        0.5 0.5 0.5
 100                  0.5 0.5 0.5          0.0 0.0 0.0
```

Now run the code with `JuTiBi.exe` and wait until the calculation is finished. You should take a look into the output on the screen and try to understand it with the help



(a) Band structure of Cu.

(b) Band structure of Fe.

Figure 5.1: Band structures of fcc-Cu (a) and bcc-Fe (b) along high symmetry k-ways using 8000 k-points. The Fermi energy lies at zero.

of chapter 4. The interesting data of the band structure is now contained in the file `bandenergies_up.dat`. Plot the first column against the 5th column (see subsection 4.2.5 for more details). The band structure should look like in figure 5.1(a). Note that all eigenenergies saved in the files containing the band energies are shifted already by the Fermi energy.

5.3.2 Magnetic system: bcc Fe

Goals:

- using the self-consistent TB-scheme
- output of "simple" magnetic system

files: `inputcard_Fe.dat`, `parameter_set_Fe`

As a magnetic system we should compute the band structure of bcc-Fe. Use the paramagnetic GGA parameter set for the NRL-TB parametrization [4] and enter the Bravais vectors of a bcc-lattice with a lattice parameter of $a = 2.87\,\text{Å}$. 8000 k-points are enough to describe the band structure. The irreducible k-mesh can be used to spare computational time. Set the number of electrons, the Mulliken charge and the reference charge to 8. Use an start-value of about 1.1 eV for the exchange splitting with a Stoner parameter of $I = 0.96\,\text{eV}$[3]:

```
XC_ENERGY
 1  0.11d0   0.11d0   1.1d0

STON_PARA
 1  0.096d0 0.096d0 0.96d0
```

Switch on the self-consistent scheme by setting `TB_SELFC=T` and specify the type of mixing (more details in section 3.8). I recommend to use Broyden mixing, but you can also use linear mixing, which is in particular helpful for hardly converging systems. Now start JuTiBi and observe the convergence of the magnetic moment. The magnetic moment converges to a value of about $2.6\,\mu_B$.

```
** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
1   8.00000   0.00000   0.00000   2.64127     0.0000     0.0000
```

Keep in mind that this is the magnetic net-moment in the d-orbitals[4]! To see the physically more meaningful magnetic Mulliken-moment take a look into the file `conv_charges.dat`, which is explained in detail in section 4.2.2. As one can see a value of $2.3\,\mu_B$ is in nice agreement with *ab-initio* calculations.

```
### Mulliken-Magnetic Moments (total, x, y, z) :
# s-moment:  0.02955   0.00000   0.00000  -0.02955
# p-moment:  0.06823   0.00000   0.00000  -0.06823
# d-moment:  2.43360   0.00000   0.00000   2.43360
 ------------------------------------------------
# total moment:  2.33582   0.00000   0.00000   2.33582
```

The band structure decomposed into the majority and minority spin bands is displayed in fig. 5.1(b). It would be good if you make a safety copy of the file `TB_SC_values.dat`, because we can use it for later exercises.

[3] in the d-orbitals

[4] This is the only quantity beside the Mulliken charge, which enters the self-consistency.

It is also enlightening to change a little bit the start-values of the exchange energies and observe the convergence behaviour within Broyden mixing. In some cases the Broyden mixing can lead to unphysical results, however usually this problem can be solved by using more linear pre-iteration steps (`N_INIT_LIN`) before using Broyden mixing.[5]

Exercises for "experts": It could be interesting to calculate the band structure of Cu within a magnetic self-consistent calculation. Then you should start with a (not too large) exchange splitting of about 0.5 eV and a Stoner parameter of $I = 0.73\,\text{eV}$ (see [48]) and observe if the magnetic moment converges against zero. This is indeed the case in Cu, however in fcc-Pt (use the parameter set from [4]) it is much more complicated. For an equilibrium lattice constant of $a = 3.92\,\text{Å}$ and a Stoner parameter of $I = 0.58\,\text{eV}$ the system converges to a small magnetic moment with a magnitude of about $0.01\,\mu_B$. The exact value depends on the number of k-points and one can observe the behaviour that the moment becomes smaller for more k-points. But the convergence is rather bad. This is an indication that the Stoner parameter is chosen a little bit too large. However, in an investigation of my diploma thesis this Stoner parameter leads to reasonable results compared to corresponding *ab-initio* results (see fig. 4.3 in [16]).

5.4 Density of states (DOS): bcc Fe

Goals:

- use (pre-)converged start values
- calculating the DOS (total, partial)
- get a feeling for the quantities used to calculate the DOS

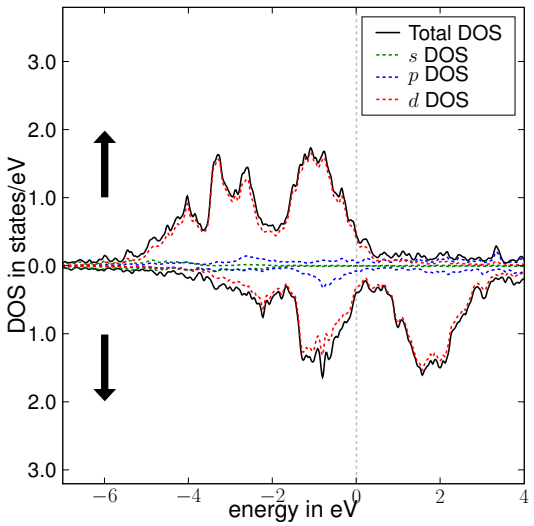
files: `inputcard_Fe_DOS.dat`

In this exercise you should calculate the DOS of the bcc-Fe system. I hope you saved the file `TB_SC_values.dat` of the bcc-Fe band structure calculation.[6] The converged values for the bcc-Fe calculation are saved under the last iteration step. Therefore something like the following can be found in the file:

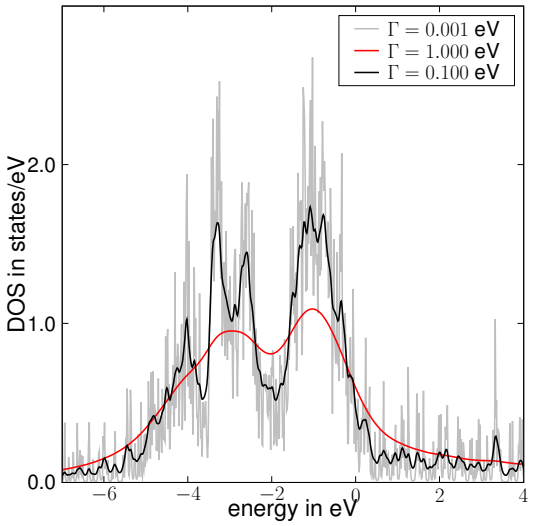
```
ITER_   7
    1   0.1267807628   0.1267807628   1.2678076284
    1   8.0000000000   8.0000000000
```

[5] Unfortunately this is not valid always. In some cases the linear pre-iteration steps could lead to a more difficult convergence. I do not know the origin of this behaviour yet.

[6] If not you can also do this exercise, but you will miss the first goal of this exercise.



(a) DOS of bcc-Fe.



(b) DOS in dependence of width of Lorentzians.

In above case the 7th iteration gave me the converged result for the exchange energies. Therefore to calculate the bcc-Fe system, one can use these converged results to spare some computational time[7]. To do this you can either copy this converged values into the `inputcard` as new start values, or you use `RESTA_SC=T` and `RESTA_IT=7`.

Now you need to specify the properties for the DOS calculation. First we want to calculate also the partial DOS, therefore switch `DOSORBAT=T`. Then specify the width of the lorentzian functions, the factor α and the number of energy points for the energy mesh (see subroutine calc_DOS in section 6.2). A good choice to obtain a nice smooth DOS (see left panel of fig. 5.4) around the Fermi energy is as follows:

```
NUMENDOS=5000
LORWIDTH=0.1d0
FACLOR= 20
```

I would recommend to play a little bit with these quantities. They are all coupled and therefore it is good to get a feeling how to obtain a reasonable look of the DOS. A much too small width of the Lorentzians leads to a very spiky, not very meaningful DOS, whereas a too large width smears out all interesting physical properties of the DOS. In the right panel of fig. 5.4 one can see the DOS for this extreme cases compared to the choice in the left panel of fig. 5.4.

[7] In this particular case it does not give you a big advantage. However, for larger systems it could be very handy.

5.5 MCA of an Fe monolayer

Goals:

- performing surface calculations
- using SOC in JuTiBi
- calculating the MCA and the angular moments

files: `inputcard_MCA_Fe_monolayer_x.dat` and `inputcard_MCA_Fe_monolayer_z.dat`

In this exercise you should compute the magneto-crystalline anisotropy energy (MCA) of a free standing (unrelaxed) Fe(001)-monolayer. First you should enter the lattice structure, which should look as follows:

```
ALATBASIS=  2.87d0 1.0d0 1.0d0

BRAVAIS
1.0d0 0.0d0 0.0d0
0.0d0 1.0d0 0.0d0
0.0d0 0.0d0 0.0d0
```

Set the number of k-points to 40000 and be aware to set the number of k-points along the z-direction to 1 and not zero. You should use such a large amount of k-points, because the MCA is very low in energy and therefore we need a very precise numerical description. To calculate the MCA we have to switch on SOC, therefore the full k-mesh has to be used instead of the irreducible part.

```
BZDIVIDE= 200 200 1
KPOIBZ= 40000
IRRBZ= F
```

The SOC parameters have to be entered in the table-like structure explained in section 3.5. A SOC parameter of 0.06 eV in the d-orbitals and 0.18 eV in the p-orbitals[8] is reasonable [16].

```
-----------------|--------------------------------------
ZATOM       #ORB |  s_Orb |  px_Orb    py_Orb    pz_Orb |
45.0d0       9   |    1   |    1          1        1    |
-----------------|--------------------------------------|
SOC-Parameter:   |   ---  |           0.18d0            |
-------------------------------------------------------------
dxy_Orb    dxz_Orb    d_yz_Orb    d(x2-y2)_Orb    d(z2-r2)_Orb |
   1          1          1            1                1        |
-------------------------------------------------------------|
                                 0.06d0                      |
-------------------------------------------------------------
```

To switch on SOC set `SOCLOG=T`. You should use the non-collinear scheme with `NC_IN_UC=T` and `NCMAG=T` to calculate the MCA of a system, because it is much better tested than the case `SET_GAXIS=T` (for more details see section 3.7). Start with some appropriate start-values for the exchange energies (f.ex. 1.1 eV) and switch on self-consistency.

Now to calculate the MCA you have to perform two calculations, one with the magnetic moment along the z-direction and the other with the magnetic moment along the x-direction. To set the magnetic moment along the x-direction use a Θ-angle of 90°:

```
NC_ANGLES
1  90.0d0 0.0d0
```

For the calculation with the moment along the z-direction the converged net-moment, the total energy and the angular moment[9] should be as follows:

```
 ** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
 1   8.00000   0.00000   0.00000   3.60166     0.0000     0.0000

 ###### Energy analysis:
 Band energy in eV:   0.83670
 Double counting corrections in eV:
 LCN:    0.00000
 Stoner model:    3.11161
 Constraint for mag. moments:   0.00000
```

[8] The SOC parameter in the p-orbitals plays a minor role for the MCA.
[9] The angular moment can be found in the file `conv_charges.dat`.

```
**Total energy**:    3.94830

### Angular moments (total , x , y, z) :
# total ang. moment: 0.17849   0.00000   0.00000   0.17849
```

For the calculation along the x-direction the code should yield following results:

```
 ** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
 1   8.00000   3.60182   0.00000   0.00000    90.0000     0.0000

###### Energy analysis:
Band energy in eV:   0.83770
Double counting corrections in eV:
LCN:    0.00000
Stoner model:    3.11189
Constraint for mag. moments:   0.00000

**Total energy**:    3.94959

### Angular moments (total , x , y, z) :
# total ang. moment: 0.13163   0.13163   0.00000   0.00000
```

Finally you should calculate the difference between the two total energies to obtain the MCA, which is about 1.3 meV in this system. Note that the angular moment along the easy axis direction is larger than along the heavy axis as it should be.

Exercises for "experts": For interested users I recommend to calculate the MCA of the $L1_0$-FePt structure in dependence of the lattice constant ratio $\frac{c}{a}$. The results of the JuTiBi code are discussed in my diploma thesis [16]. If you are not sure how to implement different types of basis atoms you should first take a look into the next exercise.

5.6 Using non-collinear magnetism

The final exercises are all about calculations of non-collinear magnetic systems. First you will calculate the magnon dispersion of bcc-Fe, then some calculations in 1-dimensional systems should be performed.

5.6.1 Spin-Spiral calculations (1): bcc Fe

Goals:

- using the gen. Bloch theorem
- calculating magnon dispersions
- learning how to perform force theorem calculations

files: `inputcard_bccFe_force_theorem.dat` and `inputcard_bccFe_self_consistent.dat`

In this exercise you should calculate the magnon dispersion of bcc-Fe. I am assuming that you have done the exercise of the calculation of the bcc-Fe band structure. Therefore entering the lattice structure, preparing a self-consistent calculation etc. should be known. It is enough to use 8000 k-points in the full Brillouinzone, but note that you have to use the full k-mesh instead of the irreducible one. Remember that you have to set `SOCLOG=T` even for a non-collinear calculation without SOC, but you should set the SOC parameters to zero in this case. Switch on `NCMAG` and `BLO_WAVE` and specify a cone-angle for the spin-spiral behind the keyword `BLO_THETA`. I recommend to use 30° in this exercise. Then specify a q-way for the magnon dispersion. In the below example the high-symmetry way $\Gamma \rightarrow N \rightarrow P \rightarrow \Gamma \rightarrow H \rightarrow N$ is used.

```
LOG_Q_WAY=T
NUMQWAYS=5

NUM_QWAY   CREA_QWAY
 10              0.0d0 0.d0 0.d0        0.5d0 0.5d0 0.d0
 10              0.5d0 0.5d0 0.d0       0.5d0 0.5d0 0.5d0
 10              0.5d0 0.5d0 0.5d0      0.d0 0.d0 0.d0
 10              0.d0 0.d0 0.d0         1.0d0 0.0d0 0.0d0
 10              1.0d0 0.0d0 0.0d0      0.5d0 0.5d0 0.0d0
```

If $\Theta \neq 90°$ the Θ-angle can change during the self-consistency[10]. Usually it will change only a little bit for the larger q-values and this is not a problem. However, you should keep in mind that self-consistent spin-spiral calculations with cone-angles $\Theta \neq 90°$ (or no-planar spirals) could lead to problems.

Now run the calculation and plot the 1st column of the file `Eq.dat` against the 6th column, which contains the total energies. Then you should obtain the black curve with the triangles in fig. 5.2 as magnon dispersion. Be aware that this calculation takes some time ($\sim 1h$). If you do not want to spend so much time reduce the number of q-points and/or k-points. The magnon dispersion for a couple of Θ-angles can be observed in my diploma thesis [16].

This calculation has consumed some computational time, because a self-consistent calculation for each q-point had to be performed. However, there is a possibility to

[10]Unfortunately I realized that the Θ-constraint implemented into the JuTiBi code does not work for the cone-angle of a spin-spiral. I have not yet found a possibility to avoid this problem.

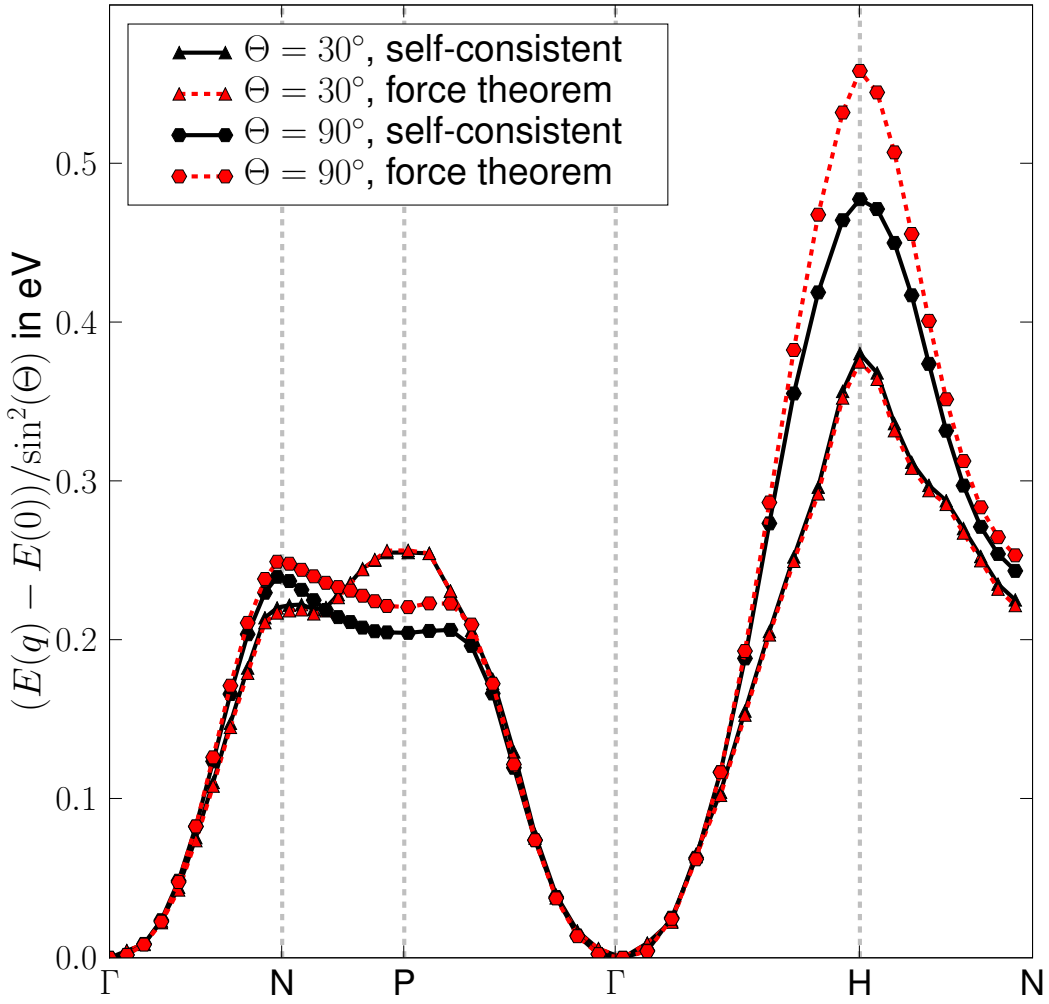


Figure 5.2: Magnon dispersion of bcc-Fe for a cone-angle of 30° and 90°. The red curves are the force theorem calculations, whereas the black curves are the results of the self-consistent calculations. The hexagons display the result of the spin-spiral with a cone-angle of 90°.

spare computational time by exploiting the force theorem (see section 2.9). In this case you have to perform a self-consistent calculation for the ferromagnetic case first and use the converged values for "one-shot calculations" with $\boldsymbol{q} \neq 0$. The converged values for $\boldsymbol{q} = 0$ should be:

```
XC_ENERGY
    1    0.1267807629    0.1267807629    1.2678076287

CHARGE_0
    1    8.0d0    8.0d0
```

Now copy these values to the corresponding positions in the `inputcard`[11] and switch off self-consistency. Therefore only one iteration-step will be performed and you should get the red curve in fig. 5.2 in a short time. As you can observe the force theorem is in a satisfactory agreement with the self-consistent result especially in the region of smaller q-values[12]. Even for a cone-angle of $\Theta = 90°$ the results of the force theorem calculation are in nice agreement compared to the self-consistent result for smaller q-values (see fig. 5.2). Therefore the force theorem approximation can be used for all spin-spiral calculations without remorse.

5.6.2 Spin-Spiral calculations (2): Fe chain

Goals:

- performing calculations for 1-dim. structures
- learning the difference between a non-collinear calculation exploiting the gen. Bloch theorem and performing a calculation in a magnetic super cell
- performing calculations with more than one basis atom

files: `inputcard_genBT` and `inputcard_mag_unitcell`

In the JuTiBi code you can describe spin-spirals within the gen. Bloch theorem, but of course using a description in a magnetic supercell with manually defined magnetic moments for each basis atom is also possible. In the section 2.8 these two methods are explained in detail. Take also a look to the fig. 2.13. In this exercise you should treat a free-standing mono-atomic Fe chain with both methods and test if they are equivalent descriptions for a spin-spiral with a cone angle of 90° and $q = 0.25$[13].

[11] Or use the method with `RESTA_SC=T`.
[12] In this region also the Θ-angle does not change within the self-consistent scheme.
[13] Therefore exactly the case in figure 2.13 should be calculated.

First enter the lattice structure with a lattice constant of $a = 2.22\,\text{Å}$:

```
ALATBASIS=  2.22d0 1.0d0 1.0d0

BRAVAIS
1.0d0 0.0d0 0.0d0
0.0d0 0.0d0 0.0d0
0.0d0 0.0d0 0.0d0
```

Specify the number of k-points along the x-direction to 400 and set it to 1 for the other directions. For an 1-dimensional Fe structure the LDA parameter set yields the better description. Therefore copy the LDA parameter set from the webpage [4] into the file `SKP_input_papakonst`. Increase the cutoff-radius and the cluster radius in the case of an error message like `Increase Cutoff radius of Cluster generation`. Now perform a self-consistent calculation for the q=0.25 case by using the following input in the `inputcard`. Remember that we perform a calculation for a single q-point, therefore switch off `LOG_Q_WAY`.

```
BLO_WAVE=T
BLO_THETA= 90.0d0
BLO_Q= 0.25d0 0.0d0 0.d0
```

The converged values for the magnetic moment and the total energy should be as follows:

```
** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
1   8.00000   2.90316   0.00000   0.00000    90.0000     0.0000

**Total energy**:    4.14186
```

Now you should do the same calculation in the magnetic unit-cell. Therefore you need to define 4 basis atoms and a lattice structure as displayed below:

```
ALATBASIS=  2.22d0 1.0d0 1.0d0

BRAVAIS
4.0d0 0.0d0 0.0d0
0.0d0 0.0d0 0.0d0
0.0d0 0.0d0 0.0d0
```

```
CARTESIAN= F
BASATOMS
  0.00d0  0.d0   0.d0
  0.25d0  0.d0   0.d0
  0.50d0  0.d0   0.d0
  0.75d0  0.d0   0.d0

NUMBASIS=4
```

Note that in this case the basis vectors are defined in Bravais representation! Of course one can also use again a Bravais vector $(1, 0, 0)$ and use a 4-times larger lattice constant a instead. To compare the energies with the calculation performed within the gen. Bloch theorem you should take care to use the same k-point density. Therefore only 100 k-points should be used now. If you work with more than 1 basis atom following entries have to be specified for each of these basis atoms: the table for the specification of the orbitals[14], the atom type (even if `PAPA_BINA=F`), the start values of the Mulliken charge and reference charge, the start values of the exchange energies and at last the angles to define the direction of the magnetic moments (even if `NC_IN_UC=F`). Use the same exchange energies and Mulliken charges for all basis atoms, because they are the same due to symmetry. Only the directions of the magnetic moments should be chosen such that you simulate a spin-spiral with $\Theta = 90°$ and $q = 0.25$, therefore:

```
NC_IN_UC=T
NC_ANGLES
1  90.0d0 0.0d0
2  90.0d0 90.0d0
3  90.0d0 180.0d0
4  90.0d0 270.0d0
```

Remember to switch of `BLO_WAVE`, because we are not exploiting the gen. Bloch theorem in this case. Also take care to use the correct number of electrons per unit cell, which should be 32 in this case!

Now start a self-consistent calculation and you should obtain following converged values for the magnetic moments and the total energy:

```
  ** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
  1   8.00000   2.90316   0.00000   0.00000    90.0000     0.0000
  2   8.00000   0.00000   2.90316   0.00000    90.0000    90.0000
  3   8.00000  -2.90316   0.00000   0.00000    90.0000   180.0000
  4   8.00000   0.00000  -2.90316   0.00000    90.0000   -90.0000

  **Total energy**:   16.56743
```

Of course the total energy has to be divided by 4 and then you can see that both methods are equivalent.

[14]Leave one line free between the tables!

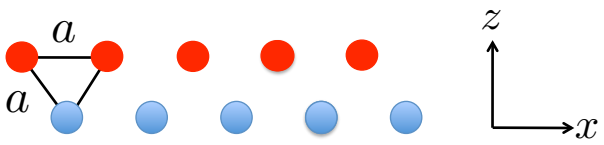


Figure 5.3: Structure of the Fe/Pt chain. The red atoms are the Fe atoms, whereas the blue atoms represent the Pt atoms. The lattice parameter a is 2.22 Å.

5.6.3 Treating SOC in Spin-Spirals: FePt chain

Goals:

- include SOC in a 1st order perturbation theoretical treatment
- performing calculations with different types of basis atoms

files: `inputcard_DMI_Fe_Pt_chain.dat`

In the last exercise you learn how to add SOC in spin-spiral calculations, which can be used to investigate the Dzyaloshinskii-Moriya interaction in non-inversion-symmetric systems. The simplest periodical system with a broken inversion-symmetry is a free standing zig-zag chain consisting of two different atom types. You should use Fe and Pt as atom types. In fig. 5.3 you can extract the structural parameters.

Using two different atom types in the JuTiBi code can be realized by setting the corresponding NRL-TB parameter sets under the keyword `STA_READ` for the first atom type and `STA2_READ` for the second one. In this exercise use the GGA parameter set for Fe and the LDA parameter set for Pt from the webpage [4]. Then specify the atom type of each basis atom under the keyword `TYP_BASAT`. I recommend to use 400 k-points for the calculation. You should exploit the force theorem. Therefore perform a ferromagnetic calculation first, which should give you the following converged moments and charges:

```
** Mulliken charge | m_x | m_y | m_z | Theta | Phi **
1   8.10684   0.00000   0.00000   2.78948     0.0000     0.0000
2   9.89316   0.00000   0.00000   0.56884     0.0000     0.0000
```

Now use this converged values to perform a force theorem calculation for flat spin-spirals[15] with q along $0.0 \rightarrow 0.1$. To switch on the flat-spiral set `FLAT_SPIR=T` and remember to leave the cone angle to zero. We want to treat SOC in 1st order perturbation theory, therefore switch on `SOC_PERTU` and specify the SOC parameters in the tables of the `inputcard`. Use the SOC parameters given in the appendix B. The results of the calculation are displayed in fig. 5.4. The linear behaviour around $q = 0$ is clearly observable. Due to the small energy-scale of the SOC-contribution, the curve is not

[15] These type of spin-spiral has to be used to obtain DMI (see section 2.10).

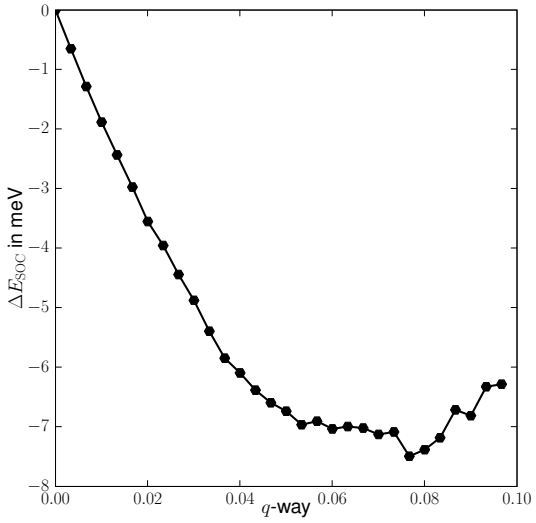


Figure 5.4: The SOC contribution to the energy of the Fe/Pt chain.

perfectly smooth. Using more k-points would lead to a smoother curve as one can see in my diploma thesis [16].

Exercises for "experts": If you feel ready to perform a really lengthy calculation (about 1 day), then try to compare the 1st order SOC treatment in the small chemical unit cell exploiting the gen. Bloch theorem to a full SOC calculation within the magnetic unit cell (not exploting the gen. Bloch theorem, see last exercise). Of course you can not go to arbitrary small q-values, because the necessary magnetic unit cell would become too large, but it is feasible to do a calculation up to $q = 0.0125$, which corresponds to an 160-atomic unit cell. I would recommend to increase the number of k-points to at least 2000 k-points for the calculation in the chemical unit cell to get a nice comparison. The results should show you that in the vicinity of $q = 0$ the slope of the curve should be the same. Take a look into my diploma thesis if you want to see details.

6 Code Structure

First an overview of the code structure is presented in section 6.1, after which a detailed description of each participating subroutine is given in section 6.2.

6.1 Overview of the code structure

In the figure on the right hand side an overview of the code structure is displayed. The preparation steps are discussed more detailed in the subsection 6.1.1. The three do-loops over the q-vectors, the iteration steps and the k-points are at the core of the JuTiBi code. For each q-, k-point and iteration step the Hamiltonian is created and diagonalized to obtain the eigenenergies and eigenvectors, which are needed to calculate the Fermi energy ε_F, the charges and the magnetic moments. These charges and magnetic moments enter into the self-consistency (see fig. 2.9) and if they are converged the total energy of the system is determined. A detailed description of this structural part of the code can be found in the subsection 6.1.2. Finally the density of states, the band structure, the Fermi surface etc. of the converged system can be calculated (see subsection 6.1.3).

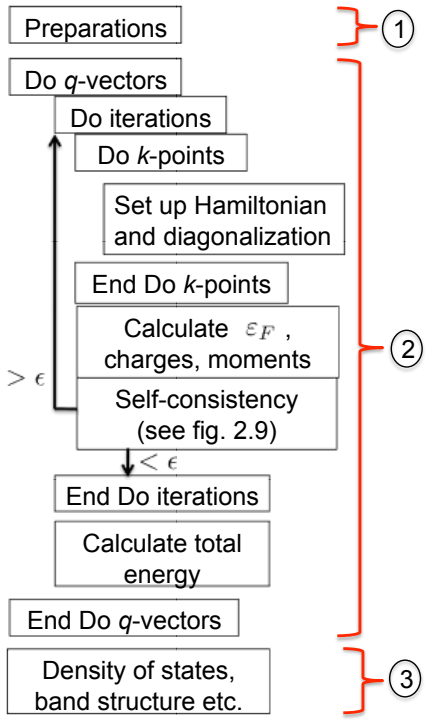


Figure 6.1: Overview of the code structure.

The next sections and subsections of this chapter discuss the code structure in detail. The called subroutines are indicated by the statement "call [name of subroutine]". A detailed description of the subroutines can be found in section 6.2. Together with the list of variables (see appendix E) this should allow to work into the code in a short time to modify it for own purposes etc. Users, who intent to use the code only for calculations, can skip this part.

6.1.1 Preparation

The preparation steps involve the creation of the k-mesh, the construction of the neighbour shells, reading in the charges, magnetic moments, the structure of the Hamiltonian as the Slater-Koster parameters and the creation of the spin-orbit coupling matrix[1].

```
Variable declaration
call readdim
open files
allocate majority of arrays
call lattix_TB

if (dimlattice==2.or.dimlattice==3) then
  call pointgrp
  call findgrp
  call bzirr3d
else if (dimlattice==1) then
  call onedim_kmesh
end if

call rrgen
call clsgen99
if (bondcut) then
  call cut_bondings
end if

call protohamiltonian
if (skpvary) then
  call create_papa_hopping
else
  call SKP_by_hand
end if
call proto_SOC
```

6.1.2 At the heart of the code

In this part of the code the Hamiltonian is set up and is diagonalized for each k- and q-point in each iteration step. Note that the magnetic part of the Hamiltonian is constructed outside of the do-loop over the k-points due to its k-independence. If `soglog=F` the Hamiltonian is diagonalized separately in the majority and minority spin channel, whereas in the case `soglog=T` the Hamiltonian has to be diagonalized in the full spin-space.

[1]Note that $\mathcal{H}_{\mathrm{SOC}}$ is k- and q-independent and does not change within the self-consistency.

After obtaining all eigenvectors and eigenenergies for all k-points the Fermi energy and the charges (see eqs. 2.47, 2.48 and 2.54) can be determined. These charges are used to calculate the magnetic moments and Mulliken-charges of the basis atoms on which the self-consistency is based. If these quantities are converged the total energy and the corresponding moments and charges are computed.

```
DO q-vectors

  DO iteration steps
  :
  :  call create_Hmagnetic
  :
  :  DO k-points
  :  |
  :  |    if (skpvary) then
  :  |      call create_Hk_papa
  :  |    else
  :  |      call create_H0_SKP_by_hand
  :  |    end if
  :  |
  :  |    if (.not.soclog) then
  :  |      call create_H_loc_neut
  :  |      putting together the Hamiltonian for each spin-channel
  :  |      call Hamilton_diag
  :  |      call save_ee_ev
  :  |    else
  :  |      set up H0 and S
  :  |      if (flat_spiral) then
  :  |        construct spin-flip block of H0 and S
  :  |      end if
  :  |      call create_H_loc_neut
  :  |      if (set_gaxis) then
  :  |        call global_spin_rot
  :  |      end if
  :  |      call Hamilton_diag
  :  |      call save_ee_ev
  :  |    end if
  :  |
  :  END DO k-points
  :
  :  call sort_energies
  :  call calc_Fermi_energy
  :  call calc_charges
  :
```

```
  :  call sc_mixing :if converged -|
  :                                |
  END DO iteration steps           |
                                   |
  call calc_final_charges       <-|
  call calc_total_energy

END DO q-vectors
```

6.1.3 Optional calculations

The final part of the code covers the calculation of the density of states and the computation of the band structure of the converged system along a defined high-symmetry k-way. For 3-dimensional systems a file can be created allowing to visualize the Fermi surface with the program XCrysDen (see [50]).

```
if (log_fermi_surface.and.dimlattice==3) then
  call export_data_xcrysden
end if
if (berrycurv=T) then
  call export_data_berrycurv
end if
call calc_DOS
calculate the band structure of the converged system
close files and deallocate arrays
```

6.2 Description of the subroutines

This section lists the most important subroutines used in the JuTiBi code. For each subroutine the input-, input/output- and output-variables are displayed. A description for all these variables can be found in the appendix E. Additionally a usually brief description for each subroutine is presented. For users, who only want to use the code for calculations this chapter is probably a little bit to special. For these users I would only recommend to take a look into the description of the subroutine calc_DOS, because there are described some theoretical aspects, which are not mentioned in chapter 2.

Concerning the description of the input and output variables of the subroutines, I tried to call the variables in the main program the same as in the subroutines. But for a few subroutines the variables are called different. In these cases I display the connection between the variables in writing `variable_sub=variable_main`. In the list E only the variable names as called in the main program are displayed.

Another detail, which could be interesting for users, who intent to modify the code: All subroutines and variables are also explained in the code (usually in the header), but in some cases the explanations are more detailed in this documentation than in the code.

Therefore I recommend to work with both, the documentation and the descriptions in the code, to understand the code.

6.2.1 readdim

Input: This routine has no input variables.
Input/Output: This routine has no input/output variables.
Output: `numbasis`, `npoibz`, `nkxyz`, `irr`, `numendpts`, `numptsway`, `dimkway`, `max_num_cluster`, `numbasis_max`, `max_number_cluster_atoms`, `spinlog`, `soclog`, `skpspin`, `skpvary`, `dimHamilton`, `lmax`, `numelectrons_uc`, `num_energy_DOS`, `lor_width`, `faclor`, `smear_mag`, `bondcut`, `Dosorbat`, `log_spec_orbat`, `spec_at`, `spec_orb`, `log_fermi_surface`, `set_gaxis`, `new_gaxis`, `ncmag`, `nc_in_uc`, `blo_wave`, `tb_sc`, `mix_lin`, `mix_broy`, `mix_alpha`, `max_iter`, `n_init_lin`, `sc_cond`, `stoner_para`, `loc_charge_U`, `restart_sc`, `restart_iteration`, `fermi_broadening`, `log_theta_constraint`, `U_con`, `fat_bands`, `numendpts_qway`, `numpts_qway`, `log_q_way`, `log_qmesh`, `npoiqbz`, `nqxyz`, `irr_q`, `hard_disk`, `soc_pertubation`, `log_change_fermi_energy`, `dos_local`
external routines: Ioinput
Description: This subroutine reads in the majority of the variables of the `inputcard`. Most of them are logical variables and integers. Some of these integers are needed to allocate coming arrays. The most important variables are also displayed as output in the shell.

6.2.2 Ioinput

Input: `CHARKEY`, `ILINE`, `IFILE`
Input/Output: This routine has no input/output variables.
Output: `CHAR`, `IER`
external routines: none
Description: This subroutine is used to find a keyword inside of the first 2000 lines of the `inputcard`. If the keyword `CHARKEY` is found it reads in the first block of characters (except of indents) stored in variable `CHAR`. It reads in the same line if "=" is set directly behind the keyword, or else in the line `ILINE` under the keyword. Including indents the keyword has to consist of 10 characters. The character "=" does not have to be set behind the whole keyword, it can be set directly behind the part of the keyword, behind which are only indents (e.g. keyword is "NUMBASIS ", then NUMBASIS= is allowed). Be careful in reading data in line `ILINE` under the keyword, because data in front of the keyword is not read in. We show some examples for the keyword "NUMBASIS ":

```
NUMBASIS=5          correct
NUMBASIS=   5       correct
NUMBASIS            correct for ILINE=1
   5
NUMBASIS            correct for ILINE=2


   5
```

NUMBASIS wrong!
5

6.2.3 lattix_TB

Input: `IBASIS=numbasis`
Input/Output: This routine has no input/output variables.
Output: `alatc`, `BRAVAIS=abravais`, `RECBV=recbravais`, `basis`, `DIMBRAVAIS=dimlattice`
external routines: Ioinput
Description: In this subroutine the Bravais structure is read in. The lattice parameters, the Bravais vectors and the position of the basis atoms (either in cartesian- or in Bravais-representation) are imported from the `inputcard`. The dimension of the Bravais lattice is determined and the reciprocal Bravais vectors are calculated by inverting the Bravais matrix. Notice that the basis atoms in the array `basis` are given always in cartesian representation regardless which representation was used in the `inputcard`.

6.2.4 pointgrp

Input: This routine has no input variables.
Input/Output: This routine has no input/output variables.
Output: `rotmat`, `rotname`
external routines: none
Description: In this subroutine the rotation matrices of the symmetries of all 32 lattice point groups are defined and named after the convention in [51].

6.2.5 findgroup

Input: `bravais=abravais`, `recbv=recbravais`, `rbasis=basis`, `alat=alatc`, `nbasis=numbasis`, `rsymat=rotmat`, `rotname`
Input/Output: This routine has no input/output variables.
Output: `bravais1=abravais_au`, `recbv1=recbravais_au`, `rbasis1=basis_au`, `abclat=abclatc`, `nsymat=numsym`, `isymindex=symindex`
external routines: latvec
Description: The main function of the subroutine is to determine the symmetries of the lattice structure. Per default basis atoms are considered as non-equivalent for the symmetry determination. To determine the symmetries all 64 symmetry operations of the point groups are applied to the lattice. If the transformed vectors are again lattice vectors the symmetry is a lattice symmetry. To compare the transformed vectors to the lattice vectors a logical function called latvec is used.

In this subroutine the Bravais vectors and basis vectors are multiplied with the lattice parameters a, b and c to obtain the correct symmetries for non-cubic systems. These variables `abravais_au`, `recbravais_au` and `basis_au` are usually given in Å or in a.u. depending on the choice in the `inputcard`. I would recommend to give all length

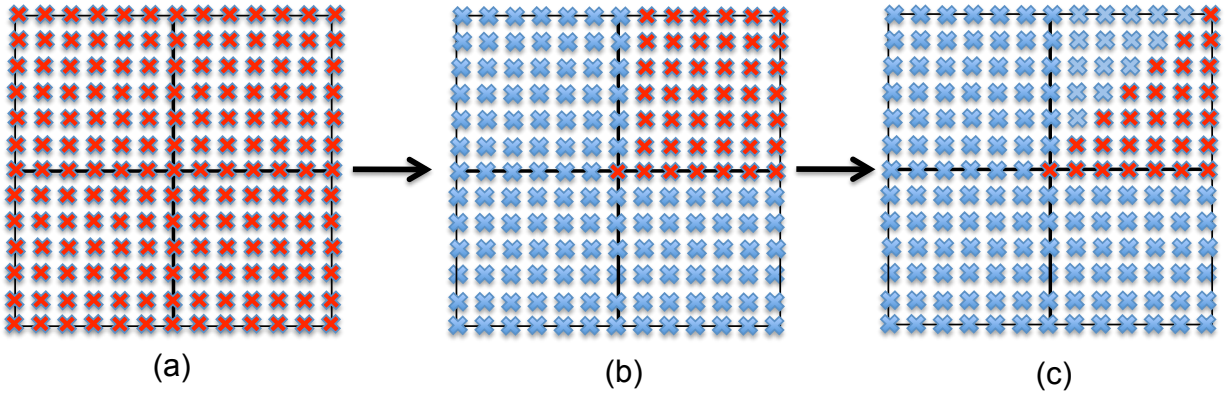


Figure 6.2: This figure should give an idea how the code is constructing an irreducible part of a k-mesh. As an example we choose a simple quadratic lattice. (a) An equidistant k-mesh in the full reciprocal unit cell of the quadratic lattice is constructed. (b) In this case the 4-fold rotation-symmetry leads to the exclusion of all blue k-points, because the red k-points can be mapped onto the blue ones. (c) After applying all 8 symmetry-operations of the quadratic lattice the irreducible part is determined.

in Å, because for the implementation of the NRL-TB parametrization the code is written such that it uses Å as length-unit.

6.2.6 bzirr3d

Input: `kpoibz`=`npoibz`, `nkxyz`, `recbv=recbravais`, `bravais=abravais`, `abclat=abclatc`, `rsymat=rotmat`, `nsymat=numsym`, `isymindex=symindex`, `irr`
Input/Output: This routine has no input/output variables.
Output: `nkp`, `nk=numkmesh`, `kp`, `wtkp`, `volbz`
external routines: cross, ddet33, ddot1, dinit, dinv33, dmpy, dnrm2, scal1, dswap1, errmsg, nrmliz, rotmat, symlat
and some Lapack routines: dcopy, daxpy, idamax
Description: This subroutine and its external routines are only described very briefly in this documentation. More information can be found in the documentation about the SPRKKR code [53], which uses the same routine to create the k-meshs.

First, the subroutine creates an equidistant k-mesh with `nkxyz(:)` k-points along the Bravais vector directions Then it calculates an irreducible part of this k-mesh by applying the symmetry-operations of the lattice to the k-points and leaving only the k-points, which are not reproduced by symmetry-operations (see fig. 6.2). In addition the weights of the k-points are determined, which are useful for integrations over the Brillouin zone. Note that the k-points are given in units of 2π and therefore contain the lattice parameters a, b and c.

6.2.7 onedim_kmesh

Input: `alat=alatc(1)`, `recbv=recbravais`, `bravais=abravais`, `nbasis=numbasis`, `basis`, `npoibz`, `nkx=nkxyz(1)`, `irr`
Input/Output: This routine has no input/output variables.
Output: `bravais1=abravais_au`, `recbv1=recbravais_au`, `basis1=basis_au`, `nsymat=numsym`, `nkp`, `numkmesh`, `kp=kp_1dim`, `wtkp`, `volbz`
external routines: none
Description: This subroutine calculates the symmetries of a 1-dim. lattice (i.e. the inversion-symmetry of the system) and creates an equidistant k-mesh in the Brillouin zone. If `irr=true` the irreducible part of the Brillouin zone is created, therefore half of the BZ if inversion-symmetry is fulfilled. The inversion-symmetry examination assumes the basis atoms as different by default. Note that the k-points are in units of 2π, but the lattice parameter a is contained.

6.2.8 rrgen

Input: `BV=abravais_au`, `abclatc`, `DIMBRAVAIS=dimlattice`, `NRD=max_num_cluster`
Input/Output: This routine has no input/output variables.
Output: `RR=cluster`, `NR=num_cluster`
external routines: dsort, scalpr, vadd, veq, vmul, Ioinput
Description: The routine *rrgen* prepares a cluster containing all Bravais vectors of the lattice inside a sphere with a radius of at least $R_{\max}$ (in units of a). This cluster is used in the routine clsgen99 to determine the neighbour shells for each basis atom. The external routines are not explained in this documentation, but they are also used in the SPRKKR code [53].

6.2.9 clsgen99

Input: `NAEZ=numbasis`, `RR=cluster`, `RBASIS1=basis_au`, `ALAT=abclatc`, `NAEZD=numbasis_max`, `NRD=max_num_cluster`, `NACLSD=max_number_cluster_atoms`
Input/Output: This routine has no input/output variables.
Output: `Z=atomic_number`, `NACLS=num_atom_cluster`, `ATOM=cluster_atom_index`, `EZOA=cluster_atom_bravais`, `RCLS=pos_cluster_atom`, `RCUT=R_cutoff`
external routines: dsort, rinit, Ioinput
Description: This subroutine uses the prepared cluster of Bravais vectors from the subroutine rrgen and includes the basis atoms to obtain neighbour shells for each basis atom. The shells are essential for the description of the hopping elements and on-site energies within the Slater-Koster parametrization and the NRL-TB parametrization. The atoms within a cluster are saved in the array `pos_cluster_atom(:,:,:)` via a certain order: First the atoms are ordered concerning the bonding-distance, then the z-component, y-component and at last the x-component. Figure 6.3 adumbrates the method used to construct the shells.

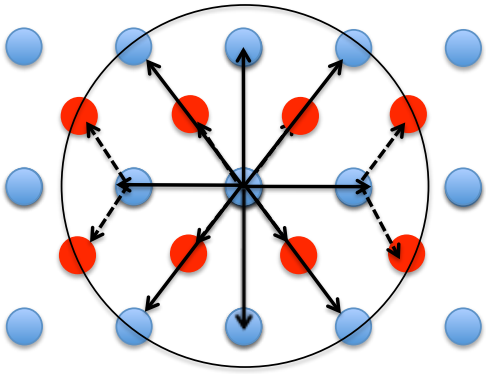


Figure 6.3: This is a simple example for the cluster generation of a rectangular lattice with two basis atoms (red and blue). Within a certain cut-off radius `R_cutoff` a Bravais cluster is generated in the subroutine rrgen. This cluster is indicated with the black solid arrows as Bravais vectors. In the subroutine clsgen99 the full Cluster is created by adding the basis atoms.

6.2.10 cut_bondings

Input: `max_number_cluster_atoms`, `numbasis`
Input/Output: `pos_cluster_atoms`, `num_atom_cluster`
Output: This routine has no output variables.
external routines: none
Description: This optional subroutine is only activated, if the logical `bond_cut` is true. The subroutine is cutting off the bondings, which are specified in the `inputcard` by erasing them from the array `pos_cluster_atom(:,:,:)` and `num_atom_cluster(:)`. It should be remarked, that this subroutine wasn't used frequently, therefore debugging could be necessary.

6.2.11 protohamiltonian

Input:`numbasis`, `max_number_cluster_atoms`, `pos_cluster_atom`, `num_atom_cluster`, `spinlog`, `soclog`, `dimHamilton`, `lmax`, `Stoner_para`, `skpvary`, `max_iter`, `restart_sc`, `restart_iteration`, `nc_mag`, `blo_wave`, `nc_in_uc`, `basis`, `soc_pertubation`
Input/Output: This routine has no input/output variables.
Output: `hamorb`, `hambasis`, `hampos`, `xn`, `yn`, `zn`, `nnearneigh`, `soc_p`, `soc_d`, `soc_p_pert`, `soc_d_pert`, `xcenergy`, `total_moment`, `total_mullikan_charge`, `mullikan_charge_start`, `nc_angles`, `blo_theta`, `blo_q`, `distance`, `ovlap`, `mag_x`, `mag_y`, `mag_z`, `abs_mag_d`, `papa_binary`, `type_baseatom`, `flat_spiral`
external routines: Ioinput, moments_local_to_global
Description: The subroutine protohamiltonian defines the structure of the basis representation of the Hamiltonian (`hamorb(:)`, `hambasis(:)` and `hampos(:,:)`). Some for the future steps necessary logical variables as f. ex. the treatment of an overlap matrix (ovlap) or the usage of a flat spin-spiral are read in. In addition the charges

and magnetic moments for the 1st iteration step are calculated and imported from the `inputcard` or if wished imported from the history of an older self-consistent calculation (`restart_sc=true`). The subroutine moments_local_to_global is used to transform the magnetic moments (m_x, m_y, m_z) from the local frame ($m_x = m_y = 0$ and $m_z = |\vec{m}|$) into the global frame.

6.2.12 moments_local_to_global

Input: `i_iteration`, `numbasis`, `max_iter`, `total_moment`, `nc_angles`, `blo_wave`, `nc_in_uc`, `blo_theta`, `blo_q`, `basis`, `flat_spiral`
Input/Output: This routine has no input/output variables.
Output: `mag_x`, `mag_y`, `mag_z`
external routines: none
Description: This subroutine is used to transform the magnetic moments (m_x, m_y, m_z) from the local frame ($m_x = m_y = 0$ and $m_z = |\vec{m}|$) into the global frame. The transformation distinguishes between the case of a spin-spiral (`blo_wave=true`) and non-collinear magnetism in the unit cell via setting the angles by hand (`nc_in_uc=true` and `nc_angles(:,:,:)`). In the case, where both possibilities are switched on the `nc_angles` play the role of (simple) phase-factors, which can change the cone angle and the rotation angle coming from the spin-spiral q-vector on each atom. In the case of a flat-spiral only Θ enters as phase-factor. If necessary it can be used in each iteration step (`i_iteration`).

6.2.13 SKP_by_hand

Input: `spinlog`, `numbasis`, `ovlap`
Input/Output: This routine has no input/output variables.
Output: `onsite_s`, `onsite_p`, `onsite_d`, `vss_sigma`, `vsp_sigma`, `vpp_sigma`, `vsd_sigma`, `vpd_sigma`, `vdd_sigma`, `vpp_pi`, `vpd_pi`, `vdd_pi`, `vdd_delta`, `oss_sigma`, `osp_sigma`, `opp_sigma`, `osd_sigma`, `opd_sigma`, `odd_sigma`, `opp_pi`, `opd_pi`, `odd_pi`, `odd_delta`, `Sonsite_s`, `Sonsite_p`, `Sonsite_d`, `Svss_sigma`, `Svsp_sigma`, `Svpp_sigma`, `Svsd_sigma`, `Svpd_sigma`, `Svdd_sigma`, `Svpp_pi`, `Svpd_pi`, `Svdd_pi`, `Svdd_delta`, `Soss_sigma`, `Sosp_sigma`, `Sopp_sigma`, `Sosd_sigma`, `Sopd_sigma`, `Sodd_sigma`, `Sopp_pi`, `Sopd_pi`, `Sodd_pi`, `Sodd_delta`
external routines: Ioinput
Description: This subroutine reads in the Slater-Koster parameters (SKPs) for the Hamiltonian and the overlap matrix (if `ovlap=true`). In principle there are two major possibilities to describe the SKPs in the code: The first one is the NRL-TB parametrization and the other one is setting the SKPs by hand. In the case of the NRL-TB parametrization this subroutine is inactive and the subroutine create_papa_hopping takes over the role to import the SKPs. There are different possibilities to set the SKPs by hand, such as using identical atoms (`identatoms=true`, in this case use the `inputcard` for the SKPs) or treating different atom types (`identatoms=false`, in this

case use the file `SKP_input` for the SKPs!). One can read in spin-dependent SKPs ($V_{ll'm}^{\uparrow\rightarrow\downarrow}$ included) in the case `SKP_spin=true` or spin-independent SKPs (`SKP_spin=false`).

6.2.14 create_papa_hopping

Input: `numbasis`, `max_number_cluster_atoms`, `alatc`, `cluster_atom_index`, `distance`, `xn`, `yn`, `zn`, `nnearneigh`, `char_log`, `max_nnear`, `papa_binary`, `type_baseatom`
Input/Output: This routine has no input/output variables.
Output: `papa_hopping`, `papa_onsite`
external routines: papakonst_input
Description: This subroutine calculates all necessary hopping elements up to a cut-off radius within the NRL-TB scheme. First the NRL-TB parameters (93 parameters for each atom type) are imported from the file `SKP_input_papakonst`. Then the distance dependent Slater-Koster parameters are calculated, which are directly used for the Slater-Koster transformations (see table A.1) to obtain all hopping elements (45 types for each bonding). The individual hopping elements are saved in following structure:

1: $s-s$, 2: $s-p_x$, 3: $s-p_y$, 4: $s-p_z$, 5: p_x-p_x, 6: p_y-p_y, 7: p_z-p_z, 8: p_x-p_y, 9: p_x-p_z, 10: p_y-p_z, 11: $s-d_{xy}$, 12: $s-d_{xz}$, 13: $s-d_{yz}$, 14: $s-d_{x^2-y^2}$, 15: $s-d_{z^2}$, 16: p_x-d_{xy}, 17: p_x-d_{xz}, 18: p_x-d_{yz}, 19: $p_x-d_{x^2-y^2}$, 20: $p_x-d_{z^2}$, 21: p_y-d_{xy}, 22: p_y-d_{xz}, 23: p_y-d_{yz}, 24: $p_y-d_{x^2-y^2}$, 25: $p_y-d_{z^2}$, 26: p_z-d_{xy}, 27: p_z-d_{xz}, 28: p_z-d_{yz}, 29: $p_z-d_{x^2-y^2}$, 30: $p_z-d_{z^2}$, 31: $d_{xy}-d_{xy}$, 32: $d_{xy}-d_{xz}$, 33: $d_{xy}-d_{yz}$, 34: $d_{xy}-d_{x^2-y^2}$, 35: $d_{xy}-d_{z^2}$, 36: $d_{yz}-d_{yz}$ (!), 37: $d_{yz}-d_{x^2-y^2}$, 38: $d_{yz}-d_{z^2}$, 39: $d_{xz}-d_{xz}$, 40: $d_{xz}-d_{yz}$, 41: $d_{xz}-d_{y^2-z^2}$, 42: $d_{xz}-d_{z^2}$, 43: $d_{x^2-y^2}-d_{x^2-y^2}$, 44: $d_{x^2-y^2}-d_{z^2}$, 45: $d_{z^2}-d_{z^2}$

In addition the on-site energies are calculated within the NRL-TB scheme. The input-character `char_log` is used to either determine the hopping elements of the Hamiltonian (`char_log='H'`) or to determine the overlap-matrix elements (`char_log='O'`) .

6.2.15 proto_SOC

Input: `numbasis`, `dimHamilton`, `hamorb`, `hambasis`, `soc_p`, `soc_d`, `set_gaxis`, `new_gaxis`
Input/Output: This routine has no input/output variables.
Output: `SOC_matrix`, `L_x`, `L_y`, `L_z`, `rot_spinonian`, `rot_spinonian_inv`
external routines: none
Description: This subroutine creates the angular moment operators in atomic orbital representation and uses them to construct the k-independent spin-orbit coupling matrix. In the case `set_gaxis=true` additionally the spin-rotation matrices for a global spin rotation are calculated. The structure of the rotation matrix is as follows:

$$\begin{pmatrix} \cos\left(\frac{\phi}{2}\right) - \mathrm{i}n_z \cdot \sin\left(\frac{\phi}{2}\right) & -\sin\left(\frac{\phi}{2}\right)[n_y + \mathrm{i}n_x] \\ \sin\left(\frac{\phi}{2}\right)[n_y - \mathrm{i}n_x] & \cos\left(\frac{\phi}{2}\right) + \mathrm{i}n_z \cdot \sin\left(\frac{\phi}{2}\right) \end{pmatrix}$$

with ϕ the rotation angle and $\boldsymbol{n} = (n_x, n_y, n_z)$ the normalized vector pointing along the rotation axis to obtain the global spin rotation. These values are calculated automatically in the code. The user has to enter only the new direction for the magnetic

moments. These rotation matrices act on the magnetic part of the Hamiltonian before the diagonalization.

Note that a global spin rotation can be also adjusted by setting the angles `nc_angles` in the case `nc_in_uc=true`. I would recommend to set the angles by hand within `nc_in_uc=true` instead of using `set_gaxis=true` to realize a global spin rotation, because the first method is much more tested. The global spin rotation within `set_gaxis=true` can not be used together with `nc_in_uc=true`!

6.2.16 create_q_way

Input: `dimlattice`, `numendpts`=`numendpts_qway`, `numptsway`=`numpts_qway`, `dimkway`=`dimqway`
Input/Output: This routine has no input/output variables.
Output: `kway`=`qway`, `kway_param`=`qway_param`
external routines: scalpr
Description: This subroutine creates q-vectors along a defined way in the Brillouin zone. Important to note is that the q-vectors are in units of 2π and therefore contain the lattice parameters a, b and c.

6.2.17 create_Hmagnetic

Input: `dimHamilton`, `numbasis`, `hamorb`, `hambasis`, `nc_in_uc`, `blo_wave`, `ncmag`, `nc_angles`, `i_iteration`, `max_iter`, `stoner_para`, `mag_x`, `mag_y`, `mag_z`, `total_moment`, `type_baseatom`, `log_theta_constraint`, `U_con`, `blo_theta`, `blo_q`, `basis`, `flat_spiral`
Input/Output: This routine has no input/output variables.
Output: `H_mag`
external routines: moments_local_to_global
Description: These subroutine creates the magnetic part of the Hamiltonian `H_mag`, which is the only part (beside `H_SOC`) in the NRL-TB scheme, in which the spin-blocks are not the same. The magnetic Hamiltonian is described by the Stoner model (see section 2.6), which can be simply extended to non-collinear magnetism (see section 2.8). `H_mag` is always in the representation of the global spin frame, which makes it necessary to take out the q-dependent spin-rotation in the case of `blo_wave=true` with the help of the subroutine moments_local_to_global, because it is already contained in the gen. Bloch theorem. But notice that the eigenvectors have to be rotated with the q-dependent spin-spiral angles after the diagonalization to obtain the correct directions for the magnetic moments (see appendix A.2). It is possible to treat a spin-spiral system with additional phase factors for the mag. moments of the basis atoms by switching on `blo_wave` and `nc_in_uc`. In this case the `nc_angles` play the role of the additional phase factors.

Additionally a constraint to fix the Θ-angle of the magnetic moments is included in `H_mag`, if `log_theta_constraint=true` (see section 2.8).

6.2.18 create_H0_SKP_by_hand

Input: numbasis, lmax, dimHamilton, max_number_cluster_atoms, pos_cluster_atom, alatc, cluster_atom_index, hamorb, hambasis, hampos, xn, yn, zn, nnearneigh, kpoint, onsite_s, onsite_p, onsite_d, vss_sigma, vsp_sigma, vpp_sigma, vsd_sigma, vpd_sigma, vdd_sigma, vpp_pi, vpd_pi, vdd_pi, vdd_delta, Sonsite_s, Sonsite_p, Sonsite_d, Svss_sigma, Svsp_sigma, Svpp_sigma, Svsd_sigma, Svpd_sigma, Svdd_sigma, Svpp_pi, Svpd_pi, Svdd_pi, Svdd_delta, blo_wave, flat_spiral, blo_q, spinlog, skpspin, ispin
Input/Output: This routine has no input/output variables.
Output: Hamiltonian (respectively Overlapian)
external routines: scalpr
Description: This subroutine creates a certain spin-block (depending on ispin; ispin=1: $\mathcal{H}_0^{\uparrow\uparrow}$, ispin=2: $\mathcal{H}_0^{\downarrow\downarrow}$, ispin=3: $\mathcal{H}_0^{\uparrow\downarrow}$ in case flat_spiral=true) of the k-dependent Hamiltonian $\mathcal{H}_0$ (and the overlap matrix, if ovlap=true) within the Slater-Koster scheme. The (spindependent) Slater-Koster parameters, which are imported in the routine SKP_by_hand are used in the Slater-Koster transformations (see table A.1) to calculate the matrix elements of the real space Hamiltonian for arbitrary bonding directions. The translational symmetry of the system is taken into account by calculating $\mathcal{H}_0(\boldsymbol{k})$ by summing up over all Bravais vectors (Bloch Theorem, see eq. 2.9). For a spin-spiral calculation additional spin-dependent phase-factors are considered (see eqs. 2.79-2.81 or eqs. A.12 in the case of a flat-spiral).

6.2.19 create_Hk_papa

Input: numbasis, lmax, dimHamilton, max_number_cluster_atoms, max_nnear, pos_cluster_atom, alatc, cluster_atom_index, hamorb, hambasis, hampos, nnearneigh, kpoint, papa_hopping, papa_onsite, char_log, blo_wave, flat_spiral, blo_q, ispin
Input/Output: This routine has no input/output variables.
Output: Hamiltonian (respectively Overlapian)
external routines: scalpr
Description: In principle this subroutine has the same function as create_H0_SKP_by_hand. Therefore it creates certain spin-blocks of the Hamiltonian $\mathcal{H}_0$ (respectively the overlap matrix) depending on ispin. But this routine uses the hopping parameters papa_hopping(:,:,:) and on-site parameters papa_onsite(:,:) created in the subroutine create_papa_hopping within the NRL-TB parametrization.

6.2.20 create_H_loc_neut

Input: dimH_loc, numbasis, dimHamilton, hambasis, loc_charge_U, Overlapian, total_mullikan_charge, mullikan_charge_start, i_iteration, max_iter, type_baseatom
Input/Output: This routine has no input/output variables.
Output: H_loc_charge_neutr

external routines: none
Description: This subroutine creates the local charge neutrality part of the Hamiltonian $\mathcal{H}_{\mathrm{LCN}}$ via eq. 2.44. Due to the overlap matrix the local charge neutrality part is k-dependent, therefore it is located within the do-loop over the k-points. The local charge neutrality keeps the Mulliken-charges `total_mullikan_charge(:,:)` in a small region around the desired charges `mullikan_charge_start(:)`. The larger the local charge neutrality constants `loc_charge_U`, the more strict the constraint.

6.2.21 global_spin_rot

Input: `dimHamilton`, `rot_spinonian`, `rot_spinonian_inv`
Input/Output: `Hamiltonian`, `Overlapian`
Output: This routine has no pure output.
external routines: ZHEMM
Description: This subroutine rotates all magnetic moments of the Hamiltonian (without $\mathcal{H}_{\mathrm{SOC}}$) into the desired direction `new_gaxis(:)`, which is contained in the rotation matrix `rot_spinonian`. Important to note is that the global spin rotation does not work together with non-collinear magnetism, therefore all spins have to point along the z-direction. If a global rotation is wished for a non-collinear system put it into the individual angles of the magnetic moments in the inputcard under the keyword `"NC_ANGLES"` and switch on `nc_in_uc=true`.

6.2.22 Hamilton_diag

Input: `dimHamilton`, `ovlap`
Input/Output: `Hamiltonian`, `Overlapian`
Output: `bandenergies`
external routines: ZHEEV, ZHEGV
Description: This subroutine diagonalizes the Hamiltonian. In the case of `soclog=false` the spin-blocks are separately diagonalized for `ispin=1` and `ispin=2` to spare computational time. The eigenvectors are stored in the Hamiltonian, therefore it is overwritten. Notice that also the overlap matrix is overwritten. The eigenvalues are saved in the array `bandenergies`.

6.2.23 rotate_EV

Input: `dimHamilton`, `numbasis`, `basis`, `hambasis`, `blo_q`, `abclatc`, `flat_spiral`
Input/Output: `Hamiltonian_wS`
Output: This routine has no pure output variables.
external routines: scalpr
Description: This routine rotates the eigenvectors in spin-space with the q-dependent spiral-angles to obtain the correct directions for the magnetic moments in the global representation. A detailed derivation of this necessity can be found in A.2. In the case `blo_wave=false` this routine is inactive.

6.2.24 save_ee_ev and save_ee_ev_on_harddisk

Input: `soclog`, `dimlattice`, `ikmesh`, `nkxyz`, `numkmesh`, `volbz`, `wtkp`, `ispin`, `dimHamilton`, `bandenergies`, `Hamiltonian`, `Overlapian_Save`, `numelectrons_uc`, `numbasis`, `basis`, `hambasis`, `blo_q`, `abclatc`, `flat_spiral`, `dim_kmesh_ev`, some running variables necessary for saving the arrays
Input/Output: This subroutine has no input/output variables.
Output: `kmesh_energies`, `numelectrons`, `kmesh_eigenvectors` and `kmesh_eigenvectors_mod` (for `hard_disk=false`)
external routines: ZHEMM, rotate_EV
Description: These subroutines save the eigenenergies into the array `kmesh_energies` and the eigenvectors depending on the logical `hard_disk` into the array `kmesh_eigenvectors` (and the modified EV $\boldsymbol{c}' = \underline{\boldsymbol{S}} \cdot \boldsymbol{c}$ of eq. 2.32 into the array `kmesh_eigenvectors_mod`) or onto the hard disk. The eigenenergies and eigenvalues are saved in a special order depending on the logical variables `spinlog` and `soclog`. These order-structures should be explained in detail here:

In the case of `spinlog=true` and `soclog=false` the eigenenergies and eigenvectors are saved in a order according to the spin. Of course this structure can not be used for a Hamiltonian with SOC. In the case of a non-magnetic treatment, therefore `spinlog=false`, there are no spin-channels any more. The eigenvectors are saved column-wise, therefore in the case of `soclog=false` the array containing the eigenvectors has `dimHamilton` rows, whereas in the case of `soclog=true` the array has $2 \cdot$ `dimHamilton` rows (the first `dimHamilton` entries are the $\uparrow$-contributions to the EV and the last `dimHamilton` entries are the $\downarrow$-contributions). The following cases show how the eigenenergies (and eigenvectors) of different k-points and bands are sorted along the row.

(a) `spinlog=true`, `soclog=false`

$$\Big(\underbrace{\uparrow}_{\text{numkmesh}\cdot\text{dimHamilton}} \Big| \underbrace{\downarrow}_{\text{numkmesh}\cdot\text{dimHamilton}} \Big)$$

(b) `spinlog=true`, `soclog=true`

$$\Big(\underbrace{\qquad}_{2\cdot\text{numkmesh}\cdot\text{dimHamilton}} \Big)$$

(c) `spinlog=false`

$$\Big(\underbrace{\qquad}_{\text{numkmesh}\cdot\text{dimHamilton}} \Big)$$

If we go deeper into the structure, the order mechanism is the same for all three cases. The eigenenergies and eigenvectors are ordered first according to the k-points itself, then according to the band index and at last according to the degeneracies of the k-point in the BZ (if `irr=true`, otherwise the degeneracy of a k-point is always 1). Saving the eigenvectors and eigenenergies multiple according to the degeneracies is a possibility to

incorporate the weights of the k-points. The structure of the array has the following form:

$$\left(..., \underbrace{\varepsilon^n_{\boldsymbol{k}_1}, \varepsilon^n_{\boldsymbol{k}_1}, \varepsilon^n_{\boldsymbol{k}_1}, \varepsilon^n_{\boldsymbol{k}_1}}_{\text{degeneracy of } \boldsymbol{k_1}}, \underbrace{\varepsilon^{n+1}_{\boldsymbol{k}_1}, \varepsilon^{n+1}_{\boldsymbol{k}_1}, \varepsilon^{n+1}_{\boldsymbol{k}_1}, \varepsilon^{n+1}_{\boldsymbol{k}_1}}_{\text{degeneracy of } \boldsymbol{k_1}}, ..., \underbrace{\varepsilon^m_{\boldsymbol{k}_2}, \varepsilon^m_{\boldsymbol{k}_2}}_{\text{degeneracy of } \boldsymbol{k_2}}, ... \right)$$

Additionally the subroutine rotates the eigenvectors with the routine rotate_EV before saving them in the case of a spin-spiral calculation. This is necessary to obtain the correct directions for the magnetic moments for all basis atoms in the unit cell (see A.2).

6.2.25 sort_energies

Input: `dimW` ,`W=kmesh_energies`
Input/Output: This routine has no input/output variables.
Output: `IND=index_kmesh_energies`
external routines: none
Description: This subroutine sorts the array `kmesh_energies` containing all eigenenergies from the smallest to the largest values. Notice that not the array itself is sorted, but the positions in the array are sorted, because this is much faster for larger arrays. We should give a simple example:

For the array $W = (1.5, 0.7, 3.8)$ the routine would calculate: `IND=(2,1,3)`.

6.2.26 calc_Fermi_energy

Input: `dim_kmesh_ev`, `numkmesh`, `kmesh_energies`, `fermi_broadening`, `numelectrons_uc`
Input/Output: `fermi_energy`
Output: This routine has no pure output.
external routines: none
Description: This subroutine calculates the Fermi energy within a Newton method by using the equation $N_e = \sum_{\boldsymbol{k},n} f(\varepsilon_F, \varepsilon_{\boldsymbol{k},n})$ (see eq. 2.27). As starting point for the Newton method a roughly determined Fermi energy by filling up the electrons from the lowest energetic states to the larger ones is used.

6.2.27 get_eigenvectors

Input: `dim_eigenvec`, `kmesh_eigenvectors`, `kmesh_eigenvectors_mod`, `soclog`, `hard_disk`, `ii`
Input/Output: This routine has no input/output variables.
Output: `eigenvectors`, `eigenvectors_mod`
external routines: none
Description: This subroutine writes out the `ii`-th eigenvector of the large array `kmesh_eigenvectors` (and `kmesh_eigenvectors_mod`) into the array eigenvectors in the case

hard_disk=false. In the case hard_disk=true it is written out from the unformatted files eigenvector.unformatted and eigenvectors_mod.unformatted.

6.2.28 calc_charges

Input: dimHamilton, numkmesh, lmax, numbasis, dim_eigenvec, dim_kmesh_ev, hamorb, hambasis, soclog, spinlog, hard_disk, fermi_dirac, kmesh_eigenvectors, kmesh_eigenvectors_mod
Input/Output: This routine has no input/output variables.
Output: charge, mullikan_charge, cross_charge, mullikan_cross_charge
external routines: get_eigenvectors
Description: This subroutine calculates the matrix elements of the density matrix (see eq. 2.47 and 2.48). These elements are needed to obtain the charges and magnetic moments of the system. Not only the net-charges are calculated, but also the Mulliken-charges (see eqs. 2.54) of the system.

6.2.29 sc_mixing

Input: numbasis, lmax, numkmesh, dim_sc, mix_lin, mix_broy, ncmag, i_iteration, max_iter, mix_alpha, n_init_lin, stoner_para, type_baseatom, blo_q, blo_theta, abclatc, blo_wave, flat_spiral, charge, mullikan_charge, cross_charge
Input/Output: This routine has no input/output variables.
Output: total_mullikan_charge, total_moment, mag_x, mag_y, mag_z, abs_mag_d, total_mullikan_moment, xc_energy, nc_angles
external routines: broyden, scalpr
Description: This subroutine calculates all interesting moments (net-moments, Mulliken-moments, angles of the mag. moments etc.) with the help of the obtained charges in the routine calc_charges. The net-moments for the *d*-orbitals (H_mag!) and the Mulliken-charges of the atoms (H_LCN!) are the necessary variables in the self-consistency (see figure 2.9). This routine mixes the old variables with the new ones, either within linear mixing (mix_lin=true) or within Broyden mixing (mix_broy=true). For the Broyden mixing a routine broyden, based on the paper by D. D. Johnson (see [49]), is used.

6.2.30 calc_final_charges

Input: numbasis, lmax, numkmesh, dimHamilton, dim_eigenvec, dim_kmesh_ev, kmesh_eigenvectors, kmesh_eigenvectors_mod, hard_disk, soclog, L_x, L_y, L_z, fermi_dirac, max_iter, hambasis, charge, mullikan_charge, cross_charge, mullikan_cross_charge, blo_q, abclatc, nc_angles, mag_x, mag_y, mag_z, abs_mag_d, total_mullikan_charge, last_it, scal_md_ms, scal_md_mp
Input/Output: This routine has no input/output variables.
Output: This routine has no output.
external routines: get_eigenvectors
Description: This subroutine writes out the net-charges, Mulliken-charges, net-magnetic

moments, Mulliken-magnetic moments and the angular moments of the converged systems into the file `conv_charges.dat`.

6.2.31 calc_total_energy

Input: `blo_wave`, `nc_in_uc`, `soclog`, `log_theta_constraint`, `soc_pertubation`, `hard_disk`, `log_change_fermi_energy`, `numbasis`, `dim_kmesh_ev`, `dim_eigenvec`, `last_it`, `max_iter`, `numkmesh`, `dimHamilton`, `dimqway`, `hamorb`, `hambasis`, `hampos`, `soc_p_pert`, `soc_d_pert`, `blo_q`, `abclatc`, `qway_param`, `fermi_energy`, `numelectrons`, `fermi_broadening`, `kmesh_eigenvectors`, `kmesh_eigenvectors_mod`, `kmesh_energies`, `type_baseatom`, `scal_ms_ms`, `scal_md_mp`, `fermi_dirac`, `stoner_para`, `loc_charge_U`, `total_mullikan_charges`, `mullikan_charge_start`, `mag_z`, `abs_mag_d`, `nc_angles`, `blo_theta`, `U_con`, `i_q_vec`, `i_q_vec_max`, `log_q_way`, `log_qmesh`, `lmax`, `abravais_au`, `flat_spiral`
Input/Output: This routine has no input/output variables.
Output: This routine has no output.
external routines: get_eigenvectors, calc_Fermi_energy, proto_SOC
Description: This subroutine calculates the band energy (eq. 2.34), the double countings (eqs. 2.42 and 2.58) and the total energy of the system. In the case of calculating the magnon dispersion of a spin-spiral the relevant energies are stored in `Eq.dat`. In addition the SOC-contribution in 1st order is calculated for spin-spirals if `soc_pertubation=true` and saved into the file `Eq_1storder_SOC.dat`. The equations 2.82, 2.83, 2.84 and 2.88 are incorporated. Therefore also the fermi energy can be recalculated after adding the 1st order SOC contribution by setting `log_change_fermi_energy=true`. The layer- and orbital-resolved analysis of the SOC-contributions are saved in the files `Eq_1storder_SOC_layer_resolved.dat` and `Eq_1storder_SOC_layer_orbital_resolved.dat`.

6.2.32 create_input_Jij

Input: `qp=blo_q`, `blo_theta`, `total_energy`, `abs_mag_d`, `i_q_vec`, `nqp=i_q_vec_max`, `numbasis`, `abravais_au`, `flat_spiral`
Input/Output: This routine has no input/output variables.
Output: This routine has no output.
external routines: none
Description: This subroutine creates the file `jenerg.dat` as input for the python script `plot_Jij.py` programmed by D. Bauer. This program determines the Heisenberg exchange-coupling parameters J_{ij} and the DM-constant D of the system by linear least square fits (see section 2.10). These parameters can be for example used in thermodynamic spin-dynamic simulations [54, 55].

6.2.33 export_data_xcrysden

Input: `kmesh_energies`, `fermi_energy`, `nkxyz`, `recbravais_au`, `alatc`, `dimHamilton`, `numkmesh`, `spinlog`
Input/Output: This routine has no input/output variables.

Output: This routine has no output.
external routines: none
Description: This subroutine creates the file `fermi_surface.bxsf`, which can be exported into the external program XCrysDen to determine the Fermi surface of the system. Details about the structure of this file can be found in the internet [50]. The program is not able to treat 2-dim systems and the calculation of the k-mesh in the irreducible BZ, yet. At least the last point can be extended easily for later purposes.

6.2.34 export_data_berrycurv

Input: `numbasis`, `alatc`, `abravais`, `type_baseatom`, `basis`, `stoner_para`, `soc_p`, `soc_d`, `loc_charge_U`, `fermi_energy`, `dimHamilton`, `lmax`, `hamorb`, `hambasis`, `hampos`, `blo_theta`,`blo_q`, `max_iter`, `last_it`, `ncmag`, `blo_wave`, `nc_in_uc`, `flat_spiral`, `log_theta_constraint`,`U_con`, `mullikan_charge_start`, `set_gaxis`, `new_gaxis`, `max_number_cluster_atoms`, `cluster_atom_index`, `nnearneigh`, `ovlap`, `pos_cluster_atom`, `max_nnear`, `papa_onsite`, `papa_onsite_ov`, `papa_hopping`,`papa_hopping_ov`, `total_moment`, `nc_angles`, `mag_x`, `mag_y`, `mag_z`, `total_mullikan_charge`
Input/Output: This routine has no input/output variables.
Output: This routine has no output.
external routines: none
Description: This subroutine writes out all necessary data into three files (`basics.berrycurv`, `hopping.berrycurv` and `charges.berrycurv`) to reconstruct the Hamiltonian depending on a k-point. It is in particular designed to fit to the code of H. Zhang, capable to calculate the Berry curvature for a tight-binding system. To activate this subroutine, different to other cases not the `inputcard` is used, but rather a file `inp.berrycurv` has to be provided (write `T` at the beginning of the file) in the input-folder. Works only for `soclog=T` and `skpvary=T`!

6.2.35 transform_DOS_local

Input: `nc_in_uc`, `blo_wave`, `flat_spiral`, `numbasis`, `nc_angles`, `blo_theta`, `blo_q`, `basis_au`, `dimHamilton`, `hambasis`
Input/Output: `eigenvectors`, `eigenvectors_mod`
Output: This routine has no output.
external routines: none
Description: This subroutine rotates the eigenvectors from the global spin-frame into the local spin-frame. This is needed to obtain the partial DOS in the local spin-frame. The structure of the spin-rotation matrices is the same than in the subroutine rotate_EV.

6.2.36 calc_DOS

Input: `nc_in_uc`, `blo_wave`, `flat_spiral`, `numbasis`, `nc_angles`, `blo_theta`, `blo_q`, `basis_au`, `dimHamilton`, `hambasis`, `dim_kmesh_ev`, `dim_eigenvec`, `spinlog`, `soclog`, `num_energy_DOS`, `faclor`, `lor_width`, `kmesh_energies`, `index_kmesh_energies`, `hamorb`,

`hampos`, `spec_at`, `spec_orb`, `lmax`, `numkmesh`, `kmesh_eigenvectors`, `kmesh_eigenvectors_mod`, `dos_local`, `dosorbat`, `log_spec_orbat`, `hard_disk`
Input/Output: This routine has no input/output variables.
Output: This routine has no output.
external routines: transform_DOS_local, get_eigenvectors
Description: This subroutine calculates the total DOS and the partial DOS (if `dosorbat=true`) of the system and stores them into files. We should give a detailed analysis of the files:

`spinlog=false` - Total DOS saved in `Total_DOS_up.dat` and the partial DOS in `DOS_atom_orbital_resolved.dat`

`spinlog=true`, `soclog=false` - Total DOS saved depending on the spin-channel in `Total_DOS_up.dat` and `Total_DOS_down.dat`, the partial DOS in `DOS_atom_orbital_resolved_up.dat` and `DOS_atom_orbital_resolved_down.dat`

`soclog=true` - Total DOS saved in `SOC_Total_DOS.dat`, and the partial DOS depending on the spin-channel in `SOC_DOS_atom_orbital_resolved_up.dat` and `SOC_DOS_atom_orbital_resolved_down.dat`

In addition there is the possibility to calculate the DOS in the local spin-frame by setting `dos_local=true`. This is realized via the subroutine transform_DOS_local. The topic of the DOS-calculation is treated very shortly in the theory-part of this documentation. It is helpful to give some theoretical details about the implementation in the code:

As indicated in eq. 2.30 and 2.31 the (partial) DOS is calculated via lorentzian functions of the form

$$l(\varepsilon, \varepsilon_{\boldsymbol{k},n}) = \frac{1}{\pi} \cdot \frac{\Gamma}{\Gamma^2 + (\varepsilon - \varepsilon_{\boldsymbol{k},n})^2} \,.$$

Note that the width Γ is defined as half width of the lorentzian function at half of the maximal value (i.e. half of the full half-width maximum value (FHWM)). In the `inputcard` the variable `lor_width` is defined as the FHWM of the lorentzian function! Therefore it holds: `lor_width`$=2\Gamma = w$.

It is numerically efficient to limit the number of contributing eigenvalues $\varepsilon_{\boldsymbol{k},n}$ in the sum 2.30 in the numerical determination of the density of states $D(\varepsilon)$. The eigenvalues, which are energetically far way from the energy ε can be neglected. This is realized by setting an input-value `faclor`, referred to as α in the following, determining the range to $2\alpha \cdot w$ symmetric around ε in which the sum will be executed. Of course it is also possible to fix the energy $\varepsilon_{\boldsymbol{k},n}$ and to determine the relevant region of energies ε for which $D(\varepsilon)$ is evaluated. This is the method how it is realized in the code. But before we are going into more details, let us first say more about creating the energy-mesh for the DOS:

$$\varepsilon_i = (\varepsilon^{\min}_{\boldsymbol{k},n} - \alpha \cdot w) + (B + 2\alpha \cdot w) \cdot \frac{i-1}{N_\varepsilon - 1} \qquad \text{for} \quad 1 \leq i \leq N_\varepsilon \,,$$

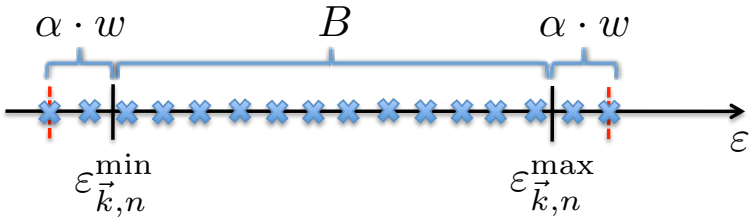


Figure 6.4: This figure displays the within the code constructed energy-mesh for the DOS-calculation. The set is chosen in a way that the start point $\min_{\boldsymbol{k},n}(\varepsilon_{\boldsymbol{k},n}) - \alpha \cdot w$ and the end point $\max_{\boldsymbol{k},n}(\varepsilon_{\boldsymbol{k},n}) + \alpha \cdot w$ are contained in the energy-mesh.

with N_ε=`num_energy_DOS` the number of energy points in the equidistant energy-mesh, $B = \max_{\boldsymbol{k},n}(\varepsilon_{\boldsymbol{k},n}) - \min_{\boldsymbol{k},n}(\varepsilon_{\boldsymbol{k},n})$ the bandwidth of the eigenenergy-spectrum and $\min_{\boldsymbol{k},n}(\varepsilon_{\boldsymbol{k},n}) = \varepsilon^{\min}_{\boldsymbol{k},n}$. In figure 6.4 the constructed energy-mesh is displayed.

Now to determine the relevant region of the energy-mesh of $2\alpha \cdot w$ around $\varepsilon_{\boldsymbol{k},n}$, first the code will determine the $\bar{i}$ for which $\varepsilon_{\bar{i}}$ is the nearest energy to $\varepsilon_{\boldsymbol{k},n}$:

$$\begin{aligned} \varepsilon_{\boldsymbol{k},n} &= (\varepsilon^{\min}_{\boldsymbol{k},n} - \alpha \cdot w) + (B + 2\alpha \cdot w) \cdot \frac{x-1}{N_\varepsilon - 1} \\ \Leftrightarrow x - 1 &= \frac{N_\varepsilon - 1}{B + 2\alpha \cdot w} \cdot \left(\varepsilon_{\boldsymbol{k},n} - (\varepsilon^{\min}_{\boldsymbol{k},n} - \alpha \cdot w)\right) \\ \Rightarrow \quad \bar{i} &= \mathrm{nint}(x+1)\,, \end{aligned}$$

therefore

$$\bar{i} = \mathrm{nint}\left(\frac{N_\varepsilon - 1}{B + 2\alpha \cdot w} \cdot \left(\varepsilon_{\boldsymbol{k},n} - (\varepsilon^{\min}_{\boldsymbol{k},n} - \alpha \cdot w)\right) + 1\right)\,.$$

Now determine the range Δi of the running index for which $\varepsilon \in [\varepsilon_{\boldsymbol{k},n} - \alpha \cdot w, \varepsilon_{\boldsymbol{k},n} + \alpha \cdot w]$:

$$\begin{aligned} \alpha \cdot w &= \frac{B + 2\alpha \cdot w}{N_\varepsilon - 1} \cdot \Delta x \\ \Leftrightarrow \Delta x &= \frac{N_\varepsilon - 1}{B + 2\alpha \cdot w} \cdot \alpha \cdot w \\ \Rightarrow \Delta i &= \mathrm{nint}\left(\frac{N_\varepsilon - 1}{B + 2\alpha \cdot w} \cdot \alpha \cdot w\right)\,. \end{aligned} \tag{6.1}$$

To conclude the code is considering only the mesh-energies with the running indices $i \in [\bar{i} - \Delta i, \bar{i} + \Delta i]$ to add contributions coming from the eigenenergy $\varepsilon_{\boldsymbol{k},n}$ to the DOS.

6.2.37 create_k_way

Input: `dimlattice`, `numendpts`, `numptsway`, `dimkway`
Input/Output: This routine has no input/output variables.
Output: `kway`, `kway_param`

external routines: scalpr
Description: This subroutine creates k-vectors along a defined way in the Brillouin zone. Important to note is that the k-vectors are in units of 2π and therefore contain the lattice parameters a, b and c.

7 Outlook and Improvements

In the last chapter of this documentation I want to propose some future improvements of the JuTiBi code. By doing this I hope to give some ideas to the users, which are motivated to improve the code. Of course I am also interested in your ideas how to improve the code. Therefore you can send constructive suggestions to t.schena@fz-juelich.de. Here is the list of proposed improvements:

I think really useful could be a MPI-parallelization in the k-points, in particular for the investigation of such small quantities as DMI or MCA, which need a lot of k-points to converge. Beside the useful speed up of the code, also the memory limitations concerning the number of k-points can be lowered. However, for users who want to modify the code to treat very large supercell structures this improvement yields no big advantage.

Another big issue could be the implementation of the tetrahedron method [56] to improve the numerical quality of the code and to allow an accurate determination of Fermi surfaces. There is a recently developed code by B. Zimmermann, which contains the tetrahedron method for 2- and 3-dimensional structures. Therefore it has the advantage to treat also surface systems, which can not be treated within XCrysDen [50].

The constraint for the Θ-angle of the magnetic moments implemented in the JuTiBi code can be also improved. Beside implementing a constraint for the ϕ-angle, which is missing in the code, I would think about changing the type of constraint. The recent method is not able to properly fix the magnetic moment for all systems. In particular the magnetic moments of a very complex systems can be difficult to fix. I would propose to test a constraint, which induced magnetic field is not pointing perpendicular to the magnetic moment but rather along the direction. Another possibility could be to work in the local frame of the magnetic moment, so that they can not change their direction during self consistency.

The last improvement is a rather important one, but unfortunately also a very difficult task. It would be very nice to have universal and transferable parameter sets for each element, which allow a nice quantitative description of the systems, not depending on their structure and composition. I think this is almost impossible to achieve. Therefore it would be already nice to perform simple *ab-initio* calculations (as f.ex. LMTO) to obtain the specific parameters accurately describing the non-magnetic electronic structure of the system. With these parameters of this specific system magnetic calculations can be performed on top within the JuTiBi code to spare a lot of computational time and obtain very accurate results. However, there are disadvantages of this method compared to the recent application in the JuTiBi code. The method would become more complex, because an additional *ab-initio* calculation has to be performed for each new structure before using the JuTiBi code. Furthermore an increased complexity of the method leads to a more difficult interpretation of results. Nevertheless to obtain quantitatively reliable

values within the tight-binding scheme I do not see any alternative.

A Appendix

A.1 Angular moment operator in atomic orbital representation

Calculating the angular moment operator $\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$ in representation of the atomic orbitals $|\mu\rangle$ is done straight forward by using the following equations for L_z and the ladder operators $L_\pm$:

$$L_z \cdot |l, m\rangle = m \cdot |l, m\rangle \tag{A.1}$$

$$L_\pm \cdot |l, m\rangle = \sqrt{l \cdot (l+1) - m \cdot (m \pm 1)} \cdot |l, m \pm 1\rangle \ , \tag{A.2}$$

where $|l, m\rangle$ are the eigenfunctions of L^2 and L_z, which are the complex spherical harmonics in real space representation.

With $L_x = \frac{1}{2} \cdot (L_+ + L_-)$ and $\mathrm{i} \cdot L_y = \frac{1}{2} \cdot (L_+ - L_-)$ we derive the following expressions for the angular moment in atomic orbital representation $(s, p_x, p_y, p_z, d_{xy}, d_{xz}, d_{yz}, d_{x^2-y^2}, d_{z^2})$:

$$[\underline{\boldsymbol{L_x}}]^\nu_\mu = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\mathrm{i} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{i} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathrm{i} & -\mathrm{i}\sqrt{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{i} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{i}\sqrt{3} & 0 & 0 \end{pmatrix} \tag{A.3}$$

$$[\underline{\boldsymbol{L_y}}]^\nu_\mu = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{i} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\mathrm{i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{i} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathrm{i} & \mathrm{i}\sqrt{3} \\ 0 & 0 & 0 & 0 & -\mathrm{i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\mathrm{i}\sqrt{3} & 0 & 0 & 0 \end{pmatrix} \tag{A.4}$$

$$[\underline{\boldsymbol{L_z}}]^{\nu}_{\mu} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\mathrm{i} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\mathrm{i} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\mathrm{i} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2\mathrm{i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.5}$$

To obtain the SOC-matrix in atomic orbital representation, one has to use the equation

$$\mathcal{H}_{\mathrm{SOC}} \propto \boldsymbol{L} \cdot \boldsymbol{S} \tag{A.6}$$

with $\boldsymbol{S} = \frac{\hbar}{2} \cdot \boldsymbol{\sigma}$.

A.2 Derivation of the phase factors

The Bloch-waves

$$|\Phi^{\uparrow}_{i\mu}(\boldsymbol{k})\rangle = \frac{1}{\sqrt{N}} \cdot \sum_{n} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \cdot |\boldsymbol{n}, i, \mu\rangle \cdot \begin{pmatrix} \mathrm{e}^{-\frac{\mathrm{i}}{2}\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \\ 0 \end{pmatrix} \tag{A.7}$$

$$|\Phi^{\downarrow}_{i\mu}(\boldsymbol{k})\rangle = \frac{1}{\sqrt{N}} \cdot \sum_{n} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \cdot |\boldsymbol{n}, i, \mu\rangle \cdot \begin{pmatrix} 0 \\ \mathrm{e}^{\frac{\mathrm{i}}{2}\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \end{pmatrix} . \tag{A.8}$$

satisfy the generalized Bloch theorem [9, 10, 11] and they form an orthonormalized set of basis functions. The Hamiltonian displays as follows in representation of these Bloch waves:

$$\langle \boldsymbol{\Phi}^{\sigma}_{i,\mu} | \, \mathcal{H} \, | \boldsymbol{\Phi}^{\sigma'}_{j,\nu} \rangle = \frac{1}{N} \sum_{n,n'} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{R}_{n'}-\boldsymbol{\tau}_i)} \cdot (\tilde{\boldsymbol{\chi}}^{\sigma}_{n',i,\mu})^{\dagger} \cdot (H^{j,\nu}_{i,\mu})^{\sigma\sigma'}(\boldsymbol{R}_{n'} - \boldsymbol{R}_n) \cdot \tilde{\boldsymbol{\chi}}^{\sigma'}_{n,j,\nu}$$

with

$$\begin{aligned} \tilde{\boldsymbol{\chi}}^{\sigma}_{n,i,\mu} &= \mathcal{U} \cdot \boldsymbol{\chi}^{\sigma}_{n,i,\mu} \\ &= \begin{cases} \begin{pmatrix} \mathrm{e}^{(-\mathrm{i}/2\cdot\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i)} \\ 0 \end{pmatrix} & \text{if } \sigma = \uparrow \\ \begin{pmatrix} 0 \\ \mathrm{e}^{(\mathrm{i}/2\cdot\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_i))} \end{pmatrix} & \text{if } \sigma = \downarrow \end{cases} , \end{aligned} \tag{A.9}$$

where $\mathcal{U}$ is the spin transformation matrix from eq. 2.65 with $\phi = \boldsymbol{q} \cdot (\boldsymbol{R}_n + \boldsymbol{\tau}_i)$ and $\Theta = 0$. The χ^{σ} are the $\uparrow$- and $\downarrow$-spins in the global frame, therefore $(1,0)^{\mathrm{T}}$ and $(0,1)^{\mathrm{T}}$. Therefore it holds:

$$
\begin{aligned}
(H_{i,\mu}^{j,\nu})^{\sigma\sigma'}(\boldsymbol{k},\boldsymbol{q}) &= \frac{1}{N}\cdot\sum_{n,n'} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{R}_{n'}-\boldsymbol{\tau}_i)}\cdot(\boldsymbol{\chi}_{n',i,\mu}^{\sigma})^{\dagger}\cdot[\mathcal{U}^{\dagger}(\boldsymbol{q}\cdot(\boldsymbol{R}_{n'}+\boldsymbol{\tau}_i))\cdot \\
&\quad (H_{i,\mu}^{j,\nu})^{\sigma\sigma'}(\boldsymbol{R}_{n'}-\boldsymbol{R}_n)\cdot\mathcal{U}(\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j))]\cdot(\boldsymbol{\chi}_{n,j,\nu}^{\sigma'}) \,. \qquad \text{(A.10)}
\end{aligned}
$$

$\mathcal{H}_0$ and $\mathcal{H}_{\mathrm{LCN}}$ is proportional to the unit matrix in spin-space. With $t_{i,\mu\to j,\nu}(\boldsymbol{R}_{n'}-\boldsymbol{R}_n)$ the hopping element of the Hamiltonian from the state i,μ to the state j,ν, it holds due to the spin-independence of the hopping elements:

$$
[H_0]_{i\mu}^{j\nu} = \begin{pmatrix} t_{i,\mu\to j,\nu} & 0 \\ 0 & t_{i,\mu\to j,\nu} \end{pmatrix} .
$$

The magnetic part $\mathcal{H}_{\mathrm{mag}}$ is invariant under this transformation, because it consists only of (spin-dependent) onsite-elements.

Therefore the expression $\mathcal{U}^{\dagger}(\boldsymbol{q}\cdot(\boldsymbol{R}_{n'}+\boldsymbol{\tau}_i))\cdot\mathcal{U}(\boldsymbol{q}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j))$ yields the specific phase-factors $(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\sigma\sigma'}$:

$$
\begin{aligned}
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\uparrow} &= \mathrm{e}^{-\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \\
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\downarrow\downarrow} &= \mathrm{e}^{\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)} \\
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\downarrow} &= 0 \qquad \text{(A.11)}
\end{aligned}
$$

The dependence on the cone angle Θ is not contained in the phase-factors. By treating the system in the global frame they are contained in the description of $\mathcal{H}_{\mathrm{mag}}$. By using instead a local frame description as in [16] the phase-factors are more complicated due to their Θ-dependence, but the magnetic part is very simple (it is diagonal in spin).

For a spin-spiral of the type (c) in fig. 2.15 the phase-factors are a little bit different. In an analogous way one can derive the phase factors for these "flat-spirals" as follows:

$$
\begin{aligned}
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\uparrow} &= \cos\left(\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)\right) \\
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\downarrow\downarrow} &= \cos\left(\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)\right) \\
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\downarrow} &= -\sin\left(\frac{\boldsymbol{q}}{2}\cdot(\boldsymbol{R}_n+\boldsymbol{\tau}_j-\boldsymbol{\tau}_i)\right) \\
(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\downarrow\uparrow} &= [(s_i^j(\boldsymbol{q}\cdot\boldsymbol{R}_n))^{\uparrow\downarrow}]^* \qquad \text{(A.12)}
\end{aligned}
$$

Another important remark: Notice that the eigenvectors after the diagonalization $\mathcal{H}$ do not yet have the correct structure to yield the correct directions of the magnetic moments, if the system contains of basis atoms with $\boldsymbol{\tau}_i\cdot\boldsymbol{q}\neq 0$. To obtain the correct eigenvectors, the corresponding elements have to be rotated with the q-dependent spin-rotation matrices

$$
\underline{\boldsymbol{U}}_i = \begin{pmatrix} \exp^{-\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot\boldsymbol{\tau}_i} & 0 \\ 0 & \exp^{\mathrm{i}\frac{\boldsymbol{q}}{2}\cdot\boldsymbol{\tau}_i} \end{pmatrix} .
$$

To demonstrate this necessity, we should take a look to the following system:

Imagine we have a unit cell of a 1-dim. system consisting of 4 basis atoms along the direction of a spin-spiral vector $\boldsymbol{q} = 0.25\,\boldsymbol{e}_x$ and a cone angle of Θ, as displayed in figure 2.13a. We can treat the system within the usual scheme of non-collinear magnetism indicated in eq. 2.68, which leads to following structure of the Hamiltonian:

$$\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_{\text{mag}}(\Theta, \phi_1, ..., \phi_4)\,.$$

with ϕ_i the azimuthal angle of the magnetic moment of basis atom i. But of course we can also treat the system within the gen. Bloch theorem, without exploiting the advantage of using the smaller unit cell (see fig. 2.13b). The ϕ-angles can be written as $\boldsymbol{q} \cdot \boldsymbol{\tau}_i$ and the structure of the Hamiltonian by considering the appearing phase-factors would be as follows:

$$\mathcal{H}^{\text{gen. BT}} = \mathcal{H}_0^{\text{gen. BT}}(\boldsymbol{q}) + \mathcal{H}_{\text{mag}}^{\text{gen. BT}}(\Theta)\,.$$

The transformations $\underline{\boldsymbol{U}}_i$ connect the two cases by the formalized equation:

$$\mathcal{H}^{\text{gen. BT}} = \mathcal{U} \cdot \mathcal{H} \cdot \mathcal{U}^\dagger$$

Therefore to obtain the same structure for the eigenvectors as in the usual scheme within eq. 2.68, the eigenvectors have to be rotated with the transformation $\mathcal{U}$ after the diagonalization. Note that the energies will not change due to the unitary structure of the transformation $\mathcal{U}$, but only the structure of the eigenvectors.

H_s^s	$= V_{ss\sigma}$
H_s^x	$= lV_{sp\sigma}$
H_x^x	$= l^2V_{pp\sigma} + (1-l^2)V_{pp\pi}$
H_x^y	$= lmV_{pp\sigma} - lmV_{pp\pi}$
H_x^z	$= lnV_{pp\sigma} - lnV_{pp\pi}$
H_s^{xy}	$= \sqrt{3}lmV_{sd\sigma}$
$H_s^{x^2-y^2}$	$= \frac{1}{2}\sqrt{3}(l^2-m^2)$
$H_s^{z^2}$	$= \left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{sd\sigma}$
H_x^{xy}	$= \sqrt{3}l^2mV_{pd\sigma} + m(1-2l^2)V_{pd\pi}$
H_x^{yz}	$= \sqrt{3}lmnV_{pd\sigma} - 2lmnV_{pd\pi}$
H_x^{xz}	$= \sqrt{3}l^2nV_{pd\sigma} + n(1-2l^2)V_{pd\pi}$
$H_x^{x^2-y^2}$	$= \frac{1}{2}\sqrt{3}l(l^2-m^2)V_{pd\sigma} + l(1-l^2+m^2)V_{pd\pi}$
$H_y^{x^2-y^2}$	$= \frac{1}{2}\sqrt{3}m(l^2-m^2)V_{pd\sigma} - m(1+l^2-m^2)V_{pd\pi}$
$H_z^{x^2-y^2}$	$= \frac{1}{2}\sqrt{3}n(l^2-m^2)V_{pd\sigma} - n(l^2-m^2)V_{pd\pi}$
$H_x^{z^2}$	$= l\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{pd\sigma} - \sqrt{3}ln^2V_{pd\pi}$
$H_y^{z^2}$	$= m\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{pd\sigma} - \sqrt{3}mn^2V_{pd\pi}$
$H_z^{z^2}$	$= n\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{pd\sigma} + \sqrt{3}n(l^2+m^2)V_{pd\pi}$
H_{xy}^{xy}	$= 3l^2m^2V_{dd\sigma} + (l^2+m^2-4l^2m^2)V_{dd\pi} + (n^2+l^2m^2)V_{dd\delta}$
H_{xy}^{yz}	$= 3lm^2nV_{dd\sigma} + ln(1-4m^2)V_{dd\pi} + ln(m^2-1)V_{dd\delta}$
H_{xy}^{xz}	$= 3l^2mnV_{dd\sigma} + mn(1-4l^2)V_{dd\pi} + mn(l^2-1)V_{dd\delta}$
$H_{xy}^{x^2-y^2}$	$= \frac{3}{2}lm(l^2-m^2)V_{dd\sigma} + 2lm(m^2-l^2)V_{dd\pi} + \frac{1}{2}lm(l^2-m^2)V_{dd\delta}$
$H_{yz}^{x^2-y^2}$	$= \frac{3}{2}mn(l^2-m^2)V_{dd\sigma} - mn\left[1+2(l^2-m^2)\right] V_{dd\pi} + mn\left[1+\frac{1}{2}(l^2-m^2)\right] V_{dd\delta}$
$H_{xz}^{x^2-y^2}$	$= \frac{3}{2}nl(l^2-m^2)V_{dd\sigma} + nl\left[1-2(l^2-m^2)\right] V_{dd\pi} - nl\left[1-\frac{1}{2}(l^2-m^2)\right] V_{dd\delta}$
$H_{xy}^{z^2}$	$= \sqrt{3}lm\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{dd\sigma} - 2\sqrt{3}lmn^2V_{dd\pi} + \frac{1}{2}\sqrt{3}lm(1+n^2)V_{dd\delta}$
$H_{yz}^{z^2}$	$= \sqrt{3}mn\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{dd\sigma} + \sqrt{3}mn(l^2+m^2-n^2)V_{dd\pi} - \frac{1}{2}\sqrt{3}mn(l^2+m^2)V_{dd\delta}$
$H_{xz}^{z^2}$	$= \sqrt{3}ln\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{dd\sigma} + \sqrt{3}ln(l^2+m^2-n^2)V_{dd\pi} - \frac{1}{2}\sqrt{3}ln(l^2+m^2)V_{dd\delta}$
$H_{x^2-y^2}^{x^2-y^2}$	$= \frac{3}{4}(l^2-m^2)^2V_{dd\sigma} + \left[l^2+m^2-(l^2-m^2)^2\right] V_{dd\pi} + \left[n^2+\frac{1}{4}(l^2-m^2)^2\right] V_{dd\delta}$
$H_{x^2-y^2}^{z^2}$	$= \frac{1}{2}\sqrt{3}(l^2-m^2)\left[n^2 - \frac{1}{2}(l^2+m^2)\right] V_{dd\sigma} + \sqrt{3}n^2(m^2-l^2)V_{dd\pi}$ $+\frac{1}{4}\sqrt{3}(1+n^2)(l^2-m^2)V_{dd\delta}$
$H_{z^2}^{z^2}$	$= \left[n^2 - \frac{1}{2}(l^2+m^2)\right]^2 V_{dd\sigma} + 3n^2(l^2+m^2)V_{dd\pi} + \frac{3}{4}(l^2+m^2)^2V_{dd\delta}$

Table A.1: Slater-Koster transformations for s, p and d-orbitals. The matrix elements $H_\mu^\nu(\boldsymbol{R}_n)$ of the real-space Hamiltonian depend on the direction cosines $l = \frac{(\boldsymbol{R}_n)_x}{|\boldsymbol{R}_n|}$, $m = \frac{(\boldsymbol{R}_n)_y}{|\boldsymbol{R}_n|}$ and $n = \frac{(\boldsymbol{R}_n)_z}{|\boldsymbol{R}_n|}$ of the bonding vector $\boldsymbol{R}_n$. The table is separated in s-s, s-p, p-p, s-d, p-d and d-d matrix elements by horizontal lines.

B Appendix

B.1 Parameter sets for Fe and Pt

The following parameter sets in the tables B.1, B.2 and B.3 are obtained by Mehl *et al.* by fitting the TB-bands to LDA/GGA-*ab-initio* band structures. The results are published in [3, 4, 57]. The parameters are necessary for the description of the Hamiltonian and the overlap matrix via eq. 2.21 and eq. 2.23.
Some informations about the parameters:

- The distance R is in atomic units and the Slater-Koster parameters are in Rydberg in the Mehl *et al.* parametrization. Therefore the parameters are in following units:

$$[a] = \text{Ry}, \quad [b] = \text{Ry} \cdot (\text{a.u.})^{-1}, \quad [c] = \text{Ry} \cdot (\text{a.u.})^{-2}, \quad [d] = (\text{a.u.})^{-0.5}$$

$$[\alpha] = \text{Ry}, \quad [\beta] = \text{Ry}, \quad [\gamma] = \text{Ry}, \quad [\chi] = \text{Ry}, \quad [\lambda] = (\text{a.u.})^{-0.5}\,.$$

- The on-site contributions for the overlap matrix are 1.0.
- The parameters c and χ are zero in some of the presented parameter sets, but in general they are not.
- No difference between the t_{2g}- and e_g-states is introduced in the parameter sets.
- The parametrization of the overlap matrix is via the old parametrization scheme of Mehl *et al.* [5].

The Stoner parameter in the d-orbitals of Fe is chosen to 0.96 eV and for Pt to 0.58 eV. The Stoner parameter in the s- and p-orbitals is 10-times smaller. The SOC-parameters of Fe are 0.06 eV for the d-orbitals and 0.18 eV for the p-orbitals. For Pt the SOC-parameters are much larger with values of 0.53 eV for the d-orbitals and 2.5 eV for the p-orbitals.

$R_c = 16.5\,\text{a.u.} \quad L_c = 0.5\,\text{a.u.} \quad \lambda = 1.61591889336\text{E+}00$

orbital	α	β	γ	χ
s	3.13939258386E−01	-1.50340969449E+02	5.60131523543E+04	0.0
p	7.50228117251E−01	1.32173351946E+02	-4.83053227618E+03	0.0
d	6.79135670479E−02	1.80487939584E+01	-1.71453585941E+03	0.0

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	4.55043825222E−01	-1.21734123013E+00	0.0	9.14314863459E−01
$V_{sp\sigma}$	8.72706208251E−01	-2.65085983323E−03	0.0	6.78102167954E−01
$V_{pp\sigma}$	1.02661455604E+00	4.62878498635E−02	0.0	6.49770224489E−01
$V_{pp\pi}$	-3.61138626592E+01	8.18233539363E+00	0.0	1.17142147452E+00
$V_{sd\sigma}$	5.06145301841E−01	-3.03407375680E−01	0.0	8.23582443933E−01
$V_{pd\sigma}$	3.62443449178E+00	-1.20097178354E+00	0.0	8.70891888071E−01
$V_{pd\pi}$	-1.23170095125E+00	8.46861596670E−01	0.0	9.92415550868E−01
$V_{dd\sigma}$	-1.30200356145E+00	9.13566450579E−02	0.0	7.96761216755E−01
$V_{dd\pi}$	3.03158415211E+00	-2.29519971304E−01	0.0	9.29910152711E−01
$V_{dd\delta}$	-2.42866093686E+00	3.43810222548E−01	0.0	1.01267224138E+00

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	2.08691737655E+00	1.50951711074E+00	0.0	9.38172864034E−01
$V_{sp\sigma}$	6.61324794182E+00	-2.39708289279E+00	0.0	8.42816286968E−01
$V_{pp\sigma}$	7.79047201212E+00	-2.24565537971E+00	0.0	7.63650075929E−01
$V_{pp\pi}$	-5.28178851021E+00	1.65816689927E+00	0.0	8.02822184386E−01
$V_{sd\sigma}$	-6.29773771311E+02	1.71528662797E+02	0.0	1.30437406994E+00
$V_{pd\sigma}$	-3.79544838869E+00	7.64597367732E−01	0.0	8.45424416578E−01
$V_{pd\pi}$	1.21878943829E+02	-4.37849569870E+01	0.0	1.33895185420E+00
$V_{dd\sigma}$	-4.33296538382E+00	4.93979871821E+00	0.0	1.19874436110E+00
$V_{dd\pi}$	-1.03641263737E+00	1.93977003741E−01	0.0	8.98115322198E−01
$V_{dd\delta}$	4.62136029975E+00	-1.26354179980E+00	0.0	1.05391660677E+00

Table B.1: Fe LDA-parameter set: The first table contains the necessary parameters for the on-site energies, the second table contains the parameters for the Hamiltonian matrix elements (hopping elements), whereas the third table contains the parameters for the overlap matrix.

$R_c = 16.5\,\text{a.u.} \quad L_c = 0.5\,\text{a.u.} \quad \lambda = 0.167876018007\text{E+}01$

orbital	α	β	γ	χ
s	.111237776825E+00	.157472037798E+03	.279493598875E+06	-.886322891071E+08
p	.512530808853E+00	.305694624645E+03	-.627918805139E+05	-.526504629561E+07
d	.956141408199E−01	-.229757020621E+02	.131512865599E+05	-.854972223057E+06

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	.428297660344E+00	-.736375016998E+00	.242575588592E−02	.858600612943E+00
$V_{sp\sigma}$	.817564285855E+00	-.114712495851E−02	.571303916419E−02	.713729903579E+00
$V_{pp\sigma}$	-.624426913703E−01	.169380062153E+00	-.133925020299E−01	.531763011628E+00
$V_{pp\pi}$	.457554537257E+03	-.268469067026E+02	-.213983975589E+02	.146261128932E+01
$V_{sd\sigma}$	.490269004091E+00	-.381292939420E+00	-.236890354241E−02	.816265910951E+00
$V_{pd\sigma}$	.380280165519E+01	-.116865592373E+01	.245781494444E−02	.853580010894E+00
$V_{pd\pi}$	-.170292010391E+01	.789563902306E+00	-.347134872764E−01	.955500742639E+00
$V_{dd\sigma}$	-.112692175412E+01	.906852484661E−01	-.110402112411E−01	.811112962492E+00
$V_{dd\pi}$	.628847411766E+01	-.136259655000E+01	.979967718827E−01	.931326036079E+00
$V_{dd\delta}$	-.643123923582E+03	.359148848319E+03	-.546524202557E+02	.147480783649E+01

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	-.130030306848E+02	.373133347375E+01	.641105326253E+00	.102602926730E+01
$V_{sp\sigma}$	.105130415274E+02	-.357406235042E+01	-.229978575757E+00	.966897827380E+00
$V_{pp\sigma}$	.151286790032E+02	-.335893499630E+01	-.246789283219E+00	.849462577539E+00
$V_{pp\pi}$	-.255301615149E+01	.154908422778E+01	-.108991538227E+00	.758944165379E+00
$V_{sd\sigma}$	-.596211016918E+03	.158217370850E+03	-.132981350821E+01	.128158963555E+01
$V_{pd\sigma}$	-.520996890837E+01	.163409810874E+01	-.490336116995E−01	.762616604972E+00
$V_{pd\pi}$	-.449930772185E+02	-.114818134979E+02	-.313846669042E+01	.136886120058E+01
$V_{dd\sigma}$	.267625592154E+02	-.123743969604E+02	.154936323562E+01	.974949432957E+00
$V_{dd\pi}$	-.137179418998E+01	.163248717151E+00	.281060705718E−02	.717618717462E+00
$V_{dd\delta}$	.105113669588E+02	-.450600002662E+00	-.395000205694E+00	.101542781750E+01

Table B.2: Fe GGA-parameter set: The first table contains the necessary parameters for the on-site energies, the second table contains the parameters for the Hamiltonian matrix elements (hopping elements), whereas the third table contains the parameters for the overlap matrix.

$R_c = 16.5\,\text{a.u.} \quad L_c = 0.5\,\text{a.u.} \quad \lambda = 1.48637407133\text{E+}00$

orbital	α	β	γ	χ
s	5.83155590317E−03	2.05115875626E+02	-2.04104996463E+04	0.0
p	8.17822535174E−01	5.89749525600E+01	1.12389938450E+04	0.0
d	4.98287869238E−02	-4.47348238061E+00	6.15745074751E+02	0.0

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	-1.38284631122E+00	-1.20674487008E−01	0.0	8.13944927710E−01
$V_{sp\sigma}$	1.83353981972E+00	5.86126500009E−01	0.0	8.45134159397E−01
$V_{pp\sigma}$	1.26085142803E+00	1.03890170253E+00	0.0	8.31224712555E−01
$V_{pp\pi}$	2.05175594909E+02	-3.96379653812E+01	0.0	1.13164516026E+00
$V_{sd\sigma}$	-2.89294520731E+00	2.44945737996E−01	0.0	7.98201020712E−01
$V_{pd\sigma}$	1.58551087829E+00	-6.94618881904E−01	0.0	8.07843642691E−01
$V_{pd\pi}$	8.60594193094E−01	-3.28209927985E−02	0.0	8.59013576224E−01
$V_{dd\sigma}$	-1.75160317524E+00	-3.17276845523E−01	0.0	8.74768883496E−01
$V_{dd\pi}$	7.04207263761E+00	-3.61062702293E−01	0.0	9.55737743700E−01
$V_{dd\delta}$	-1.19554932632E+00	1.68459650181E−01	0.0	8.73327306708E−01

$V_{ll'm}$	a	b	c	d
$V_{ss\sigma}$	8.48032304726E+00	-1.21344187855E+00	0.0	8.73588630500E−01
$V_{sp\sigma}$	2.07665691775E+03	-4.87117198609E+02	0.0	1.42827779253E+00
$V_{pp\sigma}$	-1.59104408799E+04	3.39006655850E+03	0.0	1.46844841218E+00
$V_{pp\pi}$	-4.68565688171E+02	1.29049135927E+02	0.0	1.24800847008E+00
$V_{sd\sigma}$	-1.88027778797E+00	4.33067941947E−01	0.0	7.48675014457E−01
$V_{pd\sigma}$	1.40632324463E−01	-1.08623564920E−02	0.0	4.30679519174E−01
$V_{pd\pi}$	-2.32135126063E+00	3.29353331687E−01	0.0	7.81550208218E−01
$V_{dd\sigma}$	7.85664648549E−01	-4.01900709200E−02	0.0	7.02103011484E−01
$V_{dd\pi}$	-3.26443115027E+00	5.45024650243E−02	0.0	9.05925825725E−01
$V_{dd\delta}$	1.65626596755E+03	-2.95917301165E+02	0.0	1.32876007979E+00

Table B.3: Pt LDA-parameter set: The first table contains the necessary parameters for the on-site energies, the second table contains the parameters for the Hamiltonian matrix elements (hopping elements), whereas the third table contains the parameters for the overlap matrix.

C Appendix - Example of the inputcard

```
***** Input file for JuTiBi code *****
-----------------------------------------------------------------------
*** Structure Properties - Lattice ***

ALATBASIS=  2.87d0 1.0d0 1.0d0   lattice constants a, b, c: first lattice constant a : scaling factor | other values are b/a and c/a
                                 the scaling factor a acts on the x-components, whereas b and c act on the y and z-components of the
                                 Bravais vectors (!for 1- and 2-dim structurs don't set b/a and c/a equal 0, set the bravais vectors 0)

Read in rows and in units of a, b and c! For a (1- or) 2-dim. geometry set (second and) third bravais-vectors to zero!
Note: 1-dim. structures have to be in the x-direction, therefore do not use (0 0 1) as first Bravais vector in this case!
BRAVAIS
-0.5d0 0.5d0 0.5d0
0.5d0 -0.5d0 0.5d0
0.5d0 0.5d0 -0.5d0

-----------------------------------------------------------------------
*** Basis atoms in unit cell ***

CARTESIAN= T      Basis vectors in cartesian (true) or Bravais (false) representation

BASATOMS          read in rows and in units of a, b, c
  0.d0  0.d0      0.d0

NUMBASIS=1                          number of basis atoms

Information: If one wishes a slab - geometry the z-component of the basisvectors is always in cartesian coordinates,
but the first two components are in cartesian or bravais representation dependent on what is switched on.
-----------------------------------------------------------------------
*** k-Mesh Properties ***

BZDIVIDE= 20 20 20          points of k mesh in direction of the reciprocal Bravais vectors
                            ATTENTION: points of k mesh should fit to relations of a,b,c | it would be also good to choose the values even
KPOIBZ= 8000                Number of points in full BZ
IrrBZ= T                    Use of irreducible part of BZ (true) or not (false)

HARD_DISK=F                 if true all eigenvectors are saved on the hard disk and not into the RAM (do not use it for too large systems!)

-----------------------------------------------------------------------
*** Properties for DOS calculation ***

NUMENDOS=1000               number of energy-values for energy-mesh in DOS-calculation
LORWIDTH=0.1d0              width of the Lorentzian curves in calculation of DOS (in eV)

FACLOR= 30                  factor of widths for calculation of DOS; it sets area in which DOS-calculation will be considered

SMEAR_MAG=T                 if true a fermi dirac function will be used for the evaluation of the magnetic moments (all charges are calculated
                            with the smearing, also this logical is set to false) | I recommend to switch it to true
FERM_BROD=0.001d0           broadening of the fermi-function in eV

DOSORBAT=T                  if true DOS per atom and per orbital will be calculated

LOGSPEC=F                   if true only one specific DOS/atom/orbital will be calculated
SPECORBAT                   which specific DOS shall be calculated (for example 1 5 for 1st atom and orbital d_xy)
 1 1

DOS_LOCAL=F                 switch to true if you want to have the DOS in local quant. axis
```

```
LOG_F_SUR=F                 if true a file for XCrysDen is created to calculate the Fermi surface

-------------------------------------------------------------------------
*** Properties of the Cluster Generation ***

MAXCLUST= 10000             maximum Bravais Vectors in Cluster
RMAXIMAL= 4.5d0             minimum radius for clustergeneration in units of a (for rrgen)
RCUTOFF=  3.5d0             cutoff radius for cluster creation (in units of a)
MAXBASIS= 50                maximum number of basis atoms

-------------------------------------------------------------------------
*** Atominfo ***

Settings: ( atomic number, number of used orbitals, specification which orbitals are used (1: used, 0: not used),
            SOC parameters under corresponding set of orbitals)
!!!LEAVE ONE LINE FREE BETWEEN DIFFERENT BASISATOMS!!! DO NOT CHANGE THIS FORM!

LMAXIMAL=9               total sum over all angular moment numbers l, therefore for all s, p and d orbitals l_max=9
NUMELUC=8                number of electrons per unit cell

-----------------|--------------------------------------
ZATOM       #ORB |  s_Orb |  px_Orb    py_Orb    pz_Orb |
45.0d0       9   |    1   |    1          1         1    |
-----------------|--------------------------------------|
SOC-Parameter:   |   ---  |           0.0d0              |
-----------------------------------------------------------------
dxy_Orb    dxz_Orb    d_yz_Orb    d(x2-y2)_Orb    d(z2-r2)_Orb |
   1          1          1            1               1        |
-----------------------------------------------------------------|
                                0.d0                             |
-----------------------------------------------------------------

-------------------------------------------------------------------------------
*** Input of the Slater-Koster parameters ***


IDENTATOM=T                 if true then all basisatoms are identical, for false they are not (-> new inputfile SKPinput)
                            if the NRL-TB scheme is used, this logical is not relevant, therefore switch it to true or false

ONLYNEXTN=F                 if true only next neighbors will be considerd, if false lower value is important

RNEIGH=3.5d0                distance in which neighbors are considered in TB-model (>= next N. distance, in units of a)

SKPVARY=T                   if true the Slater Koster parameters will be varied with the distance, the equation for
                            this dependence is displayed in PRB 54 4519 -> use inputfile SKP_input_papakonst
                            in the scheme of NRL-TB parameterization

OVLAP=T                     if true we consider an overlap matrix which is not the unity matrix
                            The overlap matrix needs as many parameters as the Slater Koster parameters!

PAPA_BINA=F                 if true binary compounds will be treated (more than binary are not possible (yet))
TYP_BASAT                   atomtype of the basis atoms (fit to order in SKP_input_papakonst)
 1   1

!!These values will be read in only if "identatom" is true and "skpvary" is false !!
!The second values are only important if "spinlog" and "skpspin" is true,
then the first value stands for the spin majority and the second for the minority!

ONSITES=0.0d0     0.0d0
ONSITEP=0.0d0     0.0d0
ONSITED=0.0d0     0.0d0

s-couplings:
VSS_SIGMA=0.0d0   0.0d0       OSS_SIGMA=0.0d0   0.0d0
VSP_SIGMA=0.0d0   0.0d0       OSP_SIGMA=0.0d0   0.0d0
VSD_SIGMA=0.0d0   0.0d0       OSD_SIGMA=0.0d0   0.0d0

p-couplings:
VPP_SIGMA=0.0d0   0.0d0       OPP_SIGMA= 0.0d0  0.0d0
```

```
VPP_PI=   0.0d0   0.0d0       OPP_PI=    0.0d0  0.0d0
VPD_SIGMA=0.0d0   0.0d0       OPD_SIGMA= 0.0d0  0.0d0
VPD_PI=   0.0d0   0.0d0       OPD_PI=    0.0d0  0.0d0

d-couplings:
VDD_SIGMA=0.0d0   0.0d0       ODD_SIGMA=0.0d0  0.0d0
VDD_PI=   0.0d0   0.0d0       ODD_PI=   0.0d0  0.0d0
VDD_DELTA=0.0d0   0.0d0       ODD_DELTA=0.0d0  0.0d0

---------------------------------------------------------------------
*** Properties for the band structure ***

Way of k-vectors:
Define the way in k space by giving the start and end points, which should be connected by a linear curve.
Put the number of points which should be used for the single way-parts before start- and end-point.
(for example: 100  0. 0. 0.   1. 0. 0.
      ‘       100  1. 0. 0.   1. 1. 0.
              100  1. 1. 0.   0. 0. 0.  )


For a 1-dim. system the way will be determined only by the x-components in the first line.
Therefore in this case only one way-part can be chosen

NUMWAYS=4               number of ways (<=10, otherwise code will crash)

NUMPTSWAY         CREATEWAY                 define way by start- and endpoints (cartesian and in units of 2*pi*(1/a,1/b,1/c))
 100                 0.5 0. 0.           0.5 0.5 0.
 100                 0.5 0.5 0.          0.5  0.5  0.5
 100                 0.5  0.5  0.5       0.0 0.0 0.0
 100                 0. 0. 0.            0.5 0.0 0.0

FAT_BANDS=T             switch to true if the weights of the eigenvectors should be also saved

---------------------------------------------------------------------
*** Properties for the self-consistent calculations (Stoner model and local charge neutrality) ***

TB_SELFC=T                    if true self consistent calculation will be executed
                              (Attention: if true, lower exchange-energies are start-values for the calculation)

SELF_U                        U_LCN for the local charge neutrality in eV (5 eV is a reasonable value)
1 5.0d0
2 5.0d0


CHARGE_0                      start-values for the mulliken-charges and the desired charge for local charge neutrality calculation
    1   8.0d0   8.0d0

Method of Mixing:

first method: (slow, but stable)
MIX_LIN=F                     linear mixing with:  alpha*u_new+(1-alpha)*u_old

MIX_ALPHA=0.1d0               should be between 0 and 1


second method: (fast, but could be unstable)
MIX_BROY=T                  Broyden-mixing
N_IN_LIN=3                  number of linear mixing steps before Broyden mixing starts

MAX_ITER=400                maximum number of iterations
SELF_COND=0.00001d0         condition for ending the self-consistent calculation: abs(u_new-u_old)<epsilon

RESTA_SC=F                  if true starting charges are imported from TB_SC_values.dat, from a specific step:
RESTA_IT=0                  this determines the specific step

---------------------------------------------------------------------
*** Magnetic Properties and SOC ***

SPINLOG= T                    true: introduce spin up and spin down energybands (-> dim(H) is 2x larger)
SKP_SPIN=F                    if true the SKP are different for different spins (--> 2x #SK-parameter, use only SKPinput as file)
```

```
                              can not be used together with NRL-TB parameterization

exchange energies (in eV)  (scheme: atom-nr.  s-orb  p-orb  d-orb)
Use positive exchange energies! For antiferromagnetism use non-collinear magnetism.
For a reasonable approximation choose s- and p-splitting 10 times smaller than d-splitting.
XC_ENERGY
    1   0.12   0.12   1.2

STON_PARA                     Stoner parameter of s-, p- and d-orbital in eV (Regard: exchange_energy=0.5*I*M)
1 0.096d0 0.096d0 0.96d0
1 0.058d0 0.058d0 0.58d0


SET_GAXIS=F                   if true a global spin-rotation is performed (switch on SOC!)


ROT_SPIN= 0.0d0 0.0d0 0.0d0     give the direction of the rotated spins
(important: Do not mix this global rotation with the individual rotation in the non-collinear case!)

SOCLOG= F                     with (true) or without (false) Spin-Orbit-Coupling

-------------------------------------------------------------------------
*** Non-collinear-Magnetism (including anti-ferromagnetism) ***
IMPORTANT: Switch the SOC-logical to true, and use SOC parameters of 0, if no SOC case is wished

NCMAG=F                       switch to true if non collinear magnetism should be considered

-------non-collinearity by hand:
NC_IN_UC=F                    switch to true if the lower angles for the mag. moments should be set by hand

If true then fill up following tabel:   (atomnr. in unit cell |  angle theta  | angle phi )
(angles in degree; rot-axis is y for theta and z for phi)

NC_ANGLES
1  0.0d0 0.0d0
2  0.0d0 0.0d0

-------Constraint for magnetic moments:
CONST_ANG=F                    switch to true if the directions of the magnetic mopmetns should be fixed
U_CONST= 0.d0                  "Lagrange Parameter" for the constraint in eV (5 eV is reasonable)


-------Spin-spirals:
BLO_WAVE=F                     switch to true if a spin-spiral should be used

If true then specify spin-spiral :
BLO_THETA= 0.0d0          cone angle in degrees (set to 0 for flat-spiral!)
BLO_Q= 0.0d0 0.0d0 0.d0   q-vector of the spiral (in units of 2*pi*(1/a,1/b,1/c)!)

FLAT_SPIR=F               switch to true to use a flat spiral for surface systems (i.e.: m rotating in x-z-plane)
                          !! cone angle has to be 0 in this case !!

-------Treatment of SOC in spin-spirals:
SOC_PERTU=F               switch to true if a 1st order perturbation theoretical treatment of SOC should be
                          executed in the spin-spiral case
CHG_FE=F                  if true the Fermi energy is changed after adding the 1st order SOC contribution

-------q-mesh:
LOG_QMESH=F               switch to true if a q-mesh should be created (necessary for J_ij calculation)
BZQDIVIDE= 10 10 10       number of points of q mesh along reciprocal Bravais directions
                          ATTENTION: points of q mesh have to fit to relations of a,b,c -
                          it would be good to choose the values even

QPOIBZ= 1000              number of points in full BZ for q-mesh
IRRQBZ=T                  use irreducible part of BZ (true) or not (false) for the q-mesh

-------q-way:
LOG_Q_WAY=F               switch on if one wants to define a q-way for the spin-spiral case
NUMQWAYS=5                number of ways (<=10, otherwise code will crash)
```

```
NUM_QWAY   CREA_QWAY                  define way by start- and endpoints (cartesian and in units of 2*pi*(1/a,1/b,1/c) )
 10             0.0d0 0.d0 0.d0       0.5d0 0.5d0 0.d0
 10             0.5d0 0.5d0 0.d0      0.5d0 0.5d0 0.5d0
 10             0.5d0 0.5d0 0.5d0     0.d0 0.d0 0.d0
 10             0.d0 0.d0 0.d0        1.0d0 0.0d0 0.0d0
 10             1.0d0 0.0d0 0.0d0     0.5d0 0.5d0 0.0d0

-----------------------------------------------------------------------
 *** Cut Bondings *** ! ALPHA STATUS

If you want to cut bondings, set the logical to .true. and describe the bonding you want to cut with its bonding vector.
You can see the according vector in the file "shells_neighbours.dat"!

BONDCUT=F                             if true specified bonds will be cut out

To define the bondings, which should be cut out write for example: 1  0.75  0.  0. ,
if the bonding with the vector (0.75,0,0) should be cut out (in units of a) in the cluster of basisatom 1.
IMPORTANT: Take care to cut bondings in a symmetric way, otherwise the Hamiltonian could be non-hermitian.

NUMCUTBON=2                           number of bond-cuts

SPECBONDS                             specific bondings, which should be cut out (each bonding in one line)
 1 0. 0. 1.5
 2 0. 0. -1.5
```

D Appendix - Keywords of the `inputcard`

This appendix shows a list containing all appearing keywords in the `inputcard`. Additionally the corresponding variables of the JuTiBi code and a short description for each keyword is given below.

Keyword	**Variable name**	**Short description**
`#ORB`	temporary value	This keyword is used to read in the number of orbitals for each atom.
`ALATBASIS`	`alatc`	The lattice constants in the form a, $\frac{b}{a}$ and $\frac{c}{a}$.
`BASATOMS`	`basis`	The vectors for the basis atoms in the unit cell are saved columnwise in the array `basis`. Note that the vectors of the basis atoms are saved in cartesian coordinates into the array regardless of the logical `CARTESIAN` in the `inputcard`. The vectors are given in units of (a,b,c) depending on the cartesian component.
`BLO_Q`	`blo_q`	The spiral vector q of a spin-spiral should be enterd in cartesian representation and in units of $2\pi \cdot (\frac{1}{a}, \frac{1}{b}, \frac{1}{c})$.
`BLO_THETA`	`blo_theta`	Enter the cone angle Θ of the spin-spiral in degrees.
`BLO_WAVE`	`blo_wave`	If true a spin-spiral is considered within the gen. Bloch theorem.
`BONDCUT`	`bondcut`	If true the user can cut out some specified bondings of the crystal.
`BRAVAIS`	`abravais`	The Bravais vectors are stored in the columns of this array. Note that the Bravais vectors are in units of (a,b,c), depending on the cartesian component.
`BZDIVIDE`	`nkxyz`	Number of k-points along the reciprocal Bravais vector directions in the k-mesh division.

Continued...

Keyword	Variable name	Short description
BZQDIVIDE	nqxyz	Number of q-points along the reciprocal Bravais vector directions in the q-mesh division.
CARTESIAN	LCART[1]	If T the vectors of the basis atoms are imported in cartesian representation, whereas for F the Bravais representation is used.
CHARGE_0	total_mullikan_charge, mullikan_charge_start	The total mulliken charge for each basis atom and the desired reference charge within the local charge neutrality description for all basis atoms. See eq. 2.44 for more details.
CHG_FE	log_change_fermi_energy	If true the Fermi energy will be recalculated after adding the 1st order contribution from SOC in the spin-spiral calculation with soc_pertubation=true.
CONST_ANG	log_theta_constraint	If true a constraint is used to pin the Θ angle of the magnetic moments. It works for spin-spirals as also for the case nc_in_uc=true.
CREA_QWAY	startpts, endpts[1]	The start- and endpoints of the q-paths.
CREATEWAY	startpts, endpts[1]	The start- and endpoints of the k-paths.
DOS_LOCAL	dos_local	If true the DOSs are calculated in the local spin-frame of the atoms.
DOSORBAT	Dosorbat	If true the partial DOS is calculated (i.e. the atom and orbital resolved DOS).
FACLOR	faclor	In a region of 2*faclor*lor_width around an energy eigenstate the DOS-contribution is calculated, whereas outside this region the DOS-contribution is approximated to zero.
FAT_BANDS	fat_bands	If true the weights of the eigenvectors for the band structure calculation is also calculated and written out. These weights can be used for example in fat-band representations.
FERM_BROD	fermi_broadening	The broadening $k_B \cdot T$ in the Fermi-Dirac smearing function in eV.

Continued...

[1]Not in the list of the appendix E.

Keyword	Variable name	Short description
FLAT_SPIR	flat_spiral	If true a spin-spiral rotating in the $x-z$-plane is calculated. Note that blo_theta has to be zero in this case. Note that this form of spiral is necessary to investigate DMI in surface systems.
HARD_DISK	hard_disk	If true the eigenvectors and modified eigenvectors (see eq. 2.32) are saved onto the hard disk, instead in the memory. ATTENTION: This is not suited for very large systems.
IDENTATOM	identatoms[1]	If true the SKPs are created within the assumption that the basis atoms are identical. Note that in the case SKPVARY=T this keyword is insignificant!
IRRBZ	irr	If true the irreducible part of the k-mesh is calculated.
IRRQBZ	irr_q	If true the irreducible part of the q-mesh is calculated.
KPOIBZ	npoibz	Number of k-points in the full k-mesh
LMAXIMAL	lmax	$\sum_l(2 \cdot l + 1)$, l: angular moment quantum number ; for treating s, p and d-orbitals leave it to the value 9.
LOG_F_SUR	log_fermi_surface	If true a file for the external program XCrysDen is created to calculate the Fermi surface of a 3-dim. system.
LOG_Q_WAY	log_q_way	If true the magnon dispersion along a q-way in the Brillouin zone can be calculated.
LOG_QMESH	log_qmesh	If true the energies of spin-spirals with q-values from the full or irr. Brillouinzone are calculated. This is necessary to construct the file jenerg.dat, which can be used to calculate the Heisenberg exchange-coupling parameters.
LOGSPEC	log_spec_orbat	If true only one specific partial DOS for an atom and an orbital is calculated.
LORWIDTH	lor_width	The width in eV of the Lorentzians used in the DOS-calculation (see eq. 2.30 and subroutine calc_DOS).
MAXBASIS	numbasis_max[1]	The maximum number of basis atoms necessary for the allocation of arrays in the cluster generation.

Continued...

Keyword	Variable name	Short description
MAXCLUST	max_num_cluster	Maximum number of Bravais vectors considered in (Bravais-) cluster generation
MAX_ITER	max_iter	Maximum number of iterations, after which the self-consistency will stop.
MIX_ALPHA	mix_alpha	The mixing parameter for the linear (simple) mixing should be between 0 and 1.
MIX_BROY	mix_broy	If true an extended Broyden mixing scheme [49] is used in the self-consistency.
MIX_LIN	mix_lin	If true simple mixing is used in the self-consistency.
N_IN_LIN	n_init_lin	Number of simple mixing steps before using Broyden mixing (only relevant if mix_broy=true).
NC_IN_UC	nc_in_uc	If true the directions of the magnetic moments can be set by hand.
NC_ANGLES	nc_angles	The angles for the magnetic d-moments for each basis atom. The first value in the row should be the Θ angle, whereas ϕ is saved as second value. The angles should be enterd in degrees (not larger than 180°). If nc_in_uc=T and blo_wave=T these angles play the role of additional phase angles!
NCMAG	nc_mag	If true non-collinear magnetism can be treated. Whether spin-spiral calculations within the gen. Bloch theorem or setting the directions of the magnetic moments by hand, this has to be still defined.
NUM_QWAY	numpts_qway	Number of q-points for the q-paths.
NUMBASIS	numbasis	Number of basis atoms.
NUMCUTBON	num_cut_bonds[1]	The number of removed bondings.
NUMELUC	numelectrons_uc	Number of electrons in the unit cell.
NUMENDOS	num_energy_DOS	Number of energy points for the energy-mesh in the DOS-calculation (see subroutine calc_DOS).
NUMPTSWAY	numptsway	Number of k-points for each k-path.
NUMQWAYS	numendpts_qway	Number of partial ways in the creation of the q-path
NUMWAYS	numendpts	Number of partial ways in the creation of the k-path

Continued...

Keyword	Variable name	Short description
ONLYNEXTN	onlynextn[1]	If true only the nearest neighbours are considered within the hopping calculation. Note that for the NRL-TB parametrization this should be deactivated.
ONSITES, etc.	onsite_s, etc.	The on-site energies of the s-, p- and d-orbitals for all basis atoms. In the case of the NRL-TB parametrization these keywords are inactive.
OSS_SIGMA, etc.	Soss_sigma, etc.	These are the spin-dependent Slater-Koster parameters for the description of the overlap matrix. For more details see Svss_sigma in this reference list.
OVLAP	ovlap	If true the generalized eigenvalue problem with an overlap matrix will be solved. This is recommended for the NRL-TB parametrization.
PAPA_BINA	papa_binary	If true the NRL-TB parametrization for binaries will be used (see eq. 2.25).
QPOIBZ	npoiqbz	Number of q-points in the full q-mesh
RCUTOFF	R_cutoff	The radius of the sphere in units of the lattice constant a in which the neighbours around a basis atom should be considered in the cluster generation in the subroutine clsgen99.
RESTA_IT	restart_iteration	Specifies the iteration step with corresponding starting charges and exchange energies in the file TB_SC_values.dat from which the self-consistency should start again.
RESTA_SC	restart_sc	If true the self-consistency starts with certain starting values for the mulliken-charges and exchange energies not from the inputcard but rather from the file TB_SC_values.dat, which contains the charges and exchange energies of each iteration step from an earlier calculation.
RMAXIMAL	RMAX1[1]	This radius (in units of a) is involved in the definition of a rough pre-constructed cluster consisting of Bravais vectors. It is only important that the radius is not too small.

Continued...

Keyword	Variable name	Short description
RNEIGH	Rneigh[1]	This defines the cutoff distance in units of the lattice constant a from which the hopping parameters will be set to zero.
ROT_SPIN	new_gaxis	The new direction for the magnetic moments,after performing the global spin-rotation (set_gaxis=true).
s_Orb, px_Orb etc.	orbitals[1]	With the help of these keywords the used (or not used) orbitals for each basis atom can be specified.
SELF_COND	sc_cond	The condition ϵ for ending the self-consistent loop: $\lvert v_{\text{new}} - v_{\text{old}}\rvert \leq \epsilon$.
SELF_U	loc_charge_U	The local charge neutrality parameters U_{LCN} for two atom-types. More than 2 atom-types should not be treated with the code.
SET_GAXIS	set_gaxis	If true a global spin-rotation is performed, so that the magnetic moments point along new_gaxis(:). Note that this rotation should not be combined with nc_in_uc=true.
SKP_SPIN	skpspin	If true the SKPs can be defined spin-dependent. Both $V_{ll'm}^{\sigma\sigma}$ and $V_{ll'm}^{\sigma\sigma'}$ can be defined. This does not work within the NRL-TB parametrization.
SKPVARY	skpvary	If true the NRL-TB parametrization [3] is used to describe the non-magnetic system.
SMEAR_MAG	smear_mag	If true a Fermi-Dirac smearing function is used to calculate charges, DOS etc.
SOC-Parameter	soc_p, soc_d	The spin-orbit coupling parameters ξ_i (see eq. 2.61) in eV for all basis atoms.
SOC_PERTU	soc_pertubation	If true spin-orbit coupling is taken into account for a spin-spiral calculation by using 1st order perturbation theory.
SOCLOG	soclog	If true spin-orbit-coupling is taken into account. In principle the logical determines, if the Hamiltonian is diagonalized in the full spin-frame or separately in spin-up and spin-down. Therefore this logical has to be also true for treating non-collinear magnetism.

Continued...

Keyword	Variable name	Short description
SPECBONDS	cluster_center, xx, yy, zz[1]	The specified direction of the bondings within the cluster which should be removed.
SPECORBAT	spec_at, spec_orb	Specifies the basis atom and orbital, for which the partial DOS should be calculated in the case log_spec_orbat=true.
SPINLOG	spinlog	If true the electron spin is considered by introducing a spin-frame for the Hamiltonian. This is necessary to treat magnetism!
STON_PARA	stoner_para	The *s*-, *p*- and *d*-orbital Stoner parameters for maximal two atom-types. More than 2 atom-types should not be treated with the code.
TB_SELFC	tb_sc	If true the self-consistent tight-binding scheme is used (see fig. 2.9).
TYP_BASAT	type_basatom	The atom type of the basis atoms, needed for the NRL-TB parametrization for binaries to assign the correct parameter set (see also the file input_SKP_papakonst).
U_CONST	U_con	The Lagrange parameter of the constraint for pinning the Θ angle of the magnetic moments. It should be large enough to ensure a effective pinning. A value of about 5 meV is reasonable.
VSS_SIGMA, etc.	Svss_sigma, etc.	The spin-dependent Slater-Koster parameters for all possible bondings within the unit cell. The energies are in the units of the user's choice in the inputcard. In the case of the NRL-TB parametrization these arrays become unimportant.
XC_ENERGY	xcenergy	The exchange energies for all basisatoms and *s*, *p* and *d* orbitals.
ZATOM	atomic_number	The atomic numbers of the basis atoms in the crystal. Unlike in *ab-initio* codes, the atomic number is rather unimportant in this program. It can be used to give the basis atom a "name".

E Appendix - List of variables

This appendix lists all input- and output-variables of the described subroutines in the section 6.2. For each variable the name, the data-type including the dimensions, the occurrence in the subroutines and an usually very short description is presented. Together with the chapters 2 and 6 this should give a rather detailed description of the JuTiBi code. Therefore this appendix is rather unimportant for users, who only want to use the code to calculate and do not have the intention to understand the code in detail or even want to modify it for their purposes. For all others I will give some more informations how to work with the list.

- The variable names are alphabetically sorted from `A` to `Z`.
- For the multi-dimensional arrays I use the term `array`(:,x) (or `array`(x,:)) to indicate that an explained property is located in the first entry (or second entry) of the array.

Variable name	Type	Occurrence	Short description
abclatc	real*8(3)	findgrp, bzirr3d, rrgen, clsgen99, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, sc_mixing, calc_final_charges, calc_total_energy	The lattice constants in the form a, b and c.
abravais	real*8(3,3)	lattix_TB, findgrp, bzirr3d, onedim_kmesh	The Bravais vectors are stored in the columns of this array. Note that the Bravais vectors are in units of (a,b,c), depending on the cartesian component.
abravais_au	real*8(3,3)	findgrp, onedim_kmesh, rrgen, calc_total_energy, create_input_Jij	The Bravais vectors are stored in the columns of this array. Note that the Bravais vectors contain the lattice constants a,b and c.
abs_mag_d	real*8(numbasis,max_iter)	protohamiltonian, sc_mixing, calc_final_charges, calc_total_energy, create_input_Jij	The absolute value of the magnetic d-moment in μ_B for all basis atoms (abs_mag_d(:,x)) and all iterations (abs_mag_d(x,:)).
alatc	real*8(3)	lattix_TB, findgrp, onedim_kmesh, create_papa_hopping, create_H0_SKP_by_hand, create_Hk_papa, export_data_xcrysden	The lattice constants in the form a, $\frac{b}{a}$ and $\frac{c}{a}$.

Continued...

Variable name	Type	Occurrence	Short description
atomic_number	real*8(numbasis)	clsgen99	The atomic numbers of the basis atoms in the crystal. Unlike in *ab-initio* codes, the atomic number is rather unimportant in this program. It can be used to give the basis atom a "name".
bandenergies	real*8(dim_ eigenvec)	Hamilton_diag, save_ee_ev, save_ee_ev_on_harddisk	The bandenergies of the k-dependent Hamiltonian for a special k-point.
basis	real*8(3,numbasis)	lattix_TB, findgrp, onedim_kmesh, protohamiltonian, moments_local_to_global, create_Hmagnetic, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk	The vectors for the basis atoms in the unit cell are saved columnwise in this array. Note that the vectors of the basis atoms are saved in cartesian coordinates regardless of the logical CARTESIAN in the inputcard. The vectors are given in units of (a,b,c) depending on the cartesian component.
basis_au	real*8(3,numbasis)	findgrp, onedim_kmesh, clsgen99, transform_DOS_local, calc_DOS	The vectors for the basis atoms in the unit cell are saved columnwise in this array. Note that the vectors of the basis atoms are saved in cartesian coordinates regardless of the logical CARTESIAN in the inputcard. The vectors contain the lattice constants a, b and c.

Continued...

Variable name	Type	Occurrence	Short description
`blo_q`	real*8(3)	protohamiltonian, moments_local_to_global, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, sc_mixing, calc_final_charges, calc_total_energy, create_input_Jij, transform_DOS_local, calc_DOS	The spiral vector q of a spin-spiral.
`blo_theta`	real*8	protohamiltonian, moments_local_to_global, create_Hmagnetic, sc_mixing, calc_total_energy, create_input_Jij, transform_DOS_local, calc_DOS	The cone angle Θ of the spin-spiral in rad.

Continued...

Variable name	Type	Occurrence	Short description
blo_wave	log	readdim, protohamiltonian, moments_local_to_global, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, sc_mixing, calc_total_energy, transform_DOS_local, calc_DOS	If true a spin-spiral is considered within the gen. Bloch theorem.
bondcut	log	readdim	If true the user can cut out some specified bondings of the crystal.
CHAR	char*80	Ioinput	Searching ILINE lines under a keyword CHAR_KEY the character CHAR contains the first 80 characters in this line of the inputcard.
CHAR_KEY	char*10	Ioinput	The keyword, which should be searched in the inputcard to import the input values.
char_log	char*1	create_papa_hopping, create_Hk_papa	This character is necessary to differ between the construction of the Hamiltonian (char_log='H') or the overlap matrix (char_log='O') within the Slater-Koster scheme.
charge	real*8(numbasis, lmax,2)	calc_charges, sc_mixing, calc_final_charges	The spin-up (:,:,1) and spin-down (:,:,2) elements of the density matrix for all atoms (:,x,x) and orbitals (x,:,x) via eq. 2.48.
cluster	real*8(3,max_num_ cluster)	rrgen, clsgen99	The Bravais vectors within the sphere created in the subroutine rrgen.

Continued...

Variable name	Type	Occurrence	Short description
cluster_atom_ bravais	int(max_number_ cluster_ atoms,numbasis)	clsgen99	cluster_atom_bravais(:,i) contains the information about the corresponding Bravais vector for all atoms within the cluster around basis atom i.
cluster_atom_index	int(max_number_ cluster_ atoms,numbasis)	clsgen99, create_papa_hopping, create_H0_SKP_by_hand, create_Hk_papa	cluster_atom_index(:,i) contains the information about the type of basis atom for all atoms within the cluster around basis atom i.
cross_charge	complex*16(numbasis, lmax)	calc_charges, sc_mixing, calc_final_charges	The non-diagonal element of the density matrix, containing the spin-crossing charges, for all atoms (:,x) and orbitals (x,:) via eq. 2.48.
dim_eigenvec	int	get_eigenvectors, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	The dimension of the eigenvectors. In the case soclog=true dim_eigenvec=2*dimHamilton, whereas for soclog=false dim_eigenvec=dimHamilton.
dim_kmesh_ev	int	save_ee_ev, save_ee_ev_on_harddisk, calc_Fermi_energy, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	The dimension of the array, which contains all eigenenergies. In the case spinlog=true dim_kmesh_ev=2*dimHamilton*numkmesh, in the case spinlog=false dim_kmesh_ev=dimHamilton*numkmesh.
dim_sc	int	sc_mixing	The dimension of the self-consistent problem. For soclog=true it is 4*numbasis, whereas for soclog=false it is 2*numbasis.

Continued...

Variable name	Type	Occurrence	Short description
`dimH_loc`	int	create_H_loc_neut	The dimension of the created local charge neutrality part of the Hamiltonian. In the case `soclog=false`: `dimH_loc`=`dimHamilton`, whereas for `soclog=true` `dimH_loc`=`2*dimHamilton`.
`dimHamilton`	int	readdim, protohamiltonian, proto_SOC, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, create_H_loc_neut, global_spin_rot, Hamilton_diag, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, calc_charges, calc_final_charges, calc_total_energy, export_data_xcrysden, transform_DOS_local, calc_DOS	The dimension of the Hamiltonian without spin! Therefore the dimension is `2*dimHamilton` for magnetic systems.
`dimkway`	int	readdim, create_k_way	Total number of k-points along the full k-path
`dimlattice`	int	lattix_TB, rrgen, create_q_way, save_ee_ev, save_ee_ev_on_harddisk, create_k_way	The dimension of the Bravais lattice.
`dimqway`	int	create_q_way, calc_total_energy	Total number of q-points along the full q-path

Continued...

Variable name	Type	Occurrence	Short description
distance	real*8(max_number_cluster_atoms,numbasis)	protohamiltonian, create_papa_hopping	The distances of the atoms in the neighbour shells from the corresponding basis atom. The distances are in units of a, therefore $\frac{b}{a}$ and $\frac{c}{a}$ are contained.
dos_local	log	readdim, calc_DOS	If true the DOSs are calculated in the local spin-frame of the atoms.
Dosorbat	log	readdim, calc_DOS	If true the partial DOS is calculated (i.e. the atom and orbital resolved DOS).
eigenvectors, eigenvectors_mod	complex*16(dim_eigenvec)	get_eigenvectors, transform_DOS_local	These arrays contain the ii-th eigenvector of the array kmesh_eigenvectors and kmesh_eigenvectors_mod, where ii is an integer imported by the subroutine get_eigenvectors.
faclor	int	readdim, calc_DOS	In a region of 2*faclor*lor_width around an energy eigenstate the DOS-contribution is calculated, whereas outside this region the DOS-contribution is approximated to zero.
fat_bands	log	readdim	If true the weights of the eigenvectors for the band structure calculation is also calculated and written out. These weights can be used for example in fat-band representations.
fermi_broadening	real*8	readdim, calc_Fermi_energy, calc_total_energy	The broadening $k_B \cdot T$ in the Fermi-Dirac smearing function in eV.
fermi_dirac	real*8(dim_kmesh_ev)	calc_charges, calc_final_charges, calc_total_energy	The Fermi Dirac smearing function for all eigenenergies.

Continued...

Variable name	Type	Occurrence	Short description
`fermi_energy`	real*8	calc_Fermi_energy, calc_total_energy, export_data_xcrysden	The Fermi energy of the system (in eV).
`flat_spiral`	log	protohamiltonian, moments_local_to_global, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, sc_mixing, calc_total_energy, create_input_Jij, transform_DOS_local, calc_DOS	If true a spin-spiral rotating in the $x-z$-plane is calculated. Note that `blo_theta` has to be zero in this case. Note that this form of spiral is necessary to investigate DMI in surface systems.
`H_loc_charge_neutr`	complex*16(`dimH_loc`,`dimH_loc`)	create_H_loc_neut	The local charge neutrality part of the Hamiltonian constructed via eq. 2.44. For more details see section 2.5.
`H_mag`	complex*16 (`2*dimHamilton`, `2*dimHamilton`)	create_Hmagnetic	The magnetic part of the Hamiltonian constructed via the methods explained in section 2.6 and 2.8. Note that a constraint for the Θ angle of the magnetic moments is also included , if `log_theta_constraint`=true.

Continued...

Variable name	Type	Occurrence	Short description
`hambasis`	int(dimHamilton)	protohamiltonian, proto_SOC, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, create_H_loc_neut, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, calc_charges, calc_final_charges, calc_total_energy, transform_DOS_local, calc_DOS	This array contains the connection between the type of basis atom and the row of the Hamiltonian. For example `hambasis(3)=2` would mean that the third row of the Hamiltonian is belonging to the basis functions centered on the 2nd basis atom in the unit cell.
`Hamiltonian`	complex*16 (`dimHamilton`, `dimHamilton`)	create_H0_SKP_by_hand, create_Hk_papa, Hamilton_diag, save_ee_ev, save_ee_ev_on_harddisk	First one spin-block of the Hamiltonian $\mathcal{H}_0$. Which spin-block is depending on `ispin`. Later Hamiltonian is extended to the full Hamiltonian in one spin-channel in the case `soclog=false`, whereas for the case `soclog=true` the array `Hamiltonian_wS` will be the full Hamiltonian. After the diagonalization the eigenvectors are saved columnwise in this array.
`Hamiltonian_wS`	complex*16 (`2*dimHamilton`, 2*dimHamilton)	global_spin_rot, Hamilton_diag, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk	The full Hamiltonian in spin-space (only used for `soclog=true`). After the diagonalization the eigenvectors are saved columnwise in this array.

Continued...

Variable name	Type	Occurrence	Short description
`hamorb`	int(`dimHamilton`)	protohamiltonian, proto_SOC, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, calc_charges, calc_total_energy, calc_DOS	This array contains the connection between the orbital-type and the row of the Hamiltonian. For example `hamorb(4)=5` would mean that the 4th row of the Hamiltonian is belonging to a d_{xy} basis function. Note that $1: s$, $2: p_x$, $3: p_y$, $4: p_z$, $5: d_{xy}$, $6: d_{xz}$, $7: d_{yz}$, $8: d_{x^2-y^2}$, $9: d_{z^2}$.
`hampos`	int(`numbasis`,`lmax`)	protohamiltonian, create_H0_SKP_by_hand, create_Hk_papa, calc_total_energy, calc_DOS	This array connects the row of the Hamiltonian with the corresponding description within the basis of the atomic orbitals, therefore the type of orbital and the basis atom. For example `hampos(1,3)=3` would mean that the 3rd row belongs to the first basis atom with a p_y orbital. In principle `hampos` acts as inverse function of the arrays `hambasis` and `hamorb`.
`hard_disk`	log	readdim, get_eigenvectors, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	If true the eigenvectors and modified eigenvectors (see eq. 2.32) are saved onto the hard disk, instead in the memory. ATTENTION: This is not suited for very large systems.
`i_iteration`	int	moments_local_to_global, create_Hmagnetic, create_H_loc_neut, sc_mixing	Numbers the iteration step.
`i_q_vec`	int	calc_total_energy, create_input_Jij	Numbers the q-vector in the do-loop over the q-points.

Continued...

Variable name	Type	Occurrence	Short description
i_q_vec_max	int	calc_total_energy, create_input_Jij	The maximal number of q-vectors in the do loop over the q-points
IER	int	Ioinput	Needed for error messages in the subroutine Ioinput.
IFILE	int	Ioinput	Specifies the assigned number for the inputcard file.
ikmesh	int	save_ee_ev, save_ee_ev_on_harddisk	Running index of the do loop over the k-points in the k-mesh.
ILINE	int	Ioinput	ILINE specifies the line under the keyword CHAR_KEY, where the subroutine Ioinput should search for the input values.
index_kmesh_ energies	int(dim_kmesh_ev)	sort_energies, calc_DOS	This array contains the positions of the sorted array kmesh_energies (see description of routine sort_energies).
irr	log	readdim, bzirr3d, onedim_kmesh	If true the irreducible part of the k-mesh is calculated.
irr_q	log	readdim, bzirr3d, onedim_kmesh	If true the irreducible part of the q-mesh is calculated.
ispin	int	create_H0_SKP_by_hand, create_Hk_papa, save_ee_ev, save_ee_ev_on_harddisk	Numbers the spin-channel; $ispin = 1$ corresponds to the majority spin, $ispin = 2$ to the minority spin. In the case of flat_spiral=true, there is also a $ispin = 3$, which corresponds to the calculation of a non-diagonal spin-block for $\mathcal{H}_0$.

Continued...

Variable name	Type	Occurrence	Short description
kmesh_eigenvectors	complex*16 (dim_eigenvec, dim_kmesh_ev)	save_ee_ev, save_ee_ev_on_harddisk, get_eigenvectors, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	All eigenvectors saved as columns for all k-points of the k-mesh. The eigenvectors are saved in a special order, which is explained in detail in the description of the subroutine save_ee_ev. For hard_disk=true the eigenvectors are not saved onto this array, but rather on the hard disk into the file eigenvectors.unformatted.
kmesh_ eigenvectors_mod	complex*16 (dim_eigenvec, dim_kmesh_ev)	save_ee_ev, save_ee_ev_on_harddisk, get_eigenvectors, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	This are all by the overlap matrix modified eigenvectors, which are needed to obtain the Mulliken charges. For more details see description of kmesh_eigenvectors.
kmesh_energies	real*8(dim_kmesh_ ev)	save_ee_ev, save_ee_ev_on_harddisk, sort_energies, calc_Fermi_energy, calc_total_energy, export_data_xcrysden, calc_DOS	All eigenenergies for all k-points of the k-mesh. The eigenvalues are saved in a special order, which is explained in detail in the description of the subroutine save_ee_ev.
kp	real*8(3,npoibz)	bzirr3d	The k-points in cartesian representation of the (irreducible) k-mesh. Note that the k-points are in units of 2π, but contain the lattice constants a, b and c.

Continued...

Variable name	Type	Occurrence	Short description
kp_1dim	real*8(npoibz)	onedim_kmesh	The k-points in cartesian representation of the (irreducible) k-mesh in the 1-dim. case. Note that the k-points are in units of 2π, but contain the lattice constant a.
kpoint	real*8(3)	create_H0_SKP_by_hand, create_Hk_papa	The k-point containing 2π and the lattice constants a, b and c, for which the Hamiltonian will be diagonalized.
kway	real*8(3,dimkway)	create_k_way	The k-vectors along the k-path. Note that the k-vectors are in units of 2π, but contain the lattice constants a, b and c.
kway_param	real*8(dimkway)	create_k_way	A length-parametrization along the k-path, in order to simplify the plotting of the band structure. Note that the length parametrization does not contain the lattice constants a, b and c.
L_x, L_y, L_z	complex*16 (2*dimHamilton, 2*dimHamilton)	proto_SOC, calc_final_charges	The cartesian components of the angular moment operator in representation of the atomic orbitals. For more details see appendix A.1.
last_it	int	calc_final_charges, calc_total_energy	The last iteration step in which the convergence condition sc_cond is fulfilled.
lmax	int	readdim, protohamiltonian, create_H0_SKP_by_hand, create_Hk_papa, calc_charges, sc_mixing, calc_final_charges, calc_total_energy, calc_DOS	$\sum_l(2 \cdot l + 1)$, l: angular moment quantum number ; for treating s, p and d-orbitals leave it to the value 9.

Continued...

Variable name	Type	Occurrence	Short description
loc_charge_U	real*8(2)	readdim, create_H_loc_neut, calc_total_energy	The local charge neutrality parameters U_{LCN} for two atom-types. More than 2 atom-types should not be treated with the code.
log_change_fermi_energy	log	readdim, calc_total_energy	If true the Fermi energy will be recalculated after adding the 1st order contribution from SOC in the spin-spiral calculation with soc_pertubation=true.
log_fermi_surface	log	readdim	If true a file for the external program XCrysDen is created to calculate the Fermi surface of a 3-dim. system.
log_q_way	log	readdim, calc_total_energy	If true the magnon dispersion along a q-way in the Brillouin zone can be calculated.
log_qmesh	log	readdim, calc_total_energy	If true the energies of spin-spirals with q-values from the full or irr. Brillouinzone are calculated. This is necessary to construct the file jenerg.dat, which can be used to calculate the Heisenberg exchange-coupling parameters.
log_spec_orbat	log	readdim, calc_DOS	If true only one specific partial DOS for an atom and an orbital is calculated.
log_theta_constraint	log	readdim, create_Hmagnetic, calc_total_energy	If true a constraint is used to pin the Θ angle of the magnetic moments. It works for spin-spirals as also for the case nc_in_uc=true.
lor_width	real*8	readdim, calc_DOS	The width in eV of the Lorentzians used in the DOS-calculation (see eq. 2.30 and subroutine calc_DOS).

Continued...

Variable name	Type	Occurrence	Short description
mag_x, mag_y, mag_z	real*8(numbasis,max_iter)	protohamiltonian, moments_local_to_global, create_Hmagnetic, sc_mixing, calc_final_charges, calc_total_energy	The magnetic d-moments in μ_B of all basis atoms (mag_x(:,x)) and all iterations (mag_x(x,:)). mag_x is the moment in x-direction, mag_y the moment in y-direction and mag_z the moment in z-direction. To calculate this moments, see eqs. 2.50- 2.52.
max_iter	int	readdim, protohamiltonian, moments_local_to_global, create_Hmagnetic, create_H_loc_neut, sc_mixing, calc_final_charges, calc_total_energy	Maximum number of iterations, after which the self-consistency will be canceled.
max_nnear	int	create_papa_hopping, create_Hk_papa	The maximum of the values nnearneig(:), therefore the number of neighbours in the cluster, which has the maximal number of neighbours.
max_num_cluster	int	readdim, rrgen, clsgen99	Maximum number of Bravais vectors considered in (Bravais-) cluster generation
max_number_cluster_atoms	int	readdim, clsgen99, cut_bondings, protohamiltonian, create_papa_hopping, create_H0_SKP_by_hand, create_Hk_papa	Maximum number of atoms within a cluster: numbasis*max_num_cluster.
mix_alpha	real*8	readdim, sc_mixing	The mixing parameter for the linear (simple) mixing should be between 0 and 1.

Continued...

Variable name	Type	Occurrence	Short description
mix_broy	log	readdim, sc_mixing	If true an extended Broyden mixing scheme [49] is used in the self-consistency.
mix_lin	log	readdim, sc_mixing	If true simple mixing is used in the self-consistency.
mullikan_charge	real*8(numbasis, lmax,2)	calc_charges, sc_mixing, calc_final_charges	The spin-up (:,:,1) and spin-down (:,:,2) elements of the density matrix for all atoms (:,x,x) and orbitals (x,:,x) via eq. 2.54.
mullikan_charge_ start	real*8(numbasis)	protohamiltonian, create_H_loc_neut, calc_total_energy	The desired reference charge within the local charge neutrality description for all basis atoms. See eq. 2.44 for more details.
mullikan_cross_ charge	complex*16(numbasis, lmax)	calc_charges, calc_final_charges	The non-diagonal element of the density matrix, containing the spin-crossing charges, for all atoms (:,x) and orbitals (x,:) via eq. 2.54.
n_init_lin	int	readdim, sc_mixing	Number of simple mixing steps before using Broyden mixing (only relevant if mix_broy=true).
nc_angles	real*8(numbasis,2, max_iter)	protohamiltonian, moments_local_to_global, create_Hmagnetic, sc_mixing, calc_final_charges, calc_total_energy, transform_DOS_local, calc_DOS	The angles for the magnetic d-moments for each basis atom (nc_angles(:,x,x)) and each iteration-step (nc_angles(x,x,:)). The first values nc_angles(x,1,x) contain the Θ angles, whereas ϕ is saved as nc_angles(x,2,x). The angles are saved in rad.

Continued...

Variable name	Type	Occurrence	Short description
nc_in_uc	log	readdim, protohamiltonian, moments_local_to_global, create_Hmagnetic, calc_total_energy, transform_DOS_local, calc_DOS	If true the directions of the magnetic moments can be set by hand.
nc_mag	log	readdim, protohamiltonian, create_Hmagnetic, sc_mixing	If true non-collinear magnetism can be treated. Whether spin-spiral calculations within the gen. Bloch theorem or setting the directions of the magnetic moments by hand, this has to be still defined.
new_gaxis	real*8(3)	readdim, proto_SOC	The new direction for the magnetic moments,after performing the global spin-rotation (set_gaxis=true).
nkp	int	bzirr3d, onedim_kmesh	The number of k-points in the (irreducible) k-mesh. If irr=false *nkp* gives the number of k-points in the full k-mesh and is therefore the same as numkmesh.
nkxyz	int(3)	readdim, bzirr3d, onedim_kmesh, save_ee_ev, save_ee_ev_on_harddisk, export_data_xcrysden	Number of k-points along the reciprocal Bravais vector directions in the k-mesh division.
nnearneigh	int(numbasis)	protohamiltonian, create_papa_hopping, create_H0_SKP_by_hand, create_Hk_papa	Number of neighbours within the defined cutoff-radius RNEIGH in the inputcard for all neighbour shells around the basis atoms.

Continued...

Variable name	Type	Occurrence	Short description
npoibz	int	readdim, bzirr3d, onedim_kmesh	Number of k-points in the full k-mesh.
npoiqbz	int	readdim, bzirr3d, onedim_kmesh	Number of q-points in the full q-mesh
nqp	int	bzirr3d, onedim_kmesh	The number of q-points in the (irreducible) q-mesh. If `irr=false` `nqp` gives the number of q-points in the full q-mesh and is therefore the same as `numqmesh`.
nqxyz	int(3)	readdim, bzirr3d, onedim_kmesh	Number of q-points along the reciprocal Bravais vector directions in the q-mesh division
num_atom_cluster	int(numbasis)	clsgen99, cut_bondings, protohamiltonian	`num_atom_cluster`(i) is the number of atoms in the cluster generated around the i-th basis atom.
num_cluster	int	rrgen	Number of Bravais vectors within the sphere created in the subroutine rrgen.
num_energy_DOS	int	readdim, calc_DOS	Number of energy points for the energy-mesh in the DOS-calculation (see subroutine calc_DOS).

Continued...

Variable name	Type	Occurrence	Short description
numbasis	int	readdim, lattix_TB, findgrp, onedim_kmesh, clsgen99, cut_bondings, protohamiltonian, moments_local_to_global, SKP_by_hand, create_papa_hopping, proto_SOC, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, create_H_loc_neut, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, calc_charges, sc_mixing, calc_final_charges, calc_total_energy, create_input_Jij, transform_DOS_local, calc_DOS	Number of basis atoms
numbasis_max	int	readdim, clsgen99	Maximum number of basis atoms in the system
numelectrons	real*8	save_ee_ev, save_ee_ev_on_harddisk, calc_total_energy	The total number of electrons in the crystal.
numelectrons_uc	real*8	readdim, save_ee_ev, save_ee_ev_on_harddisk, calc_Fermi_energy	Number of electrons in the unit cell.

Continued...

Variable name	Type	Occurrence	Short description
numendpts	int	readdim, create_k_way	Number of partial ways in the creation of the k-path
numendpts_qway	int	readdim, create_q_way	Number of partial ways in the creation of the q-path
numkmesh	int	bzirr3d, onedim_kmesh, save_ee_ev, save_ee_ev_on_harddisk, calc_Fermi_energy, calc_charges, sc_mixing, calc_final_charges, calc_total_energy, export_data_xcrysden, calc_DOS	The number of k-points in the full k-mesh.
numpts_qway	int(numendpts_qway)	readdim, create_q_way	i-th entry: Number of q-points for the i-th partial way
numptsway	int(numendpts)	readdim, create_k_way	i-th entry: Number of k-points for the i-th partial way
numqmesh	int	bzirr3d, onedim_kmesh	The number of q-points in the full q-mesh.
numsym	int	findgrp, bzirr3d, onedim_kmesh	The number of symmetries of the real space lattice. In the determination of the symmetries the non-equivalence of the basis atoms is a basic assumption.
onsite_s, onsite_p, onsite_d	real*8(numbasis)	SKP_by_hand, create_H0_SKP_by_hand	The on-site energies of the s-, p- and d-orbitals for all basis atoms.
oss_sigma (representative for all o$\mu\nu$_m)	real*8(numbasis, numbasis)	SKP_by_hand, create_H0_SKP_by_hand	These are the Slater-Koster parameters for the description of the overlap matrix. For more details see vss_sigma in this reference list.

Continued...

Variable name	Type	Occurrence	Short description
`Overlapian`	complex*16 (`dimHamilton`, `dimHamilton`)	create_H0_SKP_by_hand, create_Hk_papa, create_H_loc_neut, Hamilton_diag	First one spin-block of the overlap matrix. Which spin-block is depending on `ispin`. Later `Overlapian` is extended to the full overlap matrix in one spin-channel in the case `soclog=false`, whereas for the case `soclog=true` the array `Overlapian_wS` will be the full overlap matrix.
`Overlapian_Save`	complex*16 (`dimHamilton`, `dimHamilton`)	save_ee_ev, save_ee_ev_on_harddisk	The copy of the overlap matrix in the case `soclog=false` just before the routine Hamilton_diag. This is necessary because the overlap matrix is needed to calculate the modified eigenvalues to obtain the Mulliken charges and it is overwritten in the diagonalization routine.
`Overlapian_wS`	complex*16 (`2*dimHamilton`, `2*dimHamilton`)	create_H_loc_neut, global_spin_rot, Hamilton_diag	The full overlap matrix in spin-space (only used for `soclog=true`).
`Overlapian_wS_Save`	complex*16 (`2*dimHamilton`, `2*dimHamilton`)	save_ee_ev, save_ee_ev_on_harddisk	The copy of the overlap matrix in the case `soclog=true` just before the routine Hamilton_diag. This is necessary because the overlap matrix is needed to calculate the modified eigenvalues to obtain the Mulliken charges and it is overwritten in the diagonalization routine.
`ovlap`	log	protohamiltonian, SKP_by_hand, Hamilton_diag	If true the generalized eigenvalue problem with an overlap matrix will be solved. This is recommended for the NRL-TB parametrization.

Continued...

Variable name	Type	Occurrence	Short description
papa_binary	log	protohamiltonian, create_papa_hopping	If true the NRL-TB parametrization for binaries will be used (see eq. 2.25).
papa_hopping	real*8(max_nnear,numbasis,45)	create_papa_hopping, create_Hk_papa	This array contains all hopping elements for the bondings of each basis atom (x,:,x) to each of its neighbours (:,x,x) for all combinations of orbitals (x,x,:) within the Slater-Koster transformations.
papa_onsite	real*8(numbasis,3)	create_papa_hopping, create_Hk_papa	papa_onsite contains the on-site energies for each basis atom papa_onsite(:,x) for the s- papa_onsite(:,1), p- papa_onsite(:,2) and d-orbitals papa_onsite(:,3)
pos_cluster_atom	real*8(3,max_number_cluster_atoms,numbasis)	clsgen99, cut_bondings, protohamiltonian, create_H0_SKP_by_hand, create_Hk_papa	This is a very important array, which contains all informations about the neighbour shells for all basis atoms. Therefore pos_cluster_atom(:,:,i) contains the cartesian coordinates (pos_cluster_atom(:,x,i)) of all atoms within R_cutoff (pos_cluster_atom(x,:,i)) around the ith basis atom. Note that the coordinates are in units of the lattice constant a, therefore $\frac{b}{a}$ and $\frac{c}{a}$ are contained. In addition the positions are sorted by the distance, the z-, y- and at last the x-component.
qp	real*8(3,npoiqbz)	bzirr3d	The q-points in cartesian representation of the (irreducible) q-mesh. Note that the q-points are in units of 2π, but contain the lattice constants a, b and c.

Continued...

Variable name	Type	Occurrence	Short description
qp_1dim	real*8(npoiqbz)	onedim_kmesh	The q-points in cartesian representation of the (irreducible) q-mesh in the 1-dim. case. Note that the q-points are in units of 2π, but contain the lattice constant a.
qway	real*8(3,dimqway)	create_q_way	The q-vectors along the q-path. Note that the q-vectors are in units of 2π, but contain the lattice constants a, b and c.
qway_param	real*8(dimqway)	create_q_way, calc_total_energy	A length-parametrization along the q-path, in order to simplify the plotting of the magnon dispersion. Note that the length parametrization does not contain the lattice constants a, b and c.
R_cutoff	real*8	clsgen99	The radius of the sphere in which the neighbours around a basis atom should be considered in the cluster generation in the subroutine clsgen99.
recbravais	real*8(3,3)	lattix_TB, findgrp, bzirr3d, onedim_kmesh	The recipocal Bravais vectors are stored in the columns of this array. Note that the reciprocal Bravais vectors are in units of $2\pi \cdot (1/a, 1/b, 1/c)$, depending on the cartesian component.
recbravais_au	real*8(3,3)	findgrp, onedim_kmesh, export_data_xcrysden	The recipocal Bravais vectors are stored in the columns of this array. Note that the reciprocal Bravais vectors are in units of 2π but the lattice constants a,b and c are contained in the values.

Continued...

Variable name	Type	Occurrence	Short description
`restart_iteration`	int	readdim, protohamiltonian	Specifies the iteration step with corresponding starting charges and exchange energies in the file `TB_SC_values.dat` from which the self-consistency should start again.
`restart_sc`	log	readdim, protohamiltonian	If true the self-consistency starts with certain starting values for the mulliken-charges and exchange energies not from the `inputcard` but rather from the file `TB_SC_values.dat`, which contains the charges and exchange energies of each iteration step from an earlier calculation.
`rot_spinonian`, `rot_spinonian_inv`	complex*16(2,2)	proto_SOC, global_spin_rot	The spin-rotation matrices, which rotate the magnetic moments from the z-direction into the direction `new_gaxis`(:). This matrices become important only in the case `set_gaxis=true`.
`rotmat`	real*8(64,3,3)	pointgrp, findgrp, bzirr3d	The 64 rotation matrices of all lattice groups are contained in this array.
`rotname`	char*10(64)	pointgrp, findgrp	The names for all 64 symmetry rotation matrices for the lattice groups after the convention of [51].
`sc_cond`	real*8	readdim	The condition ϵ for ending the self-consistent loop: $\lvert v_{\text{new}} - v_{\text{old}} \rvert \leq \epsilon$.

Continued...

Variable name	Type	Occurrence	Short description
scal_md_ms, scal_md_mp	real*8(numbasis)	calc_final_charges, calc_total_energy	These are the scalar-products between the magnetic d-moment and the magnetic s- and p-moment. They are needed to determine the double counting from the Stoner part (see eq. 2.58).
set_gaxis	log	readdim, proto_SOC	If true a global spin-rotation is performed, so that the magnetic moments point along new_gaxis(:). Note that this rotation should not be combined with nc_in_uc=true.
skpspin	log	readdim, create_H0_SKP_by_hand	If true the SKPs can be defined spin-dependent. Both $V_{ll'm}^{\sigma\sigma}$ and $V_{ll'm}^{\sigma\sigma'}$ can be defined.
skpvary	log	readdim, protohamiltonian	If true the NRL-TB parametrization [3] is used to describe the non-magnetic system.
smear_mag	log	readdim	If true a Fermi-Dirac smearing function is used to calculate charges, DOS etc.
SOC_matrix	complex*16 (2*dimHamilton, 2*dimHamilton)	proto_SOC	The spin-orbit coupling matrix in the tight-binding representation. For more details see section 2.7 and the appendix A.1.
soc_p_pert, soc_d_pert	real*8(numbasis)	protohamiltonian, proto_SOC, calc_total_energy	The spin-orbit coupling parameters ξ_i (see eq. 2.61) in eV for all basis atoms. These special arrays are needed in the case of a spin-spiral calculation within a 1st order perturbation theoretical treatment of SOC.
soc_p, soc_d	real*8(numbasis)	protohamiltonian, proto_SOC	The spin-orbit coupling parameters ξ_i (see eq. 2.61) in eV for all basis atoms.

Continued...

Variable name	Type	Occurrence	Short description
soc_pertubation	log	readdim, protohamiltonian, calc_total_energy	If true spin-orbit coupling is taken into account for a spin-spiral calculation by using 1st order perturbation theory.
soclog	log	readdim, protohamiltonian, save_ee_ev, save_ee_ev_on_harddisk, get_eigenvectors, calc_charges, calc_final_charges, calc_total_energy, calc_DOS	If true spin-orbit-coupling is taken into account. In principle the logical determines, if the Hamiltonian is diagonalized in the full spin-frame or separately in spin-up and spin-down. Therefore this logical has to be also true for treating non-collinear magnetism.
Sonsite_s, Sonsite_p, Sonsite_d	real*8(numbasis,2)	SKP_by_hand, create_H0_SKP_by_hand	The spin-dependent on-site elements of the *s*-, *p*- and *d*-orbitals for all basis atoms. S_onsite(x,1) contains the on-site energies for the majority spin, whereas S_onsite(x,2) contains the minority spin elements. The energies are in the units of the user's choice in the inputcard. In the case of the NRL-TB parametrization these arrays become unimportant.
Soss_sigma (representative for all So$\mu\nu$_m)	real*8(2*numbasis, 2*numbasis)	SKP_by_hand, create_H0_SKP_by_hand	These are the spin-dependent Slater-Koster parameters for the description of the overlap matrix. For more details see Svss_sigma in this reference list.
spec_at	int	readdim, calc_DOS	Specifies the basis atom, for which the partial DOS should be calculated in the case log_spec_orbat=true.

Continued...

Variable name	Type	Occurrence	Short description
`spec_orb`	int	readdim, calc_DOS	Specifies the orbital, for which the partial DOS should be calculated in the case `log_spec_orbat=true`. Note that $1:s$, $2:p_x$, $3:p_y$, $4:p_z$, $5:d_{xy}$, $6:d_{xz}$, $7:d_{yz}$, $8:d_{x^2-y^2}$, $9:d_{z^2}$.
`spinlog`	log	readdim, protohamiltonian, SKP_by_hand, create_H0_SKP_by_hand, calc_charges, export_data_xcrysden, calc_DOS	If true the electron spin is considered by introducing a spin-frame for the Hamiltonian. This is necessary to treat magnetism!
`stoner_para`	real*8(3,2)	readdim, protohamiltonian, create_Hmagnetic, sc_mixing, calc_total_energy	The s-, p- and d-orbital (`stoner_para`(:,x)) Stoner parameters for two atom-types (`stoner_para`(x,:)). More than 2 atom-types should not be treated with the code.
`Svss_sigma` (representative for all Sv$\mu\nu$_`m`)	real*8(`2*numbasis`, `2*numbasis`)	SKP_by_hand, create_H0_SKP_by_hand	The spin-dependent Slater-Koster parameters for all possible bondings within the unit cell (f.ex. `vss_sigma`(1,2) describes the hopping from the first basis atom in the s-orbital to the second basis atom in the s-orbital in the majority spin-channle without a spin-flip, whereas `vss_sigma`(1,`numbasis`+2) would describe the same hopping but with a spin-flip). The energies are in the units of the user's choice in the `inputcard`. In the case of the NRL-TB parametrization these arrays become unimportant.

Continued...

Variable name	Type	Occurrence	Short description
symindex	int(48)	findgrp, bzirr3d	This array contains the assigned numbers of the symmetries. Together with the array rotname the symmetries of the lattice can be given by the convention of [51].
tb_sc	log	readdim	If true the self-consistent tight-binding scheme is used (see fig. 2.9).
total_energy	real*8	create_input_Jij	In this case the total energy of the special q-point, needed for the file jenerg.dat.
total_moment	real*8(numbasis,3, max_iter)	protohamiltonian, moments_local_to_global, create_Hmagnetic, sc_mixing	The total magnetic s-,p- and d- moments (total_moment(x,:,x)) for all basis atoms (total_moment(:,x,x)) and all iterations (total_moment(x,x,:)).
total_mullikan_ charge	real*8(numbasis, max_iter)	protohamiltonian, create_H_loc_neut, sc_mixing, calc_final_charges, calc_total_energy	The total mulliken charge for each basis atom (total_mullikan_charge(:,x)) and each iteration (total_mullikan_charge(x,:)). See eq. 2.35 for more details.
total_mullikan_ moment	real*8(numbasis,3)	sc_mixing	The magnetic s-, p- and d- Mulliken-moments for all basis atoms.
type_baseatom	int(numbasis)	protohamiltonian, create_papa_hopping, create_Hmagnetic, create_H_loc_neut, sc_mixing, calc_total_energy	The atom type of the basis atoms, needed for the NRL-TB parametrization for binaries to assign the correct parameter set (see also the file input_SKP_papakonst).

Continued...

Variable name	Type	Occurrence	Short description
U_con	real*8	readdim, create_Hmagnetic, calc_total_energy	The Lagrange parameter of the constraint for pinning the Θ angle of the magnetic moments. It should be large enough to ensure a effective pinning. A value of about 5 meV is reasonable.
volbz	real*8	bzirr3d, onedim_kmesh, save_ee_ev, save_ee_ev_on_harddisk	The volume of the Brillouinzone in units of $2\pi \cdot a \cdot \ b \cdot c$.
vss_sigma (representative for all v$\mu\nu$_m)	real*8(numbasis, numbasis)	SKP_by_hand, create_H0_SKP_by_hand	The Slater-Koster parameters for all possible bondings within the unit cell (f.ex. vss_sigma(1,2) describes the hopping from the first basis atom in the s-orbital to the second basis atom in the s-orbital). The energies are in the units of the user's choice in the inputcard. In the case of the NRL-TB parametrization these arrays become unimportant.
wtkp	real*8(npoibz)	bzirr3d, onedim_kmesh, save_ee_ev, save_ee_ev_on_harddisk	The weights of the k-points constructed in the (irreducible) k-mesh and saved in the array kp(:,:).
wtqp	real*8(npoiqbz)	bzirr3d, onedim_kmesh	The weights of the q-points constructed in the (irreducible) q-mesh and saved in the array qp(:,:).
xcenergy	real*8(numbasis,3, max_iter)	protohamiltonian, sc_mixing	The exchange energies for all basisatoms (xcenergy(:,x,x)) and s, p and d orbitals (xcenergy(x,:,x)) for all iterations (xcenergy(x,x,:)).

Continued...

Variable name	Type	Occurrence	Short description
xn, yn, zn	real*8(max_number_cluster_atoms, numbasis)	protohamiltonian, create_papa_hopping, create_H0_SKP_by_hand	The direction cosines of all bonding vectors for all neighbour shells around the basis atoms. These values are needed for the Slater-Koster transformations (see table A.1).

F Appendix - Files in the JuTiBi Package

This appendix lists all files contained in the JuTiBi-package:

Fortran-files (.f):
ioinput, readdim, lattix_TB, pointgrp, help_routines, findgroup, bzirr3d, onedim_kmesh, rrgen, clsgen99, cut_bondings, input_SKP, read_sc_output, moments_local_to_global, protohamiltonian, SKP_by_handpapakonst_input, create_papa_hopping, proto_SOC, create_q_way, create_Hmagnetic, create_H0_SKP_by_hand, create_Hk_papa, create_H_loc_neut, global_spin_rot, Hamilton_diag, rotate_EV, save_ee_ev, save_ee_ev_on_harddisk, sort_energies, calc_Fermi_energy, get_eigenvectors, calc_charges, broyden, sc_mixing, calc_final_charges, create_input_Jij, calc_total_energy, export_data_xcrysden, export_data_berrycurv, transform_DOS_local, calc_DOS, create_k_way, JuTiBi_main

Input (in folder input):
inputcard, SKPinput, SKP_input_papakonst

Output (in folder data_temp as .dat files):
bandenergies_up, bandenergies_down, charges, conv_charges, DOS_atom_orbital_resolved_down, DOS_atom_orbital_resolved_up, Eq_1storder_SOC_layer_orbital_resolved, Eq_1storder_SOC_layer_resolved, Eq_1storder_SOC, Eq, fermi_surface.bxsf, basics.berrycurv, hopping.berrycurv, charges.berrycurv, jenerg, kmesh, shells_neighbours, SOC_bandenergies, SOC_DOS_atom_orbital_resolved_down, SOC_DOS_atom_orbital_resolved_up, SOC_Total_DOS, TB_SC_values, Total_DOS_down, Total_DOS_up

Data for the Tutorial (in folder inputcards_for_Tutorial):

folder Copper-bandstructure: bandenergies_Cu, inputcard_Cu_converge, inputcard_Cu, parameter_set_Cu

folder DMI: Eq_1storder_SOC_FePt_chain, inputcard_DMI_Fe_Pt_chain, plot_Eq_1st_order.py

folder Iron-bandstructure: bandenergies_Fe_down, bandenergies_Fe_up, inputcard_Fe, parameter_set_Fe, plot_bands.py

folder Iron-DOS: DOS_atom_orbital_resolved_Fe_down, DOS_atom_orbital_resolved_Fe_up, inputcard_Fe_DOS, plot_DOS.py, TB_SC_values, Total_DOS_Fe_down, Total_DOS_Fe_up, Total_DOS_up_med, Total_DOS_up_smeared, Total_DOS_up_spiky

folder MCA-Femonolayer: inputcard_MCA_Fe_monolayer_x, inputcard_MCA_Fe_monolayer_z

folder Spinspirals-bccFe: Eq_bcc_Fe_ft_90, Eq_bcc_Fe_FT, Eq_bcc_Fe_sc_90, Eq_bcc_Fe_sc, inputcard_bccFe_force_theorem, inputcard_bccFe_self_consistent, plot_bcc_Fe_Eq.py

folder Spinspiral-Fechain: inputcard_gen_BT, inputcard_mag_unitcell

If a file is missing in your JuTiBi-package and you desperately need it, please feel free to contact t.schena@fz-juelich.de!

Bibliography

[1] P. Hohenberg and W. Kohn, Inhomogeneous Electron Gas, Phys. Rev. **136**(3B), B864–B871 (Nov 1964).

[2] W. Kohn and L. J. Sham, Self-Consistent Equations Including Exchange and Correlation Effects, Phys. Rev. **140**(4A), A1133–A1138 (Nov 1965).

[3] M. J. Mehl and D. A. Papaconstantopoulos, Applications of a tight-binding total-energy method for transition and noble metals: Elastic constants, vacancies, and surfaces of monatomic metals, Phys. Rev. B **54**(7), 4519–4530 (Aug 1996).

[4] `http://cst-www.nrl.navy.mil/.`

[5] D. A. Papaconstantopoulos and M. J. Mehl, Tight-Binding Method in Electronic Structure, in *Encyclopedia of Condensed Matter Physics*, edited by G. Bassani, G. Liedl, and P. Wyder, volume 1, pages 194–206, Academic Press, 2005.

[6] D. A. Papaconstantopoulos and M. J. Mehl, The Slater–Koster tight-binding method: a computationally efficient and accurate approach, Journal of Physics: Condensed Matter **15**(10), R413 (2003).

[7] E. C. Stoner, Collective Electron Specific Heat and Spin Paramagnetism in Metals, Proc. R. Soc. London Series A **154**(656) (1936).

[8] E. C. Stoner, Collective Electron Ferromagnetism, Proc. R. Soc. London Series A **165**, 372–414 (1938).

[9] C. Herring, Exchange interactions among itinerant electrons, Academic Press, 1966.

[10] L. M. Sandratskii, Energy Band Structure Calculations for Crystals with Spiral Magnetic Structure, phys. stat. sol. (b) **136**(167) (1986).

[11] L. M. Sandratskii, Symmetry analysis of electronic states for crystals with spiral magnetic order. I. General properties, Journal of Physics: Condensed Matter **3**(44), 8565 (1991).

[12] M. Heide, G. Bihlmayer, and S. Blügel, Describing Dzyaloshinskii-Moriya spirals from first principles, Physica B: Condensed Matter **404**(18), 2678 – 2683 (2009), Proceedings of the Workshop.

[13] M. Heide, *Magnetic domain walls in ultrathin films: Contribution of the Dzyaloshinskii-Moriya interaction*, PhD thesis, RWTH Aachen, 2006.

[14] G. Autès, C. Barreteau, D. Spanjaard, and M.-C. Desjonquères, Magnetism of iron: from the bulk to the monatomic wire, Journal of Physics: Condensed Matter **18**(29), 6785 (2006).

[15] G. Autès, *Transport électronique polarisé en spin dans les contacts atomiques de fer*, PhD thesis, Universite Pierre et Marie Curie - Paris VI, 2008.

[16] T. Schena, Tight-Binding Treatment of Complex Magnetic Structures in Low-Dimensional Systems, Master's thesis, RWTH Aachen, 2010.

[17] S. Baud, C. Ramseyer, G. Bihlmayer, S. Blügel, C. Barreteau, M. C. Desjonquères, D. Spanjaard, and N. Bernstein, Comparative study of ab initio and tight-binding electronic structure calculations applied to platinum surfaces, Phys. Rev. B **70**(23), 235423 (Dec 2004).

[18] J. C. Slater and G. F. Koster, Simplified LCAO Method for the Periodic Potential Problem, Phys. Rev. **94**(6), 1498–1524 (Jun 1954).

[19] W. A. Harrison, *Electronic Structure and the Properties of Solids*, Dover, 1989.

[20] P. B. Allen, J. Q. Broughton, and A. K. McMahan, Transferable nonorthogonal tight-binding parameters for silicon, Phys. Rev. B **34**(2), 859–862 (Jul 1986).

[21] F. Liu, M. R. Press, S. N. Khanna, and P. Jena, Magnetism and local order: Ab initio tight-binding theory, Phys. Rev. B **39**(10), 6914–6924 (Apr 1989).

[22] C. Barreteau, D. Spanjaard, and M. C. Desjonquères, Electronic structure and total energy of transition metals from an *spd* tight-binding method: Application to surfaces and clusters of Rh, Phys. Rev. B **58**(15), 9721–9731 (Oct 1998).

[23] W. A. Harrison, *Elementary Electronic Structure*, World Scientific, 1999.

[24] F. Ercolessi, Lecture notes on Tight-Binding Molecular Dynamics, and Tight-Binding justification of classical potentials, (2005).

[25] C. G. Broyden, A Class of Methods for Solving Nonlinear Simultaneous Equations, Mathematics of Computation **19**(92), 577–593 (1965).

[26] S. Shallcross, L. Nordström, and S. Sharma, Magnetic phase diagrams from non-collinear canonical band theory, Phys. Rev. B **76**(5), 054444 (Aug 2007).

[27] E. I. Rashba, Properties of semiconductors with an extremum loop, Sov. Phys. Solid State **2**(1109) (1960).

[28] T. Moriya, Anisotropic Superexchange Interaction and Weak Ferromagnetism, Phys. Rev. **120**(1), 91–98 (1960).

[29] I. E. Dzyaloshinskii, Thermodynamic theory of "weak" ferromagnetism in antiferromagnetic substances, Sov. Phys. JETP **5**(1259) (1957).

[30] W. Nolting, *Grundkurs Theoretische Physik 5/2: Quantenmechanik - Methoden und Anwendungen*, Springer-Verlag Berlin, 2004.

[31] B. Hardrat, A. Al-Zubi, P. Ferriani, S. Blügel, G. Bihlmayer, and S. Heinze, Complex magnetism of iron monolayers on hexagonal transition metal surfaces from first principles, Phys. Rev. B **79**(9), 094411 (Mar 2009).

[32] B. Zimmermann, Calculation of the Dzyaloshinskii-Moriya Interaction in ultrathin magnetic Films: Cr/W(110), Master's thesis, RWTH Aachen, 2010.

[33] M. Heide, G. Bihlmayer, and S. Blügel, Dzyaloshinskii-Moriya interaction accounting for the orientation of magnetic domains in ultrathin films: Fe/W(110), Phys. Rev. B **78**(14), 140403 (Oct 2008).

[34] K. Nakamura, T. Akiyama, I. Tomonori, and A. J. Freeman, Spin-spiral structures in free-standing Fe(110) monolayers, Journal of Applied Physics **99**(08N501) (2006).

[35] L. D. Landau and E. M. Lifschitz, *Lehrbuch der theoretischen Physik Band III - Quantenmechanik*, Akademie Verlag Berlin, 1979.

[36] R. Gebauer, *Nouvelles méthodes pour le calcul ab-initio des propriétés statiques et dynamiques des matériaux magnétiques*, PhD thesis, ENS Lyon, 1999.

[37] A. R. Mackintosh and O. K. Andersen, The electronic structure of transition metals, Cambridge Univ. Press, 1980.

[38] A. Oswald, R. Zeller, P. J. Braspenning, and P. H. Dederichs, Interaction of magnetic impurities in Cu and Ag, J. Phys. F.: Met. Phys. **15**(193) (1985).

[39] A. I. Liechtenstein, M. I. Katsnelson, V. P. Antropov, and V. A. Gubanov, Local spin density functional approach to the theory of exchange interactions in ferromagnetic metals and alloys, Journal of Magnetism and Magnetic Materials **67**(1), 65 – 74 (1987).

[40] S. Blügel, Exchange interactions, in *Lecture Notes of the 41st Springschool 2010: Electronic Oxides - Correlation Phenomena, Exotic Phases and Novel Functionalities*, edited by S. Blügel, T. Brückel, R. Waser, and C. M. Schneider, Schriften des Forschungszentrums Jülich Reihe Schlüsseltechnologien, 2010.

[41] A. Crépieux and C. Lacroix, Dzyaloshinsky-Moriya interactions induced by symmetry breaking at a surface, Journal of Magnetism and Magnetic Materials **182**(3), 341 – 349 (1998).

[42] I. E. Dzyaloshinskii, Theory of helicoidal structures in antiferromagnets. III., Sov. Phys. JETP **20**(665) (1965).

[43] Y. A. Izyumov, Modulated, or long-periodic, magnetic structures of crystals, Sov. Phys. Usp. **27**(845) (1984).

[44] N. Bernstein, M. J. Mehl, and D. A. Papaconstantopoulos, Nonorthogonal tight-binding model for germanium, Phys. Rev. B **66**(7), 075212 (Aug 2002).

[45] T. Tanaka, H. Kontani, M. Naito, T. Naito, D. S. Hirashima, K. Yamada, and J. Inoue, Intrinsic spin Hall effect and orbital Hall effect in $4d$ and $5d$ transition metals, Phys. Rev. B **77**(16), 165117 (Apr 2008).

[46] S. Sanvito and N. A. Hill, Ground state of half-metallic zinc-blende MnAs, Phys. Rev. B **62**(23), 15553–15560 (Dec 2000).

[47] O. Gunnarsson, Band model for magnetism of transition metals in the spin-density-functional formalism, Journal of Physics F: Metal Physics **6**(4), 587 (1976).

[48] J. F. Janak, Uniform susceptibilities of metallic elements, Phys. Rev. B **16**(1), 255–262 (Jul 1977).

[49] D. D. Johnson, Modified Broyden's method for accelerating convergence in self-consistent calculations, Phys. Rev. B **38**(18), 12807–12813 (Dec 1988).

[50] `http://www.xcrysden.org/doc/fermi.html`.

[51] J. F. Cornwell, *Group Theory in Physics, Volume 2*, Academic Press, 1984.

[52] `http://www.netlib.org/lapack/`.

[53] `http://olymp.phys.chemie.uni-muenchen.de/`.

[54] D. S. G. Bauer, Atomistic Spin-Dynamics in Confined Magnetic Nano-Structures, Master's thesis, RWTH Aachen, 2008.

[55] D. S. G. Bauer, P. Mavropoulos, S. Lounis, and S. Blügel, Thermally activated magnetization reversal in monoatomic magnetic chains on surfaces studied by classical atomistic spin-dynamics simulations, (2010).

[56] G. Lehmann and M. Taut, On the Numerical Calculation of the Density of States and Related Properties, Physica Status Solidi (b) **54**(2), 469 – 477 (1972).

[57] N. C. Bacalis, D. A. Papaconstantopoulos, M. J. Mehl, and M. Lach-hab, Transferable tight-binding parameters for ferromagnetic and paramagnetic iron, Physica B: Condensed Matter **296**(1-3), 125 – 128 (2001).