FLEUR user guide

Gregor Michalicek, Gustav Bihlmayer, Anoop Chandran,
Christoph Friedrich, Matthias Redies, Stefan Rost,
Uliana Alekseeva, Alexander Neukirchen, Henning Janssen,
Vasily Tseplyaev, Robin Hilgers and Daniel Wortmann

April 7, 2022

Supported by:

MAX DRIVING THE EXASCALE TRANSITION

www.flapw.de
fleur

This is an automatically generated pdf version of the documentation available at the FLEUR homepage: https://www.flapw.de.

Contents

1 Overview

The all-electron full-potential linearized augmented-plane-wave (FLAPW) method is a highly precise realization of density functional theory (DFT). It is often considered as a gold standard to measure the precision of other approaches. As the name suggests it treats all electrons in the framework of DFT and does not rely on the pseudopotential approximation. There are also no shape approximations to the potential involved, i.e., it is a full-potential method. It allows to describe electrons with adapted degrees of relativity based on the position relative to the atomic nuclei and the electron energy. Furthermore the LAPW basis used to represent the valence electrons is systematically extendable such that convergence with respect to the representation can be achieved and verified.

The Fleur code implements the FLAPW method. It allows the calculation of properties obtainable by DFT for crystals and thin films composed of arbitrary chemical elements. Besides standard properties such as the band structures, the density of states, typical for most DFT codes Fleur can also calculate a multitude of properties relevant for magnetic materials and phenomena. Complex non-collinear magnetic systems and systems with strong spin-orbit coupling can be accurately described. Different levels of parallelism (both MPI and OpenMP) as well as optimizations and ports for different computing hardware like CPUs or GPUs enable the deployment of Fleur on a wide range of computing resources up to the largest supercomputers.

The precision of the FLAPW method comes at the cost of complex parametrizations of the calculations. To simplify this Fleur employs a two-step calculation procedure. In the first step basic structural input is processed by the Fleur input generator . This program then creates a completely parametrized Fleur input file with material-adapted default parameters. In the second step the user typically modifies selected parameters in the Fleur input file and finally starts the calculation. This procedure implies that the user does not have to write the complex Fleur input file on his own, but an understanding of the file is required.

This user guide aims at teaching the interested user an understanding of the Fleur input file together with the general Fleur usage. For this the theroretical background chapter of the user guide presents the underlying theory as far as it is required for the understanding of the input file and connects it to the actual parameters in the input. The guide then continues in the next chapter by discussing the procedures needed to perform a range of specific calculations. In the chapter about expert knowledge and troubleshooting general

knowledge about the FLAPW method and the Fleur code is discussed. This is relevant for all calculations.

To get in touch with the program early, the beginning of the user guide discusses a quick start. This implies instructions for the installation of Fleur, as well as a technical overview on how to use the program.

When performing actual calculations the knowledge from the first chapters of the user guide should be combined with the short descriptions of the input files in the reference. This is also the part of the user guide that is most important for experienced users. Note that the reference section also contains a list of journal articles that may be of interest to the user.

Throughout the user guide different notations are used to clearly express different aspects of the Fleur usage. In particular this applies to the description of the Fleur input file. This is an XML file. To address certain elements or attributes of this file the user guide makes use of the XPath notation. An XML element will be addressed as `/path/to/element` and an attribute as `/path/to/element/@theAttribute`. If it is clear from the context the paths may be shortened. For example larger XML sections will be highlighted in separate code blocks like

```
<enclosingXMLElement someBooleanAttribute="T"
                     someFloatingPointAttribute="0.75*Pi">
   <xmlElementWithoutSubelements someIntegerAttribute="10"
                                 someCharacterStringAttribute="Hello" />
   <!-- some comment -->
</enclosingXMLElement>
```

Note that the XML in these blocks may feature a slightly different format in comparison to its appearance in the Fleur input file. For readability within the user guide some lines are broken up into several lines and some numbers are reduced to less digits. Also other in- and output may be slightly reformatted in the user guide.

Beyond the XML highlighting text sections with special importance will also be emphasized in separate blocks. Especially text that directly addresses the input file will be highlighted like

> `/comment` element.

The reader of this guide should also be aware that Fleur is in rapid development and the user guide may lag behind. This implies that certain capabilities of the code are not yet covered in the user guide and the usage of others may slightly differ. If you are interested in such a feature please contact the developers directly.

2 Quick start

Contents

Interested users may want to find out whether Fleur can be installed and run on their hardware before reading the whole user guide. For this purpose these steps are covered in this chapter before the program and its underlying theory are discussed in detail in the following chapters.

2.1 Installation of Fleur

We are aware of the fact that installing Fleur can be a tricky task on many machines. While we tried to make the process as userfriendly as possible, there are still a couple of challenges you might encounter.

note System requirements
The installation of Fleur depends onthe availablity of some software on your system. You can find this list of requirements below. If in doubt, please check with your system administrator to see if all requirements for using Fleur can be fulfilled on your system.

question Further support
For help register at the MailingListand post your questions there.

If you manage to compile on some system that can be of general interest, please consider to report to fleur@fz-juelich.de with your settings so that these can be included in the general distribution.

2.1.1 Quick guide

If you are extremely lucky (and/or your system is directly supported by us) installation can be very simple:

1. Run the configuration script `'PATH_TO_SOURCE_CODE/configure.sh`. You can do that in any directory in which the 'build' directory should be created. The script accepts some useful arguments, you can run the script with `configure.sh -h` to get a list of supported arguments.
2. The script creates the build directory and runs cmake. If all goes well (look at the output) you can then change to the build directory and run `cd build; make`.
3. If `make` does not report any error you are done!

Please be aware that there are different executables that could be be build:

- **`inpgen`**: The input generator used to construct the full input file for Fleur
- **`fleur`**: A serial version (i.e. no MPI distributed memory parallelism, multithreading might still be used)
- **`fleur_MPI`**: A parallel version of Fleur able to run on multiple nodes using MPI.

Usually only the serial or the MPI version will be build. You can run the MPI-version in serial while it is of course not possible to use the non-MPI version with MPI.

You might want to run the automatic tests.

2.1.2 Requirements

There are a couple of external dependencies in the build process of Fleur.

Required are:

- **cmake**: The build process uses cmake to configure FLEUR. You should have at least version 3.0. Some options might require newer versions. Cmake is available for free at www.cmake.org.
- **Compilers**: You will need a Fortran compiler and a corresponding C-compiler (i.e. the two have to be able to work together via the iso-c bindings of Fortran). Please check our list of compilers to see if your compiler should work.

- **BLAS/LAPACK**: These standard linear algebra libraries are required. You should try your best not to use a reference implementation from Netlib but look for an optimized version for your system. In general compiler and/or hardware vendors provide optimized libraries such as the MKL (Intel) or ESSL (IBM). If you do not have access to those, check openBLAS.
- **libxml2**: This is a standard XML-library that is available on most systems. If it is missing on your computer you should really complain with your admin. *Please note that you might need a development package of this library as well.* To compile this library yourself, see xmlsoft.org.

Optional:

Fleur can benefit significantly if the following further software components are available. Please be aware that some of these can be difficult to use for FLEUR and see the Instructions for adjusting your configuration for details on how to provide input into the build process to use these libraries.

- **MPI**: Probably most important is the possibility to compile a version of Fleur running on multiple nodes using MPI. If you have a proper MPI installation on your system this should be straightforward to use.
- **HDF5**: Fleur can use the HDF5 library for input/output. This is useful in two situations. On the one hand you might want to use HDF5 for reading/writing your charge density files to avoid having a machine-dependent format that can prevent portability. Also the HDF5 IO gives you some more features here. On the other hand you have to use parallel-HDF5 if you do IO of the eigenvectors in an MPI parallel calculation. This is needed if you can not store the data in memory or want to postprocess the eigenvectors. Please be aware that you need the Fortran-90 interface of the HDF5 and that the HDF5 library should be compiled with the same compiler you use to compile Fleur!
- **SCALAPACK/ELPA**: If you use MPI and want to solve a single eigenvalue problem with more than a single MPI-Task, you need a library with a distributed memory eigensolver. Here you can use the SCALAPACK or the [[http://elpa.mpcdf.mpg.de/|ELPA]] library. Please note that the ELPA library changed its API several times, hence you might see problems in compiling with it.
- **MAGMA**: Fleur can also use the MAGMA library to run on GPUs. If you intend to use this feature, please get in contact with us.

You should also check the output of `configure.sh -h` for further dependencies and hints.

2.1.3 Configure

The `configure.sh` script found in the main Fleur source directory can (and should) be used to start the configuration of Fleur. It is called as `configure.sh [-l LABEL ]`

`[-d] [CONFIG]`.

The most used options are:

- **-l LABEL**: This specifies a label for the build. This is used to custimize the build-directory to build.LABEL and can be used to facilitate different builds from the same source.
- **-d**: This specifies a debugging build.
- **CONFIG**: This is a string to choose one of the preconfigured configurations. It can be useful if you find one which matches your setup.

Note Check out!
More options are available. Please check theoutput of `configure.sh -h`. You might find options to include dependencies into the build usefull.

The `configure.sh` script performs the following steps:

1. It creates a subdirectory called 'build' or 'build.LABEL'. If this directory is already present, the old directory will be overwritten.
2. It copies the CONFIG dependent configuration into this directory (this is actually done in the script 'cmake/machines.sh'). The special choice of "AUTO" for CONFIG will not provide any further configuration but relies completely on cmake. You can specify a config.cmake file in the working directory (from which you call configure.sh) to modify this AUTO mode. 3 Finally cmake is run in the build directory to determine your configuration.

If you specify -d as argument of configure.sh, the string "debug" will be added to LABEL and a debugging version of FLEUR will be build, i.e., the corresponding compiler switches will be set.

2.1.3.1 How to adjust the Configuration

While `cmake` and the `configure.sh` script can determine the correct compilation switches automatically in some cases (mostly those known to us), in many other instances fine-tuning is needed. In particular you might have to:

- provide hints on which compiler to use
- provide hints on how to use libraries.

Setting the compiler to use

By defining the environment variables `FC` and `CC` to point to the Fortran and C compiler you can make sure that cmake uses the correct compilers, e.g., you might want to say `export FC=mpif90`.

Please be aware that the use of CONFIG specific settings might overwrite the environment variable.

Adding flags for the compiler

This should be done using the `-flags` option to `configure.sh`. So for example you might want to say `configure.sh -flags "-r8 -nowarn"`.

In general for a compiler not known in cmake/compilerflags.cmake you need at least an option to specify the promotion of standard real variables to double precision (like `-r8`). But additional switches can/should be used.

Adding include directories

For libraries with a Fortran-90 interface, ELPA, HDF5, MAGMA, ... you probably will have to provide an include path. This can be achieved using the `-includedir` option. So you might want to say something like `configure.sh -includedir SOMEPATH`.

Adding linker flags

To add flags to the linker you can

- add a directory in which the linker looks for libraries with `-libdir SOMEDIR`
- add the corresponding link option(s) with, e.g., `-link "-lxml2;-llapack;-lblas"`. Please note that the specification is different from the compiler switches as different switches are separated by ';'.

Building with LibXC

If the user decides to compile Fleur together with LibXC support, LibXC needs to be of version `>=5.0.0`. The easiest way to install Fleur with LibXC is to download Fleur with git:

```
git clone --branch release https://iffgit.fz-juelich.de/fleur/fleur.git
```

Then you can configure Fleur with

```
./configure.sh -libxc TRUE
```

which will download the correct libxc version, compile and link it. ***Notice this only works if you download Fleur using git.***

2.1.3.2 Compiler specifics

Fleur is known to work with the following compilers:

INTEL:

The Intel Fortran compilers (ifort) is able to compile Fleur. Depending on the version you might experience the following problems:

- Versions < 16.0 will most probably not work correctly
- Version 19.0 has issues with the debugging version of Fleur.

GFortran:

GFortran is known to work with versions newer than 6.3.

PGI:

The PGI compilers also can compile FLEUR. Here you need at least version 18.4 but might still run into some problems.

2.1.4 Tests

After the build was finished you might want to run the automatic test.

Just type `ctest` in the build directory for this purpose.

Please note: * The tests run on the computer you just compiled on. Hence a cross-compiled executable will not run. * You can use the environment variables `juDFT_MPI` to specify any additional command needed to start fleur_MPI, e.g., say `export juDFT_MPI="mpirun -n 2"` to run with two MPI tasks. * You can use the environment variable `juDFT` to pass command line arguments to Fleur, e.g., say `export juDFT='-mem'`. * To run a specific test only (or a range of tests) use the `-I` option of ctest (check `ctest -h` for details) * The tests are run in the subdirectory Testing/work. You can check this directory to see why a specific test fails.

2.1.5 Experimental features

The Fleur code is under rapid development. This implies that in each release there are features that are considered to be experimental. These features are either not well tested or their implementation is incomplete. The user will not find explicit documentation on these features in the Fleur user guide, but may become aware of them due to information from other sources. Especially the output of the `configure.sh` script may indicate certain features because it lists for several software libraries whether they have been detected and will be linked to Fleur.

In the MaX 4.0 release of Fleur expecially the functionalities related to the `MAGMA` and `Wannier90` libraries are considered to be experimental.

2.2 Running Fleur

This section adresses the question of how to run Fleur "by hand". If one is interested in running Fleur in a scripting environment the AiiDA plug-in should be considered.

There are several executables created in the build process:

- **inpgen**: The input generator used to construct the full Fleur input file using only basic structural input about the unit cell for `inpgen` itself. The generated Fleur input file features unit-cell-adapted default parameters.

and one of:

- **fleur**: A serial version (i.e. no MPI distributed memory parallelism, multithreading can still be used)
- **fleur_MPI**: A parallel version of Fleur able to run on multiple nodes using MPI. It can also be started in a serial way without the respective MPI command.

In most cases you will first run the input generator to create an inp.xml .

Afterwards you will run `fleur` or `fleur_MPI` using this inp.xml file. Please note that the Fleur executable will always read its input from an inp.xml file in the current directory.

2.2.1 Command line options

The run-time behaviour of Fleur can be modified using command line switches. These switches modify the way Fleur might operate or in some cases determine what Fleur actually does. If you want to change the calculation setup you should modify the inp.xml file.

In the following the most relevant command line options are listed. For a full list of available options, please run `fleur -h`

General options:

- `-h`: Prints a help listing all command-line options.
- `-check`: Runs only the init-part of Fleur, useful to check if setup is correct.
- `-debugtime`: Prints out all starting/stopping of timers. Can be useful to monitor the progress of the run.

Options controlling the IO of eigenvectors/values: (not all are available if you did not compile with the required libraries)

- `-eig mem`: no IO, all eigenvectors are stored in memory. This can be a problem if you have little memory and many **k** points. Default for serial version of Fleur.
- `-eig da`: write data to disk using Fortran direct-access files. Fastest disk IO scheme. *Only available in serial version of FLEUR.*
- `-eig mpi`: no IO, all eigenvectors are stored in memory in a distributed fashion. Uses MPI one-sided communication. Default for MPI version of Fleur. *Only available in MPI version of FLEUR.*
- `-eig hdf`: write data to disk using HDF5 library. Can be used in serial and MPI version (if HDF5 is compiled for MPI-IO).

Options controlling the Diagonalization: (not all are available if you did not compile with the required libraries)

- `-diag lapack`: Use standard LAPACK routines. Default in Fleur (if not parallel eigenvector parallelization)
- `-diag scalapack`: Use SCALAPACK for parallel eigenvector parallelization.
- `-diag elpa`: Use ELPA for parallel eigenvector parallelization.
- Further options might be available, check `fleur -h` for a list.

2.2.2 Environment Variables

There are basically two environments variables you might want to change when using Fleur.

- **OMP_NUM_THREADS**: As Fleur uses OpenMP it is generally a good idea to consider adjusting OMP_NUM_THREADS to use all cores available. While this might happen automatically in you queuing system you should check if you use appropriate values. Check the output of Fleur to standard out. One might want to use `export OMP_NUM_THREADS=2` or something similar.
- **juDFT**: You can use the juDFT variable to set command line switches that do not require an additional argument. For example `export juDFT="-debugtime"` would run FLEUR with this command line switch.

2.2.3 Starting Fleur with MPI parallelization

While the OpenMP parallelization is controlled with the environment variable mentioned above or other external settings the degree of MPI parallelization is controlled in the command line call to the MPI executable of Fleur. Typically `fleur_MPI` is provided as a command-line argument to the respective MPI executable `mpirun`, `mpiexec`, `srun`, or similar. The degree of MPI parallelization is also a command-line argument to the respective MPI executable. Please be aware that invoking such an executable with the non-MPI-parallelized version of Fleur will yield Fleur errors. The distribution of the Fleur calculation onto different compute nodes of a cluster is typically set by the respective variables in a Jobfile for the used queueing system.

For a guide on how to choose good parallelization schemes please have a look at the respective section .

3 Theoretical background

Contents

Within the theory part of the user documentation the parts of the underlying theory relevant to the user are sketched. This implies the theoretical aspects connected to numerical Fleur input parameters and theory parts employing approximations. For other parts of the theory the user is directed to read the original publications.

3.1 Partitioning of the unit cell and energy ranges

The setup of unit cells typically starts with the specification of the shape of the unit cell. This is done by defining a Bravais matrix containing the lattice vectors. For many unit cells the entries in the Bravais matrix are not independent of each other because the shape of the cell features certain symmetries. According to these symmetries Bravais matrices can be classified into certain crystal families and lattice systems, each specified by a reduced set of parameters.

`cell/bulkLattice` section for bulk systems and in `cell/filmLattice` for thin film systems.

The FLAPW method is based on a partitioning of the unit cell into different regions. As sketched in figure 1 it is divided into non-overlapping so-called muffin-tin (MT) spheres centered at each atomic nucleus α and an interstitial region (IR) between the spheres.

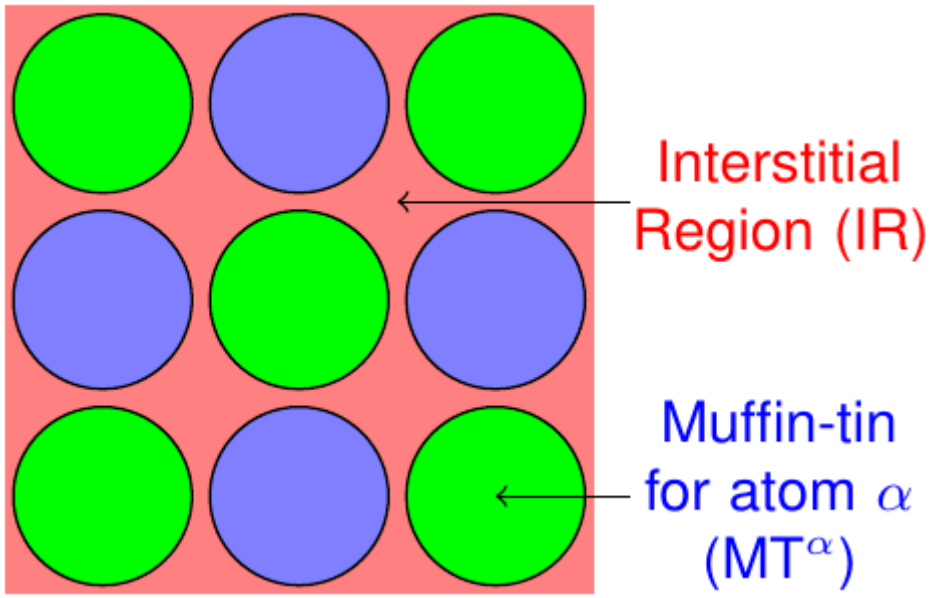

Figure 3.1: Partitioning of the unit cell in non-overlapping MT spheres and an interstitial region.

The representation of the Kohn-Sham eigenstates as well as the density and the potential varies between these regions and can be adjusted by cutoff and other numerical parameters for each region.

unit cell one or more species (atoms with same numerical parameters) are defined in `atomSpecies`. For each species many parameters are defined. This includes an identifier (`atomSpecies/species/@name`) and the atomic number (`atomSpecies/species/@atomicNumber`). The name of the chemical element is also shown (not an input parameter) (`atomSpecies/species/@element`). The respecive MT radius is set in `atomSpecies/species/mtSphere/@radius`.

symmetrical equivalence in the unit cell (`atomGroups`). Each `atomGroups/atomGroup` is connected to a species via the respective identifier (`atomGroups/atomGroup/@species`). The atom positions related to each atom of an atom group are specified in different ways within `atomGroups/atomGroup`. `atomGroups/atomGroup/relPos` specifies a position in units of lattice vectors. In `atomGroups/atomGroup/filmPos` an atom position is specified by lattice vectors in the xy plane and an absolute coordinate in z direction.**

Another subdivision is performed with respect to the energy levels of the electrons in the unit cell. Figure 2 sketches the different classes of electron states obtained by this procedure.

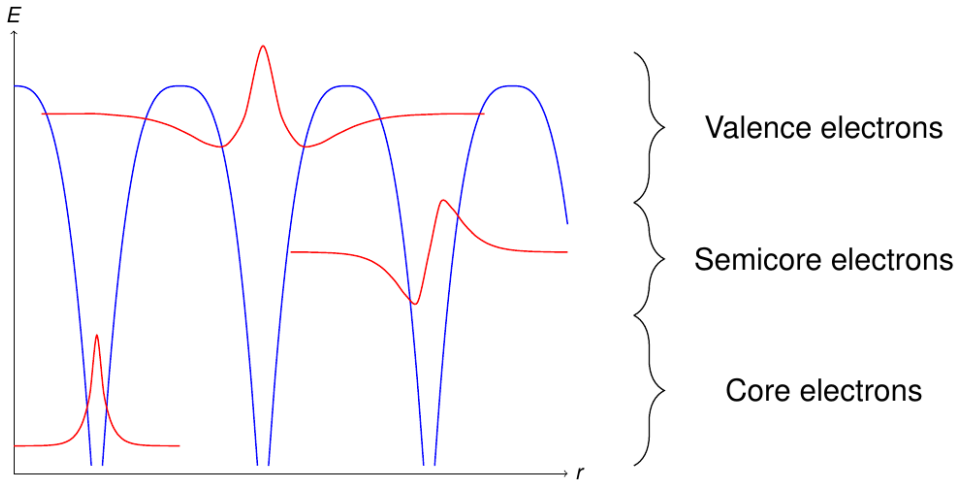

Figure 3.2: Classification of the electrons according to their energy levels. The blue curve is the potential.

Energetically deep lying electrons are denoted as core electrons. These electrons are localized within the MT sphere of the respective atom and should be described fully-relativistically.

On the other hand electrons near the Fermi energy are considered to be valence electrons. They are typically delocalized and in crystals with periodicity $\mathbf{R}$ have to be described as Bloch states

$$\Psi_{\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{kr}} \cdot u_{\mathbf{k}}(\mathbf{r}) \tag{3.1}$$

with

$$u_{\mathbf{k}}(\mathbf{r}) = u_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) \tag{3.2}$$

Energetically between the valence and the core electrons there may be electrons which can neither be considered to be fully localized within the respective MT sphere, nor delocalized.

In FLAPW the concrete classification of the electrons has to be performed by the user as an input to the calculation. This is typically supported by reasonable default classifications. However, these default classifications may not work for every unit cell since the size of MT spheres depends on bond lengths and may vary strongly for different chemical compositions.

3.2 Treatment of valence electrons

3.2.1 The LAPW basis

The valence electrons are represented by the LAPW basis. It consists of the functions:

$$\phi_{\mathbf{kG}}(\mathbf{r}) = \begin{cases} \frac{1}{\sqrt{\Omega}} e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}} & \text{for } \mathbf{r} \in \text{IR} \\ \sum\limits_{L} \left[a^{L\alpha}_{\mathbf{kG}} u^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l}) + b^{L\alpha}_{\mathbf{kG}} \dot{u}^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l}) \right] Y_L(\hat{\mathbf{r}}_\alpha) & \text{for } \mathbf{r} \in \text{MT}^{\alpha} \end{cases} \tag{3.3}$$

A sketch of an LAPW basis function is shown in figure 1.

For each sampled **k**-point of the irreducible wedge of the Brillouin zone each LAPW basis function is indexed by the reciprocal lattice vector **G** entering the equation in the plane wave in the interstitial region. A cutoff parameter $K_{\max} = |\mathbf{k} + \mathbf{G}|_{\max}$ defines which plane waves are considered in the LAPW basis.

> $K_{\max}$ is specified in units of a_0^{-1} (`calculationSetup/cutoffs/@Kmax`).

The MT part of the basis consists of a linear combination of the radial functions $u^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l})$ and $\dot{u}^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l})$ times spherical harmonics $Y_L(\hat{\mathbf{r}}_\alpha)$. Here $\mathbf{r}_\alpha = \mathbf{r} - \tau_\alpha$ is the position relative to atom α at τ_α and $L = (l, m)$ is a composite index for the angular momentum quantum number l and the magnetic quantum number m. For predefined energy parameters E^{α}_{l} the radial functions are solutions to the radial Schrödinger equation (actually the scalar-relativistic approximation to the Dirac equation) $(u^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l}))$ and their energy derivatives

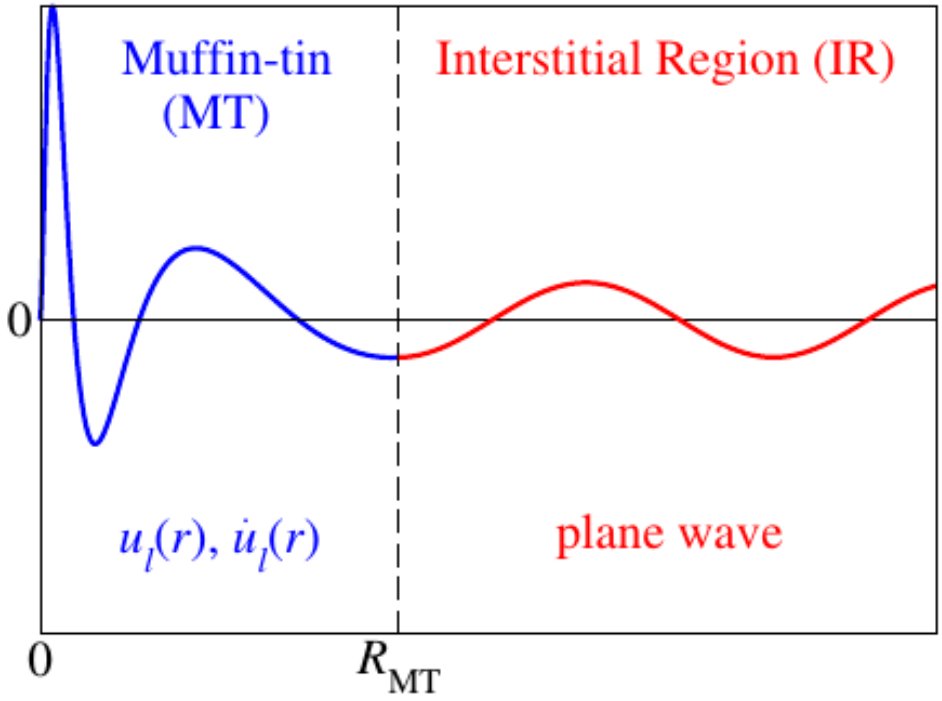

Figure 3.3: Sketch of an LAPW basis function depending on the distance to an atomic nucleus.

$(\dot{u}_l^\alpha(r_\alpha, E_l^\alpha))$. The matching coefficients $a_{\mathbf{kG}}^{L\alpha}$ and $b_{\mathbf{kG}}^{L\alpha}$ are automatically calculated by matching the MT part of the LAPW function at the MT boundary in value and slope to the plane wave in the interstitial region. The L summation is limited by the cutoff parameters $l_{\max}^\alpha$.

$l_{\max}^\alpha$ can be set for each atom species seperately (`atomSpecies/species/atomicCutoffs/@lmax`). As a rule of thumb reasonable values can be obtained from the relation $l_{\max}^\alpha \approx K_{\max} \cdot R_{\mathrm{MT}}^\alpha$.

The energy parameters E_l^α are obtained from an automatic procedure within each iteration of the self-consistency loop. By default the Fleur code uses *atomic energy parameters*, meaning that the energy parameters are obtained as eigenenergies of an artificial atomic problem specified by the radial part of the effective potential within the MT sphere together with an artificial confining continuation of this potential beyond the sphere boundary. For each l quantum number up to the f shell a main quantum number specifies which eigenenergy becomes the respective energy parameter.

determination of the energy parameters are specified for each l-channel up to $l = 3$ (`atomSpecies/species/energyParameters`). For higher l the energy parameter (not the main quantum number) is set to that of the $l = 3$ channel.

The linearization in the LAPW basis allows the accurate representation of Kohn-Sham states within a range of a few eV around the energy parameters. Furthermore one can show that the LAPW basis is orthogonal to core electron states that are completely confined within the MT sphere. This allows to calculate core electron states separately but also within the self-consistent DFT approach.

3.2.2 Local orbitals

Semicore states considerably reach out of the MT spheres but are also rather far away from the valence energy parameters. The user of an FLAPW code has the option to treat such states either as core electrons or as valence electrons. Though, treating them as core electrons can lead to the appearence of ghost bands which may mess up the calculation if they are within the energy range of the occupied bands. If semicore electrons have to be treated as valence electrons the LAPW basis can be extended by local orbitals (LOs) to represent these states.

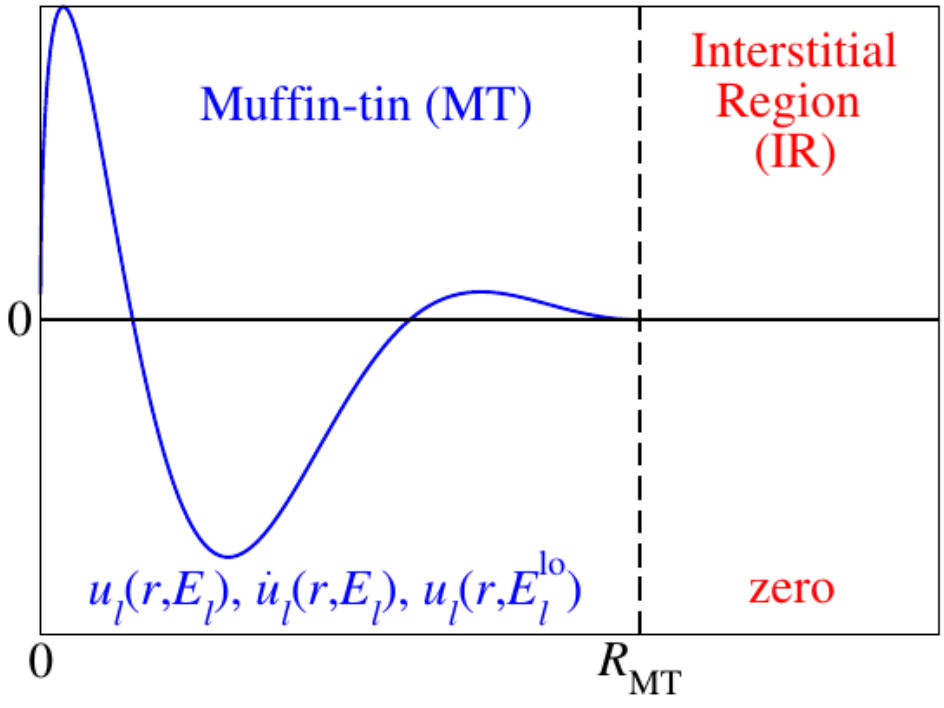

Figure 3.4: Sketch of a local orbital (LO) basis function depending on the distance to an atomic nucleus.

As sketched in figure 2 local orbitals are additional basis functions for a defined l channel (one function for each m) that are confined within a MT sphere. They are constructed as:

$$\phi_L^{\text{lo}}(\mathbf{r}) = \left[a_L^{\text{lo}} u_l^\alpha(r_\alpha, E_l^\alpha) + b_L^{\text{lo}} \dot{u}_l^\alpha(r_\alpha, E_l^\alpha) + c_L^{\text{lo}} u_{\text{LO},l}^\alpha(r_\alpha, E_l^{\text{lo}})\right] Y_L(\hat{\mathbf{r}}_\alpha) \tag{3.4}$$

They consist of the already known radial functions $u_l^\alpha(r_\alpha, E_l^\alpha)$ and $\dot{u}_l^\alpha(r_\alpha, E_l^\alpha)$ and another radial function $u_{\text{LO},l}^\alpha(r_\alpha, E_l^{\text{lo}})$. The coefficients a_L^{lo}, b_L^{lo}, and c_L^{lo} are determined by enforcing zero value and slope at the MT sphere boundaries, as well as a normalization of the LO over the MT sphere.

The third radial function $u_{\text{LO},l}^\alpha(r_\alpha, E_l^{\text{lo}})$ can be constructed from different recipies. For the representation of semicore states one typically sets $u_{\text{LO},l}^\alpha(r_\alpha, E_l^{\text{lo}}) = u_l^\alpha(r_\alpha, E_l^{\text{lo}})$, i.e., one uses a solution to the spherical potential at an energy parameter that is near the semicore state.

LOs can also be used to represent unoccupied states, e.g., by using an energy parameter within the range of these states, or to generally eliminate the linearization error for the valence electrons by setting $u_{\text{LO},l}^\alpha(r_\alpha, E_l^{\text{lo}}) = \ddot{u}_l^\alpha(r_\alpha, E_l^\alpha)$. The double dot in this expression denotes the second energy derivative.

> atom species directly below the specification of the energy parameters in XML elements `atomSpecies/species/lo`. See the input file documentation for details on the specification of the third radial function in this XML element.

3.3 The LAPW basis for thin film systems

Calculations on thin films can be performed in different ways. A popular approach to this task is to define a bulk unit cell in which the film is provided along the xy-plane and the width of the film in z direction only covers a small fraction of the unit cell's length in that direction. Such a setup defines periodic repetitions of the film in z direction and one has to make sure that the vacuum in between the films is large enough to decouple them.

On the basis of the FLAPW method a more elegant treatment of thin film systems is possible. As sketched in figure 1 one can set up unit cells with semi-infinite vacuum regions above and below the film.

An LAPW basis function for such a setup is defined as

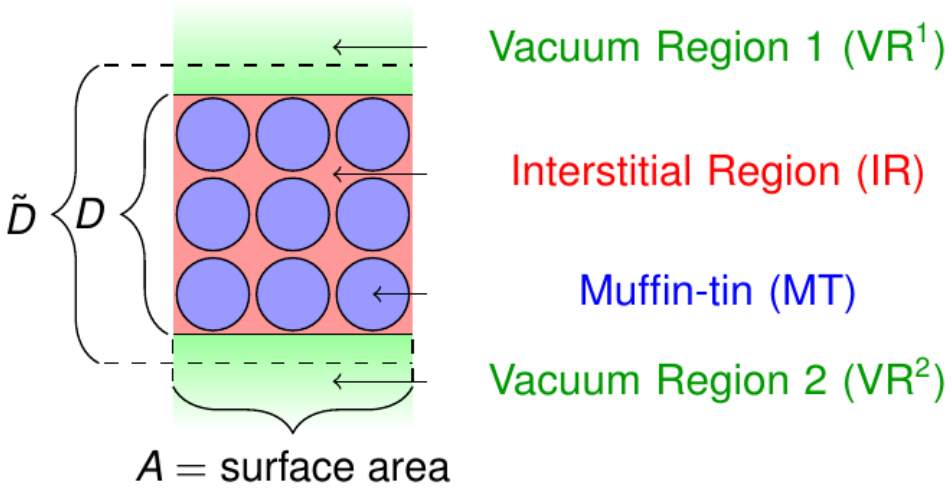

Figure 3.5: The unit cell partitioning for thin films.

$$\phi_{\mathbf{k}_\parallel \mathbf{G}}(\mathbf{r}) = \begin{cases} \frac{1}{\sqrt{\Omega}} e^{i(\mathbf{k}_\parallel + \mathbf{G})\mathbf{r}} & \text{for } \mathbf{r} \in \text{IR} \\ \sum\limits_{L} \left[a^{L\alpha}_{\mathbf{k}_\parallel \mathbf{G}} u^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l}) + b^{L\alpha}_{\mathbf{k}_\parallel \mathbf{G}} \dot{u}^{\alpha}_{l}(r_\alpha, E^{\alpha}_{l}) \right] Y_L(\hat{\mathbf{r}}_\alpha) & \text{for } \mathbf{r} \in \text{MT}^{\alpha} \\ \left[a^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}} u^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}_\parallel}(z, E^{\text{vac}}) + b^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}} \dot{u}^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}_\parallel}(z, E^{\text{vac}}) \right] \times \frac{1}{\sqrt{A}} e^{i(\mathbf{k}_\parallel + \mathbf{G}_\parallel)\mathbf{r}_\parallel} & \text{for } \mathbf{r} \in \text{VR}^{\text{vac}} \end{cases} \tag{3.5}$$

where the extension in the vacuum regions consists of the two functions $u^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}_\parallel}(z, E^{\text{vac}})$ and $\dot{u}^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}_\parallel}(z, E^{\text{vac}})$ are solutions and energy derivatives to the Schrödinger equation in the respective vacuum region

$$\left[-\frac{1}{2}\frac{\partial^2}{\partial z^2} + \frac{1}{2}\left(\mathbf{k}_\parallel + \mathbf{G}_\parallel\right)^2 + V^{\text{avg,vac}}_{\text{eff}}(z) - E^{\text{vac}} \right] u^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}_\parallel}(z, E^{\text{vac}}) = 0 \tag{3.6}$$

at the energy parameter E^{vac}. The coefficients $a^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}}$ and $b^{\text{vac}}_{\mathbf{k}_\parallel \mathbf{G}}$ are determined by enforcing continuity of value and slope of the basis function at the vacuum boundary defined by the parameter D. The z components of the plane waves in the interstitial region are constructed such that they feature a periodicty in agreement to the parameter $\tilde{D}$. Note that $\tilde{D}$ is larger than D to avoid a kink of the implicitely periodic interstitial representation of the wave functions at the interstitial-vacuum boundaries.

specified in `cell/filmLattice/@dVac` and `cell/filmLattice/@dTilda`. For each vacuum the energy parameters are set in the respective `cell/filmLattice/vacuumEnergyParameters` element for the spin-up (`cell/filmLattice/vacuumEnergyParameters/@spinUp`) and spin-down (`cell/filmLattice/vacuumEnergyParameters/@spinDown`) channel separately, relative to the vacuum potential at an infinite distance from the film. In the case of a nonmagnetic calculation the value for the spin-up electrons is used for the calculation of the energy parameter. If both vacua are equivalent only a single `cell/filmLattice/vacuumEnergyParameters` element for vacuum 1 is present. It is then also used for vacuum 2.

3.4 Details on the Hamiltonian and Overlap matrix setup

Due to the partitioning of the unit cell into different regions it is natural to express the Hamiltonian and Overlap matrices in terms of sums of contributions originating from each of these regions. Also the construction of the radial functions in the MT spheres on the basis of the spherical part of the effective potential leads to a seperataion of spherical and nonspherical contributions in the spheres. In consequence the Hamiltonian is expressed as

$$H^{\mathbf{k}}_{\mathbf{G},\mathbf{G}'} = H^{\mathbf{k},\mathrm{IR}}_{\mathbf{G},\mathbf{G}'} + \left[\sum_{\alpha} H^{\mathbf{k},\alpha,\mathrm{sphr}}_{\mathbf{G},\mathbf{G}'} + H^{\mathbf{k},\alpha,\mathrm{nsphr}}_{\mathbf{G},\mathbf{G}'}\right] + \left[\sum_{\mathrm{vac}} H^{\mathbf{k},\mathrm{vac}}_{\mathbf{G},\mathbf{G}'}\right] \tag{3.7}$$

and the Overlap matrix as

$$S^{\mathbf{k}}_{\mathbf{G},\mathbf{G}'} = S^{\mathbf{k},\mathrm{IR}}_{\mathbf{G},\mathbf{G}'} + \left[\sum_{\alpha} S^{\mathbf{k},\alpha}_{\mathbf{G},\mathbf{G}'}\right] + \left[\sum_{\mathrm{vac}} S^{\mathbf{k},\mathrm{vac}}_{\mathbf{G},\mathbf{G}'}\right]. \tag{3.8}$$

In the following the basic setup of these matrices is sketched. Since this is mainly intended to introduce cutoff parameters and other quantities that are important for the usage of the FLAPW method some contributions to the matrices will be neglected. For example the contributions related to local orbitals are not discussed here. These additional basis functions would increase the matrix size. Also the vacuum contributions will be ignored.

3.4.1 Interstitial contributions

For the construction of the interstitial contributions to the matrices one defines the step function

$$\Theta(\mathbf{r}) = \begin{array}{ll} 1 & \text{for } \mathbf{r} \in \text{IR} \\ 0 & \text{for } \mathbf{r} \in \text{MT}^\alpha \end{array} . \tag{3.9}$$

It is analytically constructed in reciprocal space as

$$\Theta(\mathbf{G}) = \delta_{\mathbf{G},\mathbf{0}} - \sum_\alpha e^{-i\mathbf{G}\tau_\alpha} \frac{4\pi (R_{\text{MT}^\alpha})^3}{\Omega} \frac{j_1(GR_{\text{MT}^\alpha})}{GR_{\text{MT}^\alpha}} \tag{3.10}$$

up to the reciprocal cutoff parameter $G_{\max}$. j_1 is the spherical Bessel function for $l = 1$ and Ω is the unit cell volume. The step function is the indicator function for the interstitial region but due to the finite reciprocal cutoff parameter its real space representation obtained by a Fourier transformation is only approximately correct. Fortunately only the correctness in reciprocal space up to the cutoff parameter is needed and this is fulfilled.

With the definition of the step function the interstitial contributions to the matrices become

$$\begin{aligned} H^{\mathbf{k},\text{IR}}_{\mathbf{GG'}} &= \left\langle \phi_{\mathbf{kG}} \middle| \hat{H} \middle| \phi_{\mathbf{kG'}} \right\rangle_{\text{IR}} \\ &= \left\langle \tilde{\phi}_{\mathbf{kG}} \middle| V(\mathbf{r})\Theta(\mathbf{r}) - \frac{1}{2}\Theta(\mathbf{r})\nabla^2 \middle| \tilde{\phi}_{\mathbf{kG'}} \right\rangle_\Omega \\ &= \frac{1}{\Omega} \int_\Omega e^{-i(\mathbf{G}-\mathbf{G'})\mathbf{r}} V(\mathbf{r})\Theta(\mathbf{r}) d^3r + \frac{(\mathbf{k}+\mathbf{G'})^2}{2\Omega} \int_\Omega e^{-i(\mathbf{G}-\mathbf{G'})\mathbf{r}} \Theta(\mathbf{r}) d^3r \qquad (3.11) \\ &= (V\Theta)(\mathbf{G}-\mathbf{G'}) + \frac{(\mathbf{k}+\mathbf{G'})^2}{2}\Theta(\mathbf{G}-\mathbf{G'}) \qquad (3.12) \end{aligned}$$

and

$$\begin{aligned} S^{\mathbf{k},\text{IR}}_{\mathbf{GG'}} &= \left\langle \tilde{\phi}_{\mathbf{kG}} \middle| \Theta(\mathbf{r}) \middle| \tilde{\phi}_{\mathbf{kG'}} \right\rangle_\Omega = \frac{1}{\Omega} \int_\Omega e^{-i(\mathbf{G}-\mathbf{G'})\mathbf{r}} \Theta(\mathbf{r}) d^3r \\ &= \Theta(\mathbf{G}-\mathbf{G'}) \end{aligned} \tag{3.13}$$

The reciprocal plane wave cutoff for the step function $G_{\max}$ is also used for the function $(V\Theta)(\mathbf{G})$.

set in `calculationSetup/cutoffs/@Gmax`. Besides $\Theta(\mathbf{G})$ and $(V\Theta)(\mathbf{G})$ it is also the cutoff for the density and the potential in the interstitial region. It has to be at least twice as large as $K_{\max}$ and also at least as large as the reciprocal cutoff for the exchange-correlation potential $G^{\text{xc}}_{\max}$.

Note that the expression for the interstitial contributions to the Hamiltonian does not lead to a Hermitian matrix. Since this property is required the actual expression to calculate this contribution is altered to guarantee it. It becomes

$$H^{\mathbf{k},\text{IR}}_{\mathbf{GG}'} = (V\Theta)(\mathbf{G}-\mathbf{G}') + \frac{(\mathbf{k}+\mathbf{G})^2 + (\mathbf{k}+\mathbf{G}')^2}{4}\Theta(\mathbf{G}-\mathbf{G}'). \tag{3.14}$$

3.4.2 Spherical MT contributions

Since the radial functions are already constructed for the spherical potential in each MT sphere the respective contributions to the Hamiltonian matrix can easily be expressed as

$$H^{\mathbf{k},\alpha,\text{sphr}}_{\mathbf{GG}'} = \sum_{l=0}^{l^\alpha_{\max}} \sum_{m=-l}^{l} E_l \left[\left(a^{L\alpha}_{\mathbf{kG}}\right)^* a^{L\alpha}_{\mathbf{kG}'} + \langle \dot{u}^\alpha_l | \dot{u}^\alpha_l \rangle \left(b^{L\alpha}_{\mathbf{kG}}\right)^* b^{L\alpha}_{\mathbf{kG}'} \right] + \left(a^{L\alpha}_{\mathbf{kG}}\right)^* b^{L\alpha}_{\mathbf{kG}'}. \tag{3.15}$$

Accordingly the MT contribution to the Overlap matrix becomes

$$S^{\mathbf{k},\alpha}_{\mathbf{GG}'} = \sum_{l=0}^{l^\alpha_{\max}} \sum_{m=-l}^{l} \left(a^{L\alpha}_{\mathbf{kG}}\right)^* a^{L\alpha}_{\mathbf{kG}'} + \langle \dot{u}^\alpha_l | \dot{u}^\alpha_l \rangle \left(b^{L\alpha}_{\mathbf{kG}}\right)^* b^{L\alpha}_{\mathbf{kG}'}. \tag{3.16}$$

Note that the expression for the Hamiltonian contributions again does not lead to a Hermitian matrix. Therefore it is altered to

$$H^{\mathbf{k},\alpha,\text{sphr}}_{\mathbf{GG}'} = \sum_{l=0}^{l^\alpha_{\max}} \sum_{m=-l}^{l} E_l \left[\left(a^{L\alpha}_{\mathbf{kG}}\right)^* a^{L\alpha}_{\mathbf{kG}'} + \langle \dot{u}^\alpha_l | \dot{u}^\alpha_l \rangle \left(b^{L\alpha}_{\mathbf{kG}}\right)^* b^{L\alpha}_{\mathbf{kG}'} \right] + \frac{1}{2}\left(\left(a^{L\alpha}_{\mathbf{kG}}\right)^* b^{L\alpha}_{\mathbf{kG}'} + \left(b^{L\alpha}_{\mathbf{kG}}\right)^* a^{L\alpha}_{\mathbf{kG}'} \right). \tag{3.17}$$

The summation over the magnetic quantum numbers m can actually be performed analytically. This makes the calculation of the spherical MT contributions efficient.

3.4.3 Nonspherical contributions in the MT spheres

For the calculation of the contribution due to the nonspherical part of the potential in a MT sphere one first defines a local Hamiltonian $t_{LL'}^{\alpha[\circ][\diamond]}$ in the basis of the radial functions $u_l^\alpha(r_\alpha, E_l^\alpha)$ and $\dot{u}_l^\alpha(r_\alpha, E_l^\alpha)$ as

$$\begin{aligned} t_{LL'}^{\alpha[\circ][\diamond]} &= \int_{\mathrm{MT}^\alpha} \sum_{L''=1}^{l_{\max}^\alpha} {}^{[\circ]}u_l^\alpha(r_\alpha, E_l^\alpha) V_{L''}^\alpha(r_\alpha) {}^{[\diamond]}u_{l'}^\alpha(r_\alpha, E_{l'}^\alpha) Y_L^*(\hat{\mathbf{r}}_\alpha) Y_{L''}(\hat{\mathbf{r}}_\alpha) Y_{L'}(\hat{\mathbf{r}}_\alpha) d^3r \\ &= \int_0^{R_{\mathrm{MT}^\alpha}} \sum_{L''=1}^{l_{\max}^\alpha} r_\alpha^2 {}^{[\circ]}u_l^\alpha(r_\alpha, E_l^\alpha) V_{L''}^\alpha(r_\alpha) {}^{[\diamond]}u_{l'}^\alpha(r_\alpha, E_{l'}^\alpha) G_{ll''l'}^{mm''m'} dr_\alpha. \end{aligned} \tag{3.18}$$

In this expression $\circ$ and $\diamond$ can each stand for a $u_l^\alpha(r_\alpha, E_l^\alpha)$ or a $\dot{u}_l^\alpha(r_\alpha, E_l^\alpha)$ and $V_{L''}^\alpha(r_\alpha)$ is the potential in the sphere. The Gaunt coefficients $G_{ll''l'}^{mm''m'}$ are defined as

$$G_{ll''l'}^{mm''m'} = \int_0^{2\pi} \int_0^{\pi} Y_L^*(\theta,\phi) Y_{L''}(\theta,\phi) Y_{L'}(\theta,\phi) \sin(\theta) d\theta d\varphi. \tag{3.19}$$

The local Hamiltonian then has to be incorporated into the global Hamiltonian for the whole unit cell by applying the matching coefficients. One obtains

$$\begin{aligned} H_{\mathbf{GG'}}^{\mathbf{k},\alpha,\mathrm{nsphr}} &= \sum_L \sum_{L'} \left(a_{\mathbf{kG}}^{L\alpha}\right)^* t_{LL'}^{\alpha[\,][\,]} a_{\mathbf{kG'}}^{L'\alpha} + \left(a_{\mathbf{kG}}^{L\alpha}\right)^* t_{LL'}^{\alpha[\,][\cdot]} b_{\mathbf{kG'}}^{L'\alpha} \\ &\quad + \left(b_{\mathbf{kG}}^{L\alpha}\right)^* t_{LL'}^{\alpha[\cdot][\,]} a_{\mathbf{kG'}}^{L'\alpha} + \left(b_{\mathbf{kG}}^{L\alpha}\right)^* t_{LL'}^{\alpha[\cdot][\cdot]} b_{\mathbf{kG'}}^{L'\alpha} \\ &= \sum_L \left(a_{\mathbf{kG}}^{L\alpha}\right)^* \left(\sum_{L'} t_{LL'}^{\alpha[\,][\,]} a_{\mathbf{kG'}}^{L'\alpha}\right) + \left(a_{\mathbf{kG}}^{L\alpha}\right)^* \left(\sum_{L'} t_{LL'}^{\alpha[\,][\cdot]} b_{\mathbf{kG'}}^{L'\alpha}\right) \\ &\quad + \left(b_{\mathbf{kG}}^{L\alpha}\right)^* \left(\sum_{L'} t_{LL'}^{\alpha[\cdot][\,]} a_{\mathbf{kG'}}^{L'\alpha}\right) + \left(b_{\mathbf{kG}}^{L\alpha}\right)^* \left(\sum_{L'} t_{LL'}^{\alpha[\cdot][\cdot]} b_{\mathbf{kG'}}^{L'\alpha}\right). \end{aligned}$$

In these expressions the sums over L and L' are actually limited by the reduced l cutoff parameter $l_{\mathrm{nonsph}}^\alpha$. While in the calculation for the spherical contributions very high l cutoff parameters were required to obtain reasonable kinetic energies and Overlap matrices, the sums involved for the nonspherical contributions are related to the potential only. In practice this implies that smaller cutoffs can be used here. Note that the reduction of the l cutoffs reduces the computational demands significantly.

> $l_{\mathrm{nonsph}}^\alpha$ are set for each atom species separately in `atomSpecies/species/atomicCutoffs/@lnonsph`.

3.5 Treatment of core electrons

The FLAPW method allows to incorporate all electrons into the DFT calculations. While the valence electrons are represented by the LAPW basis the core electron wave functions are calculated on a radial grid within the MT spheres and an extension of this grid beyond the sphere boundaries. As an approximation for the calculation of the core electron states Fleur only takes the spherical part of the effective potential $V^{\alpha}_{l=0}(r_\alpha)$ into account. Beyond the MT sphere boundary this part of the potential is extended by a cone-like confining potential which has the same value at the MT boundary as $V^{\alpha}_{l=0}(r_\alpha)$.

Due to the singularity of the effective potential at the atomic nucleus together with the concentration of the core electron states near the nucleus core electrons feature a very high kinetic energy. Therefore the determination of the core electron wave functions and eigenenergies are treated by a fully-relativistic solver on the basis of the Kohn-Sham-Dirac equations

$$\left[c\alpha \cdot \mathbf{p} + (\beta - 1)m_\mathrm{e}c^2 + V_{l=0}(r)\right]\psi_\nu(r) = E_\nu\psi_\nu(r),$$

where ψ_ν the 4-component wavefunction, E_ν the ν-th eigenenergy, c is the speed of light, and $\mathbf{p}$ the momentum operator. α is the vector of the 4×4 matrices

$$\alpha_i = \begin{pmatrix} 0 & \sigma_i \\ \sigma_i & 0 \end{pmatrix}$$

in which σ_i are the Pauli spin matrices. Finally, β is the 4×4 matrix

$$\beta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

and m_e is the rest mass of the electron. In the here-used Hartree atomic units this is actually 1. It is shown in the equation explicitely to make the user aware of the substraction of the rest mass energy. This is done to establish a consistent treatment between the core electrons and the valence electrons. In the determination of the energy eigenvalues for the valence electrons the electron rest mass was also not considered.

There are several aspects of the treatment of the core electrons that can be controlled by the user. First the user can decide whether the tail of the core electron density reaching out of the MT sphere is accurately added to the total density or the charge within this tail is just evenly distributed over the interstitial region. While the latter option is less accurate it may significantly speed up the calculation because the re-expansion of the core-tail density in the other MT spheres is computationally expensive. For details on the re-expansion the user may consult the respective section on the density .

`calculationSetup/coreElectrons/@ctail` switch controlls how the core-tail density is treated. If it is set to `T` it will be correctly added to the total density, otherwise only the overall charge is corrected by adjusting the interstitial charge. If it is set to `T` then `calculationSetup/coreElectrons/@coretail_lmax` specifies the maximal l quantum number for the re-expansion of the core tail density in the other MT spheres.

The second aspect that can be controlled by the user is the choice of whether the core density is actually updated in every iteration of the SCF calculation or a frozen core approximation is used and a previously calculated core electron density is held fixed.

controlled by the `calculationSetup/coreElectrons/@frcor` switch. If this is set to `T` the frozen core approximation will be used. Note that this only works if at least a single SCF iteration was already performed in which a core electron density was calculated. By default the switch is set to `F` and it is assumed that the user wants to obtain a self-consistent density also for the core electrons.

The last aspect to be controlled by the user is related to spin-polarized potentials. Spin is not a good quantum number in fully-relativistic calculations. This implies that the correct handling of spin-polarized potentials is more complex. It is possible to use an exact treatment of spin-polarized systems with the approach described in *H. Ebert, Fully relativistic tretment of core states for a spin-dependent potential, J. Phys: Condens. Matter 1, 9111 (1989).* But as an approximation Fleur also implements a relativistic solver that treats each spin channel separately. If only the core electron density is required this already provides highly accurate results. If the core level spectrum is of interest, the exact solver should be used.

potentials is made by setting the `calculationSetup/coreElectrons/@kcrel` parameter. If this is set to `0` the approximative method with the separate treatment of the two spin-channels is used. If it is set to `1` the exact solver is used. Note that some optional parts of Fleur and some programs in the environment of Fleur are not compatible to the exact solver.

3.6 Usage of symmetries

To speed up and guarantee the stability of calculations Fleur makes extensive use of symmetries. Symmetry operations which leave the unit cell unchanged are automatically detected by the Fleur input generator. The different atoms are then grouped into different types of symmetry-equivalent atoms.

different atoms is placed in two different sections. In the `atomSpecies` section general parameters symmetry-independent parameters for the different species are set. In the `atomGroups` section the different symmetry-equivalent atoms are grouped together in `atomGroups/atomGroup` elements. The atom positions are specified there. Each `atomGroup` is connected to a certain species via the entry in `atomGroups/atomGroup/@species` adressing the species with the same entry in `atomSpecies/species/@name`. The terms `atom group` and `atom type` are synonymous for each other.

provided in the Fleur input file within the `cell/symmetryOperations` element.

Of course, the grouping of the atoms has to be consistent with the specified symmetry operations. Since it is cumbersome to ensure this consistency manually the preferred way to deal with symmetries is to invoke the input generator such that the symmetries do not have to be changed by the user afterwards.

For certain calculations the user should ensure in the input generator input that no symmetries are found that are inconsistent to the planned calculation. For this the user can discriminate between different atoms of the same chemical element to ensure that these atoms do not end up in the same atom group. For calculations involving spin-orbit coupling the user can define the spin-quantization axis in the input generator input. This axis can also break symmtries.

The usage of symmetries allows to subdivide the Brillouin zone into many irreducible wedges that are each equivalent to each other. The specified k point sets therefore only have to sample the irreducible wedge of the Brillouin zone (IBZ) and an integration over the whole Brillouin zone can be performed by assigning adequate weights to the k points in the IBZ and summing over them. The construction of the IBZ and the assignment of the weights to the k points is automatically performed by Fleur or the input generator.

`calculationSetup/bzIntegration/kPointList` elements, each specifying the k point set. For more information please consult the appropriate section in the reference section and the documentation of the input generator for a guide to generate k point sets.

3.7 Construction of the charge density

The partitioning of the unit cell into different regions does not only affect the LAPW basis but also the representation of the charge density which also becomes region-dependent. In detail the charge density is given by

$$\rho(\mathbf{r}) = \begin{cases} \sum\limits_{\mathbf{G}} \rho_{\mathbf{G}} e^{i\mathbf{G}\mathbf{r}} & \text{for } \mathbf{r} \in \text{IR} \\ \sum\limits_{L} \rho_L^{\alpha}(r_\alpha) Y_L(\hat{\mathbf{r}}_\alpha) & \text{for } \mathbf{r} \in \text{MT}^{\alpha} \\ \sum\limits_{\mathbf{G}_\parallel} \rho_{\mathbf{G}_\parallel}^{\text{vac}}(z) e^{i\mathbf{G}_\parallel \mathbf{r}_\parallel} & \text{for } \mathbf{r} \in \text{VR}^{\text{vac}} \end{cases} . \tag{3.20}$$

In the interstitial region it is given as a plane-wave expansion, in the MT spheres as a spherical harmonics expansion, and in the vacuum region as one-dimensional functions times plane waves parallel to the respective thin film.

region and the vacuum regions is limited by the $G_{\max}$ cutoff specified in `calculationSetup/cutoffs/@Gmax`. The spherical harmonics expansion in the MT spheres is limited by the $l_{\max}^{\alpha}$ cutoffs specified for each species in `atomSpecies/species/atomicCutoffs/@lmax`.

The charge density is constructed by calculating

$$\rho(\mathbf{r}) = 2 \sum_{\mathbf{k}} \sum_{\nu} \omega_{\mathbf{k}}^{\nu} \left| \psi_{\mathbf{k}}^{\nu}(\mathbf{r}) \right|^2 , \tag{3.21}$$

where $\psi_{\mathbf{k}}^{\nu}(\mathbf{r})$ is the Kohn-Sham wave function for eigenvalue ν at k point $\mathbf{k}$ and $\omega_{\mathbf{k}}^{\nu}$ is the occupation of the respective state times the weight of the k point. The factor 2 is due to the spin-degeneracy in non-spinpolarized systems. For magnetic calculations it is replaced by a sum over the spins.

Only parts of the charge density construction are discussed below. For the valence electrons the determination of the occupation numbers is discussed. It is important to users since it depends on input parameters. In the construction of the core electron density contribution several minor approximations come into play. These are also discussed below.

3.7.1 Occupation numbers for the valence electrons

The occupation numbers $\omega_{\mathbf{k}}^{\nu}$ for the valence electrons are calculated by occupying the states according to a Fermi distribution. In a first step this approach is used together with the predefined number of valence electrons in the unit cell N_{elec} to determine the Fermi energy E_{F} by enforcing

$$N_{\text{elec}} = 2 \sum_{\mathbf{k}} \sum_{\nu} \omega(\mathbf{k}) \frac{1}{e^{(\epsilon_{\mathbf{k}}^{\nu} - E_{\text{F}})/k_{\text{B}}T} + 1} , \tag{3.22}$$

where the factor 2 is again due to spin-degeneracy, $\omega(\mathbf{k})$ is the weight of the respective k point, $\epsilon_{\mathbf{k}}^{\nu}$ is the eigenenergy of the respective state, and T is the temperature used for the definition of the Fermi distribution.

distribution is typically specified in terms of the related energy in `calculationSetup/bzIntegration/@fermiSmearingEnergy`. The number of valence electrons is specified in `calculationSetup/bzIntegration/@valenceElectrons`.

After determining the Fermi energy the occupation numbers are calculated as

$$\omega_{\mathbf{k}}^{\nu} = \omega(\mathbf{k}) \frac{1}{e^{(\epsilon_{\mathbf{k}}^{\nu} - E_{\mathrm{F}})/k_{\mathrm{B}}T} + 1}. \tag{3.23}$$

3.7.2 Density contributions from the core electrons

Besides the density due to the valence electrons there are also density contributions due to the core electrons. It is constructed by occupying the core electron states and summing the densities related to each state up for each of the atoms.

separately in `atomSpecies/species/electronConfig`. By default all core electron states are fully occupied. Of course, for the calculation of core holes these occupations can be changed by explicitely providing an electron configuration in the optional element `atomSpecies/species/electronConfig`.

One should be aware of some details of the core electron density construction. Approximately the core electrons are confined within the MT spheres, but for a rigorous treatment one also has to consider the tails of the core electron states reaching out of the MT spheres.

if the `calculationSetup/coreElectrons/@ctail` switch is set to `T`. Otherwise the charge stored in the tails of the core electron states is just evenly distributed over the interstitial region.

In an exact treatment the core electron density can be represented similarly to the overall charge density as

$$\rho_{\mathrm{c}}(\mathbf{r}) = \begin{cases} \sum\limits_{\mathbf{G}} \rho_{\mathrm{c}\mathbf{G}} e^{i\mathbf{G}\mathbf{r}} & \text{for } \mathbf{r} \in \mathrm{IR} \\ \sum\limits_{L} \rho_{\mathrm{c}L}^{\alpha}(r_\alpha) Y_L(\hat{\mathbf{r}}_\alpha) & \text{for } \mathbf{r} \in \mathrm{MT}^{\alpha} \\ \sum\limits_{\mathbf{G}_\parallel} \rho_{\mathrm{c}\mathbf{G}_\parallel}^{\mathrm{vac}}(z) e^{i\mathbf{G}_\parallel \mathbf{r}_\parallel} & \text{for } \mathbf{r} \in \mathrm{VR}^{\mathrm{vac}} \end{cases}. \tag{3.24}$$

It is constructed by first considering the (spherical) internal core electron density originating from a certain atom $\rho_{\mathrm{c}}^{\alpha,\,\mathrm{int}}(r_\alpha)$. From this one constructs the respective interstitial contribution by replacing the MT part of $\rho_{\mathrm{c}}^{\alpha,\,\mathrm{int}}(r_\alpha)$ by a smooth pseudodensity and then performing a plane-wave expansion of the resulting function. In detail one defines the pseudodensity as

$$\tilde{\rho}_{\mathrm{c}}^{\alpha}(r_\alpha) = \begin{cases} a_\alpha e^{-b_\alpha r_\alpha^2} & \text{for } r_\alpha < R_{\mathrm{MT}^\alpha} \\ \rho_{\mathrm{c}}^{\alpha,\,\mathrm{int}}(r_\alpha) & \text{for } r_\alpha \geq R_{\mathrm{MT}^\alpha} \end{cases}, \tag{3.25}$$

where a_α and b_α are determined by enforcing a continuous value and slope of the pseudo charge density. The respective plane-wave expansion coefficients are then calculated as

$$\rho_{\mathrm{c}\mathbf{G}}^{\alpha} = \sqrt{\frac{4\pi}{\Omega}} e^{i\mathbf{G}\tau_\alpha} \int_0^\infty \tilde{\rho}_{\mathrm{c}}^{\alpha}(r_\alpha) j_0(Gr_\alpha) r_\alpha^2 dr_\alpha \tag{3.26}$$

and the overall core density interstitial plane-wave expansion coefficients become

$$\rho_{\mathrm{c}\mathbf{G}} = \sum_\alpha \rho_{\mathrm{c}\mathbf{G}}^{\alpha}. \tag{3.27}$$

Of course, the interstitial plane-wave expansion stretches over the whole unit cell and core electron tails originating from a certain atom may reach into the MT sphere of another atom. These contributions also have to be added. To do so one re-expands the interstitial core electron density in terms of spherical harmonics in each MT sphere. This defines the external core electron density

$$\rho_{\mathrm{c}L}^{\alpha,\,\mathrm{ext}}(r_\alpha) = \frac{4\pi}{\sqrt{\Omega}} \sum_{\mathbf{G}} \rho_{\mathrm{c}\mathbf{G}} i^l e^{i\mathbf{G}\tau_\alpha} j_l(Gr_\alpha) Y_L^*(\hat{\mathbf{G}}). \tag{3.28}$$

The overall core electron density in the MT sphere of atom α then becomes

$$\sum_L \rho_{\mathrm{c}L}^{\alpha}(r_\alpha) Y_L(\hat{\mathbf{r}}_\alpha) = \left[\rho_{\mathrm{c}}^{\alpha,\,\mathrm{int}}(r_\alpha) - a_\alpha e^{-b_\alpha r_\alpha^2}\right] Y_{l=0,m=0}(\hat{\mathbf{r}}_\alpha) + \sum_L \rho_{\mathrm{c}L}^{\alpha,\,\mathrm{ext}}(r_\alpha) Y_L(\hat{\mathbf{r}}_\alpha). \tag{3.29}$$

Note that the re-expansion of the interstitial core electron density in terms of spherical harmonics is computationally expensive and it is questionable whether a large computational effort should be accepted since only a very small charge is affected by this calculation. Therefore one limits the L for this expansion by a new cutoff parameter $l_{\mathrm{max}}^{\mathrm{coretail}}$.

cutoff parameter is specified in the optional XML attribute `calculationSetup/coreElectrons/@coretail_lmax`. If for a certain atom species this value is larger than the respective $l^{\alpha}_{\max}$ cutoff it is reduced to this cutoff for this atom species. If the $l^{\text{coretail}}_{\max}$ parameter is not set in the input file its default value is 99.

For the vacuum regions one assumes that the core electron density in these regions originates from a narrow energy range which lies far below the vacuum potential at infinity. Under this assumption it is resonable to approximate the density in the vacuum by an exponentially decaying function. In detail one constructs the vacuum density expansion coefficients as

$$\rho^{\text{vac}}_{\text{c}\mathbf{G}_{\parallel}}(z) = \rho^{\text{vac}}_{\text{c}\mathbf{G}_{\parallel}}(z^{\text{vac}}) e^{-\frac{z^{\text{vac}}}{|z^{\text{vac}}|}\alpha^{\text{vac}}_{\mathbf{G}_{\parallel}}(z-z^{\text{vac}})}. \tag{3.30}$$

In this equation z^{vac} is the z coordinate of the vacuum boundary. The coefficients $\rho^{\text{vac}}_{\text{c}\mathbf{G}_{\parallel}}(z^{\text{vac}})$ and $\alpha^{\text{vac}}_{\mathbf{G}_{\parallel}}$ are again determined by enforcing continuity of value and slope of the density.

3.8 Construction of the potential

Just like the charge density the effective potential is also represented in a region-dependent way as

$$V_{\text{eff}}(\mathbf{r}) = \begin{cases} \sum\limits_{\mathbf{G}} V^{\text{eff}}_{\mathbf{G}} e^{i\mathbf{G}\mathbf{r}} & \text{for } \mathbf{r} \in \text{IR} \\ \sum\limits_{L} V^{\text{eff},\alpha}_{L}(r_{\alpha}) Y_L(\hat{\mathbf{r}}_{\alpha}) & \text{for } \mathbf{r} \in \text{MT}^{\alpha} \\ \sum\limits_{\mathbf{G}_{\parallel}} V^{\text{eff,vac}}_{\mathbf{G}_{\parallel}}(z) e^{i\mathbf{G}_{\parallel}\mathbf{r}_{\parallel}} & \text{for } \mathbf{r} \in \text{VR}^{\text{vac}} \end{cases}. \tag{3.31}$$

regions the plane-wave expansion is limited by the $G_{\max}$ cutoff specified in `calculationSetup/cutoffs/@Gmax`. The spherical harmonics expansion in the MT spheres is limited by the $l^{\alpha}_{\max}$ cutoffs specified for each species in `atomSpecies/species/atomicCutoffs/@lmax`

As a sum of the Hartree potential, the external potential, and the exchange-correlation potential it is given by

$$V_{\text{eff}}(\mathbf{r}) = V_{\text{H}}(\mathbf{r}) + V_{\text{ext}}(\mathbf{r}) + V_{\text{xc}}(\mathbf{r}) \tag{3.32}$$

The construction of the Hartree potential together with the external potential involves neither approximations nor special input parameters. It is therefore neglected in this discussion. For details on this part of the calculation see *M. Weinert, Solution of Poisson's equation: Beyond Ewaldtype methods, JMP 22, 2433 (1981).*

3.8.1 The exchange-correlation potential

There are some user-specified parameters that control the construction of the XC part of the potential.

First the construction of the XC potential in the interstitial region has its own reduced reciprocal cutoff parameter `calculationSetup/cutoffs/@GmaxXC`. It has to comply with the constraint

$$G_{\max} \geq G_{\max}^{\mathrm{XC}} \geq 2 \cdot K_{\max}. \tag{3.33}$$

Next the type of XC functional is specified in `xcFunctional/@name`. See the reference on the Fleur input file for a list of available options.

Finally the user can choose to enable relativistic corrections according to *A.H. MacDonnald and S.H. Vosko, A relativistic density functional formalism, JPhysC: Solid State Physics 12, 2977 (1979)* with the logical switch `xcFunctional/@relativisticCorrections`.

3.9 Treatment of collinear magnetism

For the investigation of magnetic materials objects like the LAPW basis functions, the Hamiltonian and overlap matrices, the charge density, and the effective potential have to become spin-dependent. With the spin index $\sigma \in -1/2, 1/2$ the spin-dependent quantities thus become

$$\phi_{\mathbf{kG}}(\mathbf{r}) \rightarrow \phi^{\sigma}_{\mathbf{kG}}(\mathbf{r}), \qquad H^{\mathbf{k}}_{\mathbf{G},\mathbf{G}'} \rightarrow H^{\mathbf{k},\sigma}_{\mathbf{G},\mathbf{G}'}, \qquad S^{\mathbf{k}}_{\mathbf{G},\mathbf{G}'} \rightarrow S^{\mathbf{k},\sigma}_{\mathbf{G},\mathbf{G}'}, \tag{3.34}$$

$$\rho(\mathbf{r}) \rightarrow \rho^{\sigma}(\mathbf{r}), \qquad \text{and} \qquad V_{\mathrm{eff}}(\mathbf{r}) \rightarrow V^{\sigma}_{\mathrm{eff}}(\mathbf{r}). \tag{3.35}$$

For the potential, however, the coupling of the two spins only affects the XC potential and an optional Zeeman term. The effective potential becomes

$$V^{\sigma}_{\mathrm{eff}}(\mathbf{r}) = V_{\mathrm{H}}(\mathbf{r}) + V_{\mathrm{ext}}(\mathbf{r}) + V^{\sigma}_{\mathrm{xc}}(\mathbf{r}) + \mu_{\mathrm{B}} g_{\mathrm{e}} \sigma B, \tag{3.36}$$

where μ_B is the Bohr magneton, g_e the g-factor of the electron, and B an optionally applied magnetic field. Calculations with such external fields are discussed in a separate section .

Starting with a spin-polarized density Fleur determines the spin-dependent potential. Due to this spin-dependence also the radial functions of the LAPW basis in the MT spheres become spin-dependent. The Hamiltonian and Overlap matrices are subsequently also spin-dependent. The point where the algorithm connects the two spins again is the calculation of the Fermi energy and the total charge density.

explicitly specified in the Fleur input file. The parameter `calculationSetup/magnetism/@jspins` has to be set to 2. For nonmagnetic calculations it is 1. The input generator sets it to 2 by default whenever there is an atom in the unit cell that is considered to be magnetic, i.e., Cr, Mn, Fe, Co, Ni as well as all atoms with partially filled f shells. If a spin-dependent Fleur calculation for materials without such chemical elements has to be started the user has to specify this manually.

To obtain a magnetic solution of an SCF calculation the starting density already has to break a symmetric treatment of the spins. This is done by specifying for each atom initial magnetic moments that are already a prototype for the magnetic configuration to be investigated. Of course, this initial magnetic configuration may break some symmetries being present in the unit cell otherwise.

unwanted symmetries atoms of the same chemical element which are supposed to feature differing initial magnetic moments have to be distinguished. For this they are specified with different fractional atomic numbers, i.e. `26.01` and `26.02` for two different Fe species. Only the integer part of the number specifies the chemical element, the fractional part is only used to associate the atoms with different atom species that are automatically generated in the Fleur input file. In this file for each atom species the magnetic moment is specified by using appropriate occupations in the `atomSpecies/species/electronConfig` section. Before starting the SCF calculation these parameters have to be adapted to the specific needs of the calculation. For example, for the investigation of antiferromagnetic Cr a unit cell with two atoms has to be set up, where each atom has its own Cr species. For one of the species the default magnetic moment in the Fleur input file then has to be negated manually.

When performing calculations on magnetic materials magnetism-related properties are of interest. The most important of these quantities are the magnetic moments in the MT spheres and in the whole unit cell.

magnetic moments in the MT spheres are written out in the `magneticMomentsInMTSpheres` section. For each atom type an entry `magneticMomentsInMTSpheres/magneticMoment` exists which includes beyond the magnetic moment in the respective sphere the amount of charge for each spin. For the calculation of the magnetic moment for the whole unit cell Fleur writes out the two sections `valenceDensity` and `allElectronCharges`. The former one covers the valence electrons only while the latter one also considers the core electrons. In each of these there are `spinDependentCharge` entries that write out the amount of charge for each spin in the whole unit cell as well as in the different regions of the unit cell. The magnetic moment is obtained by substracting the numbers for the two spins from each other.

3.10 Spin-orbit coupling in 2nd variation

We usually use the scalar-relativistic FLAPW method to obtain the solutions of the KS equations in a "first variation". When spin-orbit coupling (SOC) has to be included, we do not consider SOC in the interstitial and vacuum regions, assume that only spherically-symmetric MT-sphere potential components contribute to SOC and neglect the small components of the wavefunctions. Therefore, only the large components of Φ_i contribute through the SOC matrix elements of the Hamiltonian

$$\hat{\mathbf{H}}\Phi_i \;=\; \epsilon_i^0\;\Phi_i + \begin{pmatrix} \frac{1}{\zeta^2}\sigma\cdot(\nabla V_{-\sigma}(\mathbf{r})\times\mathbf{p}) & 0 \\ 0 & (\epsilon_i^0-1)\mathrm{I}_2 \end{pmatrix}\Phi_i \tag{3.37}$$

where we neglect the spin-orbit contribution to the overlap of Φ_i due to SOC.

When spin-orbit interaction gives only a small contribution to the Hamiltonian, in a collinear calculation one can make use of that by recognizing that the eigenfunctions of the scalar-relativistic Hamiltonian are much more efficient basisfunctions for the variational treatment of the relativistic Hamiltonian than the original LAPW basis functions. Thus after determining the eigenvectors $\Phi^0_{\mathbf{k}\nu'\sigma'}(\mathbf{r})$ of the scalar-relativistic Hamiltonian we expand the wavefunction $\Phi_{\mathbf{k}\nu}(\mathbf{r})$ as

$$\Phi_{\mathbf{k}\nu}(\mathbf{r}) = \sum_{\nu'\sigma'}^{2N_2} C^{\nu}_{\mathbf{k}\nu'\sigma'}\Phi^0_{\mathbf{k}\nu'\sigma'}(\mathbf{r}), \tag{3.38}$$

where the expansion coefficients $C^{\nu}_{\mathbf{k}\nu'\sigma'}$ are assumed to be already multiplied with the Pauli spinor belonging to σ'.

This constitutes a second variational eigenproblem where N_2 is the number of basisfunctions per spin with typically $N_2 \approx (1.3 - 1.5)N_1$ where N_1 is the number of occupied states.

convergence parameter. In the Fleur input file it is specified in `calculationSetup/cutoffs/@numbands`. If the parameter is set to 0 a material-dependent default value is assigned. The same is done if the assigned value is lower than the default. The actually used parameter is written out to the out.xml file in `numericalParameters/bands/@numbands`.

The expansion coefficients are determined by the second eigenproblem as

$$\sum_{\nu'',\sigma''}^{2N_2} \langle \mathbf{k}\nu'\sigma' \mid \hat{H} \mid \mathbf{k}\nu''\sigma'' \rangle C^{\nu}_{\mathbf{k}\nu''\sigma''} = \epsilon_{\mathbf{k}\nu} C^{\nu}_{\mathbf{k}\nu'\sigma'} \,. \tag{3.39}$$

(In a non-magnetic system the eigenvectors of the first variation will simply be doubled for the above expansion.)

`calculationSetup` section by setting the `l_soc="T"` switch in the Fleur input file, e.g.:

```
<soc theta=".00" phi=".00" l_soc="T" spav="F" />
```

The angles `theta` and `phi` are the polar and azimuthal angles (in units of 2π) defining the direction of the spin-quantization axis (SQA), i.e., normally the magnetization direction. With the `spav` feature you can trigger the usage of a spin-averaged potential instead of $V_{-\sigma}$ in the Hamiltonian above.

Note, that SOC usually lowers the symmetry of the crystal, e.g., cubic bcc Fe will become tetragonal if the magnetization points in one of the cubic axes. Therefore, already at the level of the input-file generator one can define `&soc 0.0 0.0 /` where the two numbers are again the angles ϑ and φ. If this is not done an activation of the SOC may conflict with the available symmetry operations determined by the input generator.

Taking SOC into account will not only add the spin-orbit contribution to the total energy (which allows calculating magnetic anisotropies) but also gives access to the orbital magnetic moments (projected onto the SQA). Note, that the SOC treatment in second variation leads to a "doubling of eigenvalues" since each state (with components in both spin-sectors now) will show up in both spin channels.

It is possible to restrict the relativistic treatment to only selected atoms in the unit cell by activating the `socscale` feature in the `special` tag of the `atomSpecies` section. E.g. setting `socscale="0.0"` switches off SOC in the selected atom species.

Apart from self-consistent SOC calculations, it is also possible to apply the magnetic force theorem to calculate the magnetocrystalline anisotopy. The corresponding documentation can be found here .

Note, that strong SOC will generally give rise to non-collinearity in the system so that this collinear + second variation scheme is not necessarily always a good choice. Then a non-collinear calculation and treatment of SOC in first variation is preferable.

3.11 Treatment of noncollinear magnetism

In the case of noncollinear magnetism, the Kohn-Sham equation obtains the form

$$-\frac{\hbar^2}{2m}\nabla^2\ \mathbf{I}_2 + \mathbf{V}\ \mathbf{\Psi}_\nu = \epsilon_\nu \mathbf{\Psi}_\nu \tag{3.40}$$

where $\mathbf{I}_2$ is the unit-matrix. The kinetic energy part of the Hamiltonian is diagonal in the two-dimensional spin space. Only the off-diagonal part of the hermitian 2×2 potential matrix

$$\mathbf{V} = V\ \mathbf{I}_2 + \mu_B\ \sigma \cdot \mathbf{B}. \tag{3.41}$$

couples the two components of the Pauli spinor $\mathbf{\Psi}_\nu$. It is obtained from the density matrix, $\rho_{\alpha\beta}$, that is given by a very simple relation in terms of the solutions of the Kohn-Sham equation, $\mathbf{\Psi}_\nu = (\psi_{\nu,1}, \psi_{\nu,2})$:

$$\rho_{\alpha\beta} = \sum_{\nu=1}^{N} \psi^*_{\nu,\alpha}\psi_{\nu,\beta} \quad \text{with} \quad \alpha, \beta \in 1, 2. \tag{3.42}$$

It is related to the density and magnetization by

$$\rho = \frac{1}{2}\left(\ n\ \mathbf{I}_2 + \sigma \cdot \mathbf{m}\right) = \frac{1}{2}\begin{pmatrix} n + m_z & m_x - im_y \\ m_x + im_y & n - m_z \end{pmatrix}, \tag{3.43}$$

where $\mathbf{I}_2$ is the unit-matrix in spin-space and σ is the vector of the Pauli matrices.

magnetism requires several optional tags in the Fleur input file. To generate templates for these tags the input generator should be invoked with one of the command-line options `-noco` or `-explicit`. This will generate an enhanced `magnetism` section within the `calculationSetup` block and further `nocoParams` tags in each `atomGroups/atomGroup`. In the Fleur input file a noncollinear calculation is activated by setting the `calculationSetup/magnetism/@l_noco` switch to `T`.

Note, that in a non-collinear calculation you can monitor the convergence of all three parts of the density matrix, the two diagonal ones (spin 1 and 2) and one of the off-diagonals (spin 3). In non-collinear calculations core-tail corrections have to be switched off: `calculationSetup/coreElectrons/@ctail` has to be set to `F`.

When performing calculations with noncollinear magnetism in Fleur the magnetization in the interstitial region is treated as a continuous vector field while in the MT spheres the magnetization is approximated to be collinear and pointing in a fixed direction. Only the magnitude of the magnetization may change within a MT sphere. This is a reasonable approach for transition metals with small spin-orbit effects. The following figure illustrates this approach.

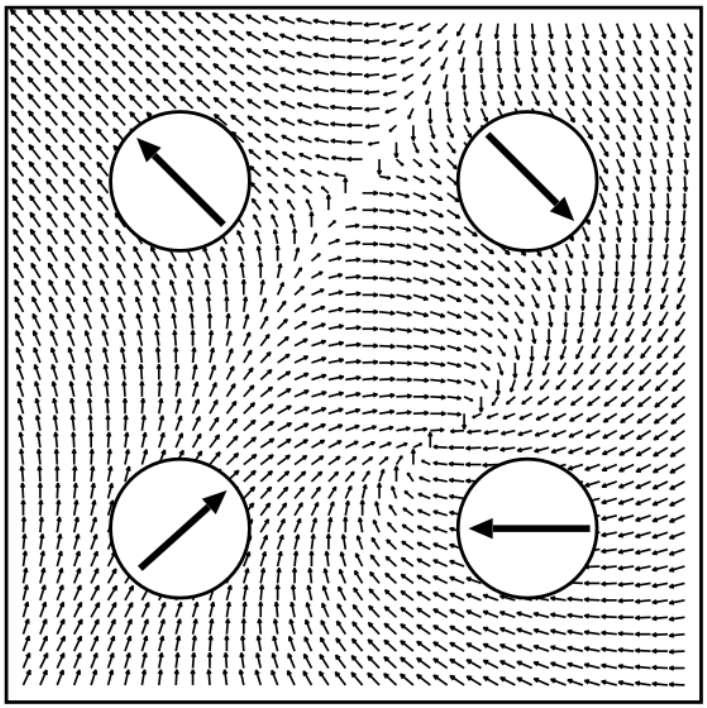

Figure 3.6: Illustration of the treatment of noncollinear magnetism in Fleur. Image taken from: Philip Kurz, Non-Collinear Magnetism at Surfaces and in Ultrathin Films (PhD Thesis, RWTH Aachen).

in the muffin-tin spheres is specified w.r.t. the global frame for the representative atom of each `atomGroup` in the associated enclosed `nocoParams` tag. The spin-quantization axis (SQA) is provided by the angles α (azimuthal angle similar to ϕ in the context of SOC) and β (polar angle similar to θ). The example below demonstrates the `nocoParams` for some atom group with defined $\alpha = \pi/2.0$ and $\beta = \pi/4.0$.

```
<nocoParams l_relax="F" alpha="Pi/2.0" beta="Pi/4.0"
```

```
b_cons_x="0.0" b_cons_y="0.0"/>
```

Assume that there are two symmetry equivalent atoms in the same `atomGroup` of a unit cell, then setting $\alpha = 0.0$ and $\beta = \pi$ describes an antiferromagnetic structure.

In detail the angles α and β define the direction of the exchange **B** field in the atoms at the start of a self-consistency step. After the diagonalization, the resulting charge density is not necessarily collinear in the muffin-tin spheres and even the average magnetization direction can point away from the defined SQA. It can happen (e.g. due to symmetry) that the magnetization stays in the direction of the SQA, but generally this will not be the case.

To consider the full non-collinear output magnetization density (also in the MT spheres) for a plot the switch `calculationSetup/magnetism/mtNocoParams/@l_mperp` has to be set to `T`. Afterwards one has to follow the normal plotting procedure for a density.

Note that if spin-orbit coupling is considered together with noncollinear magnetism, i.e., `calculationSetup/soc/@l_soc` is set to `T`, spin-orbit coupling is already treated in first variation. In this case `calculationSetup/cutoffs/@numbands` is not a relevant cutoff parameter.

magnetism the following calculation modes are not documented correctly.

Besides just keeping the angles α and β fixed there are two optional calculation modes that deal with output magnetization directions in the MT spheres that do not conicide with the input directions:

- Adjusting the SQA in the direction of the output magnetization
- Constraining the magnetization to the direction of the specified SQA

In both cases, it is necessary to know the component of the output magnetization that is perpendicular to the SQA. This is obtained by setting `calculationSetup/nocoParams/@l_mperp` to `T`.

In the first case, the **relaxation of the magnetic moments**, it is then sufficient to set for the respective atom groups `atomGroup/nocoParams/@l_relax` to `T`. After each iteration α and β are adjusted to the output magnetization, if the mixing parameter `calculationSetup/nocoParams/@mix_b` is set to `1.0` completely, for smaller parameters an intermediate direction is taken (zero means no relaxation). Be aware that such a relaxation can lead to slow convergence, sometimes it helps to remove the `mixing_history` from time to time.

For the case of **constraining the magnetization** one has to set the switch `calculationSetup/nocoParams/@l_constr` to `T`. Then, a constraining field $\Delta\mathbf{B}_c^\mu$ is calculated for each atom μ

$$\Delta \mathbf{B}_c^\mu = -\beta_c |\langle \mathbf{B}_{eff}^\mu \rangle| \frac{\mathbf{M}_\perp^\mu}{|\mathbf{M}^\mu|} \tag{3.44}$$

which is then added to the effective $\mathbf{B}$ field $\mathbf{B}_{eff}^\mu$. The ratio of perpendicular magnetization, $\mathbf{M}_\perp^\mu$ and magnetization along the SQA, $\mathbf{M}^\mu$, is proportional to the true constraining field but the value has to be determined self consistently (with β_c being the mixing factor `mix_b`). For each atom group the values of the constraining field are written as `atomGroup/nocoParams/@b_cons_x` and `atomGroup/nocoParams/@b_cons_y` in the inp.xml file for the next iteration.

3.11.1 Treatment of noncollinear magnetism inside the Muffin-tin sphere

Before discussing the setup and usage of this feature please be aware that this feature requires that FLEUR has access to a working hdf5 library.

Generating an input file using the `-noco` option of the input generator will also genreate some additional `nocoParams` in the `calculationSetup` section. The `nocoParams` which are relevant for setting up a calculation which includes noncolinearities inside the Muffin-tin sphere are `l_mtNocoPot` and `l_mperp`. Both of these must be set to `T` instead of the default `F`.

Setting the `calculationSetup/nocoParams/@l_mtNocoPot` switch to `T` allows offdiagonal elements of the potential inside the Muffin-tin sphere.

As with all noncolinear calculations in FLEUR `calculationSetup/coreElectrons/@ctail` has to be set to `F`. Furthermore `calculationSetup/magnetism/@jspins` should equal `2` if the input file has been generated correctly.

Including noncolinearities inside the Muffin-tin sphere is known to be the reason of convergence issues. We strongly recommend using LDA like funtcionals since GGA like functionals seem to compound the convergence issues. If GGA like functionals have to be used we recommend first using the `Anderson` mixing scheme and as soon as convergence problems appear switching to simple mixing.

After finishing an iteration the individual magnetization components (`mx`, `my` and `mz`) of each atom are written in the out file. Be aware that these values are calculated in the local frame of the atom which is either defined in the input file by using the angles `atomGroup/nocoParams/@alpha` and `atomGroup/nocoParams/@beta` or is subject to change if the relaxation feature activated by `calculationSetup/nocoParams/@l_relaxSQA` which keeps the z-axis of the local atom frame aligned to the direction of magnetization. If `l_relaxSQA="T"` the values of `mx`, `my` and `mz` should remain small at any time of the calculation. If one wants to know the orientation of the local frame while this feature is activated one has to take a look at the angles which are written in the out file

after each iteration along with the magnetization components. Those angles are called `nococonv%alpha` and `nococonv%beta` and are subject to change if `l_relaxSQA` is set to `T`.

Including noncolinear magnetism inside the Muffin-tin sphere into FLEUR calculations is known to work with spin-orbit coupling and local orbitals. The combination of LDA + U calculations and noncolinearities in the Muffin-tin sphere is possible but to achieve resonable results it is **mandatory** to set `l_relaxSQA="T"`. The reason for that is that the LDA + U matrix which is used in FLEUR is **always** calculated in the local frame. Therefore the charge density in the Muffin-tin needs to be kept in the same frame. This is achieved by relaxing the SQA in a way that the local frame of the atom aligns with the magnetization direction.

You can define `atomGroup/nocoParams/@alpha` and `atomGroup/nocoParams/@beta` at the beginning of a FLEUR calculation with `l_relaxSQA="T"` is set you should set **both angles in all atomgroups** to `0.0` **before** restarting the calculation. This also holds if you want to switch to `l_relaxSQA="F"` while restarting a calculation. For mixing purposes the charge density is stored unrotated in the cdn/cdn.hdf files.

The usage of spin-orbit coupling in combination with the relaxation `l_relaxSQA="T"` can cause severe convergence issues. In order to overcome those issues the parameters `calculationSetup/nocoParams/@mix_RelaxWeightOffD` has been introduced. The parameter `mix_RelaxWeightOffD` gives a weight to the magnetization components x and y. Default weight for those components is 1. However if e.g. spin-orbit coupling causes those components to appear suddenly this can cause large rotations of the spin-coordinate frames which is unfavoralbe for convergence. For sensitive system with large spin-orbit effects we recommend to choose the `mix_RelaxWeightOffD` parameter in the same magnitude as the usual mixing parameter used for your system.

3.12 Description of spin spirals

Spin spirals are periodic magnetic structures that can be described by a vector field. For a rotation axis along the z direction their magnetization $\mathbf{M}$ is given by:

$$\mathbf{M}_{n,\mu} = M \begin{pmatrix} \cos(\mathbf{q}\cdot(\mathbf{R}_n+\tau_\mu)+\alpha_\mu)\sin(\beta_\mu) \\ \sin(\mathbf{q}\cdot(\mathbf{R}_n+\tau_\mu)+\alpha_\mu)\sin(\beta_\mu) \\ \cos(\beta_\mu) \end{pmatrix} \tag{3.45}$$

where $\mathbf{R}_n$ is the lattice vector pointing from the origin to unit cell n, τ_μ is the position of atom μ in the unit cell, $\mathbf{q}$ is the so called spin-spiral vector, β_μ is the cone angle between the magnetic moment of atom μ and the rotation axis, and α_μ is an atom-dependent phase shift.

In general, the period length $\lambda = \frac{2\pi}{q}$ can be many times larger than a lattice period. Hence it is not feasible to simulate a spin spiral directly because one would need to use extremely large supercells. However, **in the absence of spin-orbit coupling**, the generalised Bloch theorem allows to overcome this problem. The theorem states that a wave function of a system featuring a spin spiral can be represented as:

$$\psi_{\mathbf{k}}(\mathbf{r}) = \begin{pmatrix} e^{i(\mathbf{k}-\mathbf{q}/2)\cdot\mathbf{r}} u^{\uparrow}_{\mathbf{k}}(\mathbf{r}) \\ e^{i(\mathbf{k}+\mathbf{q}/2)\cdot\mathbf{r}} u^{\downarrow}_{\mathbf{k}}(\mathbf{r}) \end{pmatrix} \tag{3.46}$$

treating noncollinear magnetism [1]. This implies that one should invoke the input generator with the `-noco` or `-explicit` command line switches to obtain templates for the parametrization of calculations on noncollinear magnetism. Next, the treatment of noncollinear magnetism has to be activated by setting `calculationSetup/magnetism/@l_noco` to `T`. The angles α_μ and β_μ are set in the `atomGroups/atomGroup/nocoParams/@alpha` and `atomGroups/atomGroup/nocoParams/@beta` attributes. To finally switch on the calculation of a spin spiral by employing the generalized Bloch theorem also `calculationSetup/magnetism/@l_ss` has to be set to `T`. The following example demonstrates such a setting.

```
<magnetism jspins="2" l_noco="T" l_ss="T">
   <qss>0.25 0.25 0.0</qss>
</nocoParams>
```

The **q** vector is set in the `qss` tag enclosed by `calculationSetup/magnetism` and measured in reciprocal lattice vectors.

cell are not incompatible to the **q** vector, the inpgen input should be extended by a line specifying the **q** vector. For the here-considered **q** this line is `&qss 0.25 0.25 0.0 /`.

Under the assumption that the setup in the example above belongs to an fcc unit cell with the Bravais matrix

$$A = \frac{a}{2} \cdot \begin{pmatrix} 0.0 & 1.0 & 1.0 \\ 1.0 & 0.0 & 1.0 \\ 1.0 & 1.0 & 0.0 \end{pmatrix} \tag{3.47}$$

the reciprocal unit cell would be defined by the matrix

$$B = \frac{2\pi}{a} \cdot \begin{pmatrix} -1.0 & 1.0 & 1.0 \\ 1.0 & -1.0 & 1.0 \\ 1.0 & 1.0 & -1.0 \end{pmatrix} \tag{3.48}$$

which defines a bcc-like structure. This implies that the defined $\mathbf{q} = (0.25, 0.25, 0.0)$ would point in z direction for this example.

The magnetic moment of an atom at position τ (internal coordinates) is rotated by $2\pi(\mathbf{q} \cdot \tau)$. For the given example and $\beta = \pi/2$ this implies that an increase of the $\mathbf{q}$ vector to $\mathbf{q} = (0.5, 0.5, 0.0)$ would yield a layered structure with an antiferromagnetic alignment of the magnetization directions along z.

The following figure demonstrates different spin spirals that may be obtained with respective parametrizations in the Fleur input file.

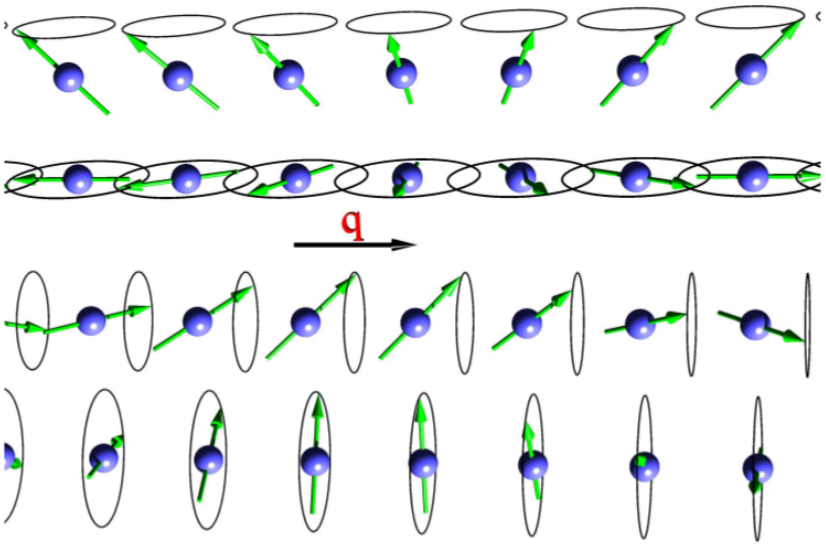

Figure 3.7: Four examples of spin spirals with spin-rotation axis perpendicular (upper two) and parallel (lower two) to the spin-spiral vetor $\mathbf{q}$. For each case two spirals with angles of $\beta = \pi/4$ and $\beta = \pi/2$ between the magnetic moment and the rotation axes are shown. Figure taken from: Philip Kurz, Non-Collinear Magnetism at Surfaces and in Ultrathin Films (PhD Thesis, RWTH Aachen).

In the absence of spin-orbit coupling (SOC) the direction of the rotation axis is energetically irrelevant, therefore the first and third spin-spiral in the figure are equivalent (the same applies to the second and fourth one). For convenience, if no SOC is considered the **rotation axis is always set to point in z-direction**. I.e., if the magnetization of an atom points along z, it will not rotate. If it points in x-direction ($\beta = \pi/2$), it describes a flat spiral (second example of the figure).

to the inclusion of spin-orbit coupling in the calculation. If spin spirals have to be calculated under consideration of SOC this can be done by employing the magnetic force theorem .

spin-spiral calculations are incompatible to the exact treatment of the core-electron tails reaching out of the MT spheres. Thus `calculationSetup/coreElectrons/@ctail` has to be set to `F`

Since the number of basis functions changes (slightly) with **q**, for consistent output quantities it is important that the basis cutoff ($K_{\max}$) is not too small, typically increased by 0.5 - 1.0 as compared to normal non-collinear calculations. Also, a dense **k**-grid is recommended when sampling **q** space, usually twice as dense as the **q**-grid works best [3].

Further reading

1. P Kurz, F Förster, L Nordström, G Bihlmayer, S Blügel, Physical Review B 69 (2), 024415 (2004)
2. M Heide, G Bihlmayer, S Blügel, Physica B: Condensed Matter 404 (18), 2678-2683 (2009)
3. M Ležaić, P Mavropoulos, G Bihlmayer, S Blügel, Physical Review B 88 (13), 134403 (2013)

4 Basic calculations

Contents

Within this chapter the workflows for basic Fleur calculations are described. This covers calculation procedures that mainly consist of certain Fleur calls. The calculation of quantities that require more elaborate workflows involving considerable pre- or postprocessing are not covered. Such workflows would typically be implemented by making use of the Aiida-Fleur interface.

4.1 Using the input generator

Performing Fleur calculations on some material always starts with the specification of a (prototype) unit cell. With this basic specification the Fleur input generator `inpgen` then generates a *Fleur input file* with a reasonable parametrization for the provided unit cell. In this section we discuss the basic usage of the input generator.

In general the input for the Fleur input generator consists of the definition of the unit cell shape as well as the definition of the atoms and atom positions in the cell. Optionally some additional parameters can be set to override or modify the automatic procedure used to determine the parametrization in the Fleur input file.

In the following the usage of the input generator is sketched in terms of a basic example. For a reference about the details of the inpgen input please see the respective section .

Using the input generator starts by writing a small text file containing the configuration of the unit cell. For a simple Si crystal such a file might look like:

```
Si bulk
&lattice latsys='cF', a0=1.8897269, a=5.43 /
2
14 0.125 0.125 0.125
14 -0.125 -0.125 -0.125
```

The first line of the file contains a comment. Here this comment is `Si bulk`. Next comes the definition of the crystal lattice. This is done in the `&lattice` line. It starts with the specification of the lattice system in `latsys='cF'`. `cf` specifies an fcc lattice. The line is completed with the setting of some parameters. In this case the parameter `a` is the lattice parameter of the fcc lattice and `a0` is a factor that is multipied to all lattice parameters. Note that the input generator expects the lattice specification in units of Bohr radii (a_0). But since in this example `a` is provided in Angstrom we need the factor `a0` to convert it to units of a_0.

In the next part of the inpgen input the atom positions are specified. This starts by providing the number of atoms in the next line. For this example system there are `2` atoms in the unit cell. The following lines define the atomic numbers and related atom positions in internal coordinates, i.e., in coordinates of the Bravais lattice vectors. `14` is the atomic number of Si and the two atoms are at $(1/8, 1/8, 1/8)$ and $(-1/8, -1/8, -1/8)$.

Consider that the name of the `inpgen` input file is `inpSi.txt`, a call to `inpgen` may look like:

```
inpgen -f inpSi.txt
```

With this command `inpgen` will generate a basic Fleur input file `inp.xml`. In addition a file `sym.xml` containing the symmetry operations and a file `kpts.xml` will be generated. These can be included into the `inp.xml` as well See corresponding documention for inpgen). Furthermore a file `struct.xsf` is generated. This is an XCrysden structure file that can be used to visualize the just defined unit cell, e.g., to check for complex structures whether the setup is as expected. The output of the input generator can be found in the file `out`. If there are problems in generating the Fleur input file hints for the reason might be found in this file or directly on the terminal. The generated file `FleurInputSchema.xsd` is temporary. It is generated in every `inpgen` and `fleur` run and only contains the formal specification of the Fleur input file format.

Of course, there are many possible options of how to write an `inpgen` input. For the details the user can consult the `inpgen` reference section . A collection of (advanced) example `inpgen` inputs is available at the examples page.

4.2 The standard self-consistent field (SCF) calculation

Directly after running the Input generator `inpgen` the Fleur input file is set up such that Fleur will perform an SCF calculation with it.

The calculation of a self-consistent density starts with the construction of a starting density which is automatically generated if no other density is present in the working directory. It is a superposition of atomic densities generated on the fly. If there already is a working density in the calculation directory Fleur starts the calculation with this density.

> stored in the file `cdn.hdf`. Without HDF5 the working density is stored in the file `cdn1` and previous densities are stored in the files `cdn??` where the question marks stand for two digits following the cdn. Note that the working densities should be consistent with the input file in the same directory.

In general an SCF run in Fleur follows the flow chart sketched in figure 1. After entering the SCF loop the current density is used to construct an effective potential for which the Kohn-Sham equations are solved. From the resulting eigenfunctions a new density is generated and it is checked whether it is similar enough to the input density auch that a self-consistent solution is found. If this is not the case a new density has to be constructed from all densities generated up to this point and another SCF iteration is started with this new density. If a self-sonsistent solution is found the program exits the loop.

The processing of the SCF loop is directly affected by several parameters in the input file. The most important parameters determine when the program comes to a controlled

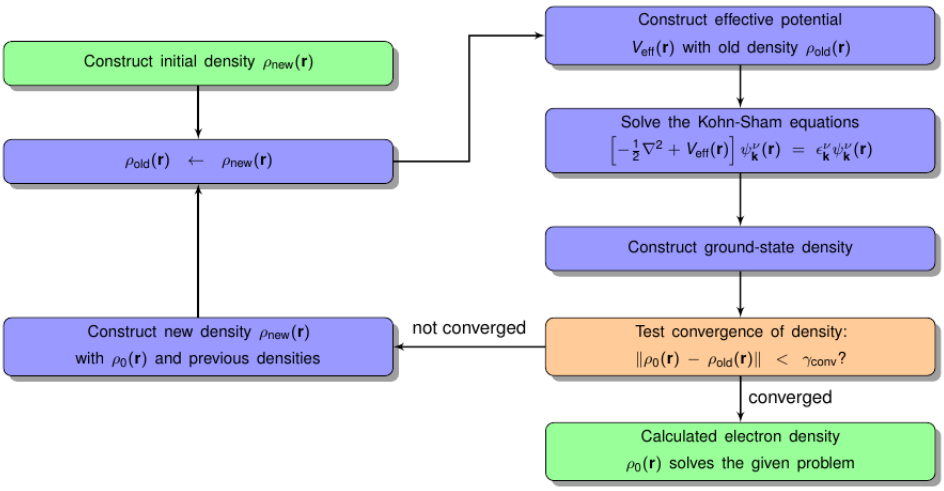

Figure 4.1: The self-consistent field cycle.

end. For this one can define the maximal number of iterations to be performed and also a convergence criterion that leads to an immediate stop.

be performed is specified in `calculationSetup/scfLoop/@itmax`. The convergence criterion defining the allowed distance between the input and output densities for a converged solution is specified in `calculationSetup/scfLoop/@minDistance`. By default a distance of less than $10^{-5}\text{me}/a_0^3$ is considered to indicate a converged calculation. Slightly larger distances are also often tolerable.

Note that the presence of these two halting criteria implies that a finished Fleur run does not necessarily result in a self-sonsistent density. One should always check the distances between the densities for the last performed iteration.

written out to the `out` file as well as to the `out.xml` file. One can easily see the distances from all iterations by executing `grep distance out`.

Beyond the halting criteria the configuration of the density mixing scheme plays an important role for SCF calculations. This is set by specifying the general type of the mixing procedure on the one hand and by parametrizing the mixing procedure on the other hand. For the parametrization a single mixing parameter α_{mix} has to be specified.

`calculationSetup/scfLoop/imix`. The `Anderson` mixing typically is a very efficient choice. If reaching convergence is difficult it is sometimes reasonable to perform parts of the calculation with `straight` mixing. The mixing parameter is set in `calculationSetup/scfLoop/alpha`.

Several mixing schemes construct the next density on the basis of all densities calculated up to some point. Since the model assumptions for these mixing schemes might not be perfectly fulfilled it may be a good idea to delete this history of previous densities from time to time.

`calculationSetup/scfLoop/maxIterBroyd` specifies after how many SCF iterations the mixing history is reset. Furthermore whenever a Fleur SCF run halts without obtaining a self-consistent density a file `mixing_history` with the mixing history up to this point is written out. It is read in again if Fleur is started in the same directory. By deleting it Fleur starts with an empty mixing history.

For metallic materials the Kerker preconditioner might speed up SCF convergence. Usage of the preconditioner becomes the more important the larger the system's unit cell or film thickness is. In Fleur it is currently available for non-magnetic and collinear-magnetic structures.

parameter `calculationSetup/scfLoop/precondParam` has to be set to a positive value. The optimal parameter is material-dependent and is about 0.8 for most metals.

4.3 Obtaining the band structure

The starting point for the calculation of a band structure is a finished Fleur calculation yielding a self-consistent density. This means that a charge density file with this density should be present in the working directory and be consistent with the input file.

The main difference between a band structure calculation and an SCF calculation is that different **k**-point sets have to be used. While a good SCF calculation uses a **k**-point set that representatively samples the Brillouin zone, the **k**-point set for a band structure calculation has the form of a **k**-point path along high-symmetry lines of the Brillouin zone.

k-point set usefull for band structures by default. You can find them in the `kpts.xml` file with a `.../kPointList/@type` parameter "path". In addition you can use the input generator to create further **k**-point sets. The **k**-point set for a Fleur run is specified by its name in `cell/bzIntegration/kPointListSelection/@listName`. For each **k**-point list the name is specified in `.../kPointList/@name`, typically in the `kpts.xml` or directly in the `inp.xml` file.

After defining the **k**-point set for the band structure the band-structure calculation has to be switched on.

`output/@band` switch has to be set to `T`. Then Fleur has to be started in the respective directory. Since every new Fleur run overwrites the old output files make sure that you create backups of these files before starting the band structure calculation. The data from the `out` and `out.xml` files may be needed in the postprocessing to the band structure calculation.

There are two sets of output files for the bandstructure data.

4.3.1 Simple text file output

The simplest band-structure output is stored in several files: * The `bands.1` (and `bands.2`) files for the data. * A `bands.gnu` file containing a simple gnuplot script to generate the bandstructure.

the file `bands.1` in terms of two columns with an x coordinate in the first column and an energy in eV in the second column. For magnetic calculations this file contains the data for the first spin while the data for the other spin is stored in the file `bands.2`.

Note that the Fermi energy calculated in Fleur depends on the **k**-point set. If the Brillouin zone is well-sampled one obtains a good value for the Fermi energy. **k**-point sets for band-structure calculations are not made for such a Brillouin zone sampling and the quality of the Fermi energy depends on whether the highest occupied state is near the high-symmetry **k**-point path or not. Therefore one has to pay attention to correct the raw band structure data to the correct Fermi energy. This can be performed while plotting the data.

script `band.gnu`. It can be used by invoking `gnuplot < band.gnu > bands.ps` and afterwards converting the postscript file into a PDF with `ps2pdf bands.ps`. The correction offset can directly be inserted into the plotting command in the script. The Fermi energy from the last calculation can be obtained from the `out.xml` file by invoking `grep FermiEnergy out.xml`. Depending on whether a `cdn1` or a `cdn.hdf` file is used some correction of the Fermi energy is automatically performed in the raw data. In the case of the `cdn.hdf` file the energy alignment of the raw data is already corrected. Details on the correction are written out to the terminal when the band-structure files are generated. In the case of the `cdn1` file it is aligned to the Fermi energy of the band structure calculation. A Fleur output to the command line will inform you about this.

There are also cases in which the Fermi energy from the band structure calculation is the better choice, i.e., if the highest occupied state is directly on the chosen **k**-point

path. But the Fermi energy from the SCF calculation can systematically be converged by choosing ever finer **k**-point meshes.

The following figure shows the band structure for Silicon using the default **k**-point path. It was directly created with the gnuplot script mentioned above. No correction of the Fermi energy had to be performed here because the highest occupied state is located at a **k** point included in the default path.

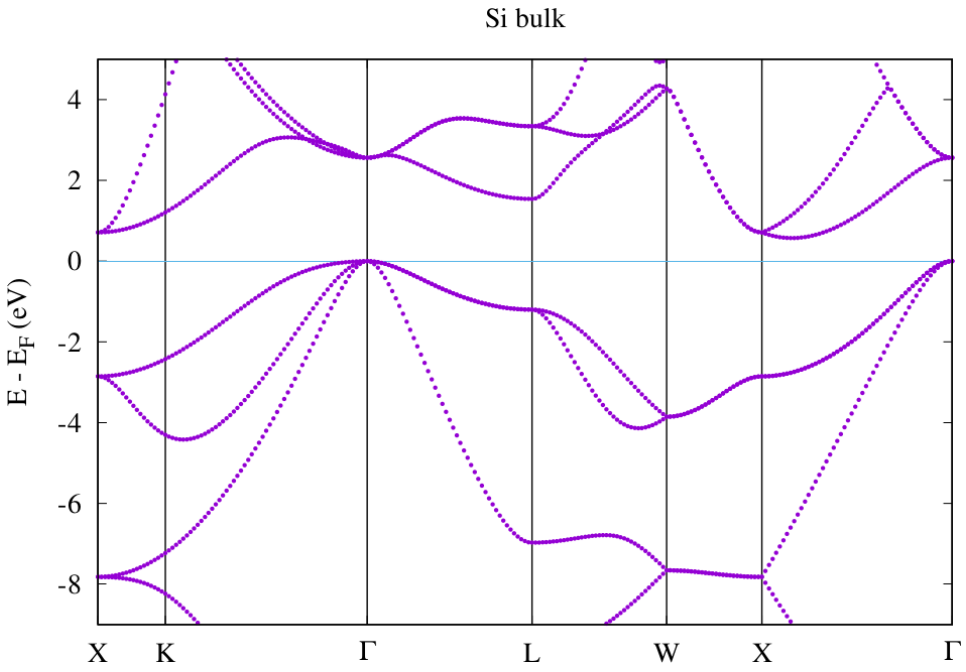

Figure 4.2: Si band structure

4.3.2 Bandstructure output in the banddos.hdf file

In case the code is linked against the hdf5 library, the `banddos.hdf` file is written. It contains all the data needed to create a band structure and also weights for each state that can be used to create band structures with additional encoded information. The weights available here are the same as those used in the DOS mode .

4.4 Obtaining a density of states (DOS)

The calculation of a density of states (DOS) is based on the presence of a self-consistent density in the working directory. With this starting point a single modification of the Fleur input file then activates the DOS generation in the next Fleur run.

`output/@dos` has to be set to `T`.

The output of a DOS calculation depends on several parameters. Most important is the range of the energy mesh on which the DOS is calculated.

parameters `output/bandDOS/@minEnergy` and `output/bandDOS/@maxEnergy` which have to be provided in Htr. To get an estimate on a possibly required offset it is a good idea to check the value of the Fermi energy with `grep FermiEnergy out.xml`.

The generation of a DOS relies on the chosen **k**-point set used to sample the Brillouin zone. Since this is always finite the contributions comming from each calculated state have to be broadened in energy to avoid the calculation of an overly spikey DOS that is mainly an artifact of the Brillouin zone sampling. For this the DOS is convoluted with a Gaussian distribution.

in `output/bandDOS/@sigma`.

Note that the optimal choice of the standard deviation depends on the fineness of the **k**-point mesh and also on material properties like the dispersion of the bands in the chosen energy range. The choice of the σ parameter therefore cannot be done automatically but is a task for the user. A too small σ yields an overly spikey DOS while a too large σ yields an overly smooth DOS that does not show any details. The following figure demonstrates this interrelation. It shows the total DOS for Si as well as the projections of the DOS onto the s and p states in one of the MT spheres. For the construction of the DOS the left plot uses the default parametrization, while the other two plots modify the σ parameter and the **k**-point mesh for the DOS calculation (not for the SCF run).

It can be seen that the default sigma parameter here yields an overly smooth DOS that abstracts from the details. Not even the band gap can be estimated in this way. When reducing the σ parameter the DOS becomes very spikey because the **k**-point density is too low to actually resolve a DOS on such a fine energy mesh. Only a reduction of σ together with an increase of the **k**-point density yields a DOS that is not overly spikey but resolves the details of the electronic structure. But even in this plot the **k**-point density is not high enough to completely overcome the spikes in the range of the unoccupied bands. Of course, it is expected that such artifacts especially show up in the range of the unoccupied bands because the dispersion for unoccupied bands is typically larger than for occupied bands.

The DOS can also be obtained by employing the so called tetrahedron method for Brillouin zone integration. In this method the irreducible wedge of the Brillouin zone (IBZ) is decomposed into small tetrahedra. The eigenvalues and all related properties

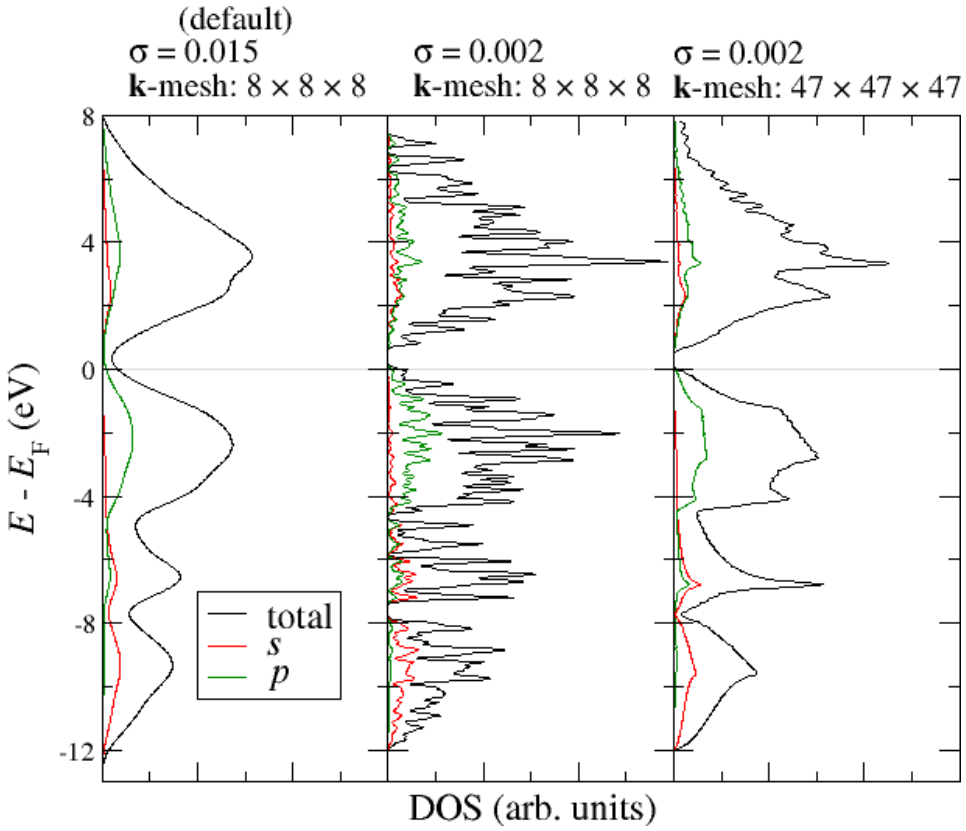

Figure 4.3: DOS for Si with different parameters.

are only obtained at the corners of each tetrahedron. Then the eigenvalues are linearly interpolated inside the tetrahedron.

`cell/bzIntegration/@mode` keyword to `tria`. In addition a corresponding k-point set should be used. You can use inpgen to generate such a k-point set.

The following figure demonstrates the effect of this method on the DOS. Both calculations use the same charge density and the same number of **k**-points inside the IBZ.

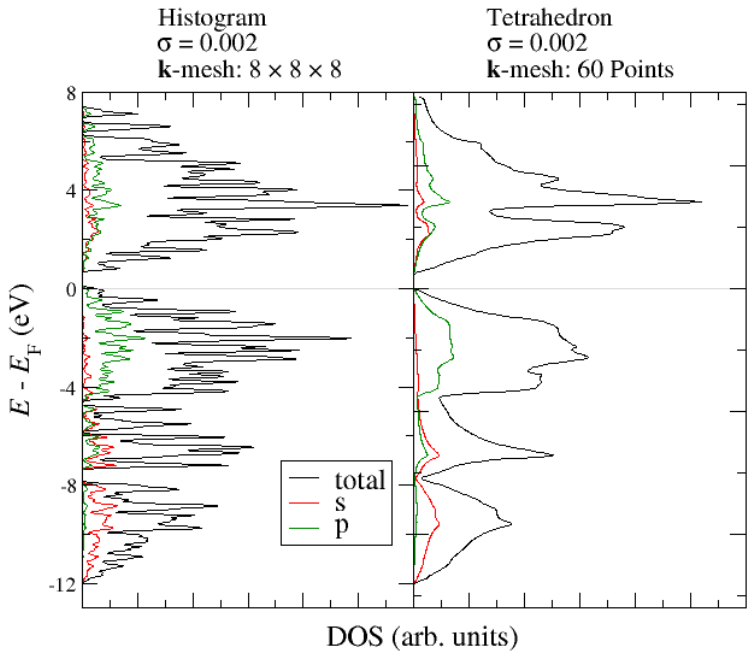

Figure 4.4: DOS for Si with histogram and tetrahedron method.

The differences between the above used histogram method and the tetrahedron method can be seen very clearly. Already with the relatively small amout of **k**-points the DOS obtained from the tetrahedron method shows very clear features similar to the DOS with the high **k**-point density above. Note however that the choice of states for the interpolation within each tetrahedron is naive. It is performed between the i-th eigenstates at the contributing k points. This ignores the possibility of band crossings and a DOS generated in this way may show nonphysical gaps if combined with a too small smoothening parameter sigma.

`LOCAL.1` for nonmagnetic calculations and an additional file `LOCAL.2` for the second spin in spin-polarized calculations. These files consist of multiple columns of values, where the first column provides the energy and all other columns provide projected DOS values.

note More info
This document only shows the most basic usage ofthe DOS feature. For more details please consult the corresponding document in the reference section and have a look at the section on the extraction and visualization of data stored in the banddos.hdf file .

4.5 Performing structural relaxations

The geometry of crystal unit cells features many degrees of freedom. The most basic of these are the Bravais lattice vectors, which can be described by the lattice type together with the lattice parameters. Next the atom positions in the unit cell also represent geometrical degrees of freedom and it may even be that the groundstate structure of the crystal is only describable in a supercell because the atom positions in neighboring unit cells interact.

The Fleur code provides a feature to semi-automatically optimize the atom positions for a given unit cell shape by minimizing the forces acting on the atoms. The other degrees of freedom have to be investigated by the user explicitely if they are of interest. Especially the lattice parameters have to be optimized manually by performing a series of calculations with test lattice parameters and extracting from these the optimal parameters minimizing the total energy. Note that all geometrical degrees of freedom may interact with each other. Also the magnetic ground state configuration may affect the optimal geometrical parameters.

can be extracted from the `out.xml` file by invoking `grep "<totalEnergy" out.xml`. A listing of the individual energy contributions is also provided within the `totalEnergy` section.

The workflow to optimize the atom positions essentially works such that Fleur is invoked several times and each time an SCF calculation with a subsequent calculation of the forces and the generation of suggested atom displacements is performed. In a preprocessing step to this workflow and other structural relaxations the Fleur input file should be adapted such that the MT spheres are slightly reduced in size to allow a movement of the atoms without obtaining overlapping spheres.

`atomSpecies/species/mtSphere/@radius`. Default radii are automatically determined by the Fleur input generator and may subsequently by modified by the user. Since the default parameters lead to nearly overlapping spheres it is a good strategy to reduce them to about 95% before performing structural relaxations. The input file understands mathematical expressions so that the user does not have to perform the involved calculations by manually. An example for the reduction of a MT radius to 95% is provided below.

```
<mtSphere radius="0.95*2.2200" gridPoints="733" logIncrement=".0160"/>
```

The reduction of the MT radii should be performed for the optimization of the atom positions as well as for manual optimizations of the lalltice parameters. Note that total energies from calculations with different MT radii are typically not comparable since the quantity of interest are energy differences between the calculations and not total energies of individual calculations. Total energy differences are typically converged with smaller basis set sizes and thus the total energies themselves are typically underconverged. If it turns out that the reduction of the MT radii was not significant enough and one still obtains overlapping MT spheres in a certain calculation probably all calculations have to be performed again with smaller MT radii. It is therefore advisable to start with a geometry of the unit cell that is already near the expected optimum.

radii. If the radii or other geometrical parameters are changed it is therefore important to also delete the charge density file if there already is one in the directory.

The next step for the optimization of the atom positions is to select the atoms for which the forces are to be calculated and the directions in which the force relaxations may be performed.

directions for force relaxations is performed for each group of symmetry equivalent atoms separately. The switch to calculate forces for the atoms of a certain group is found in `atomGroups/atomGroup/force/@calculate`. Note that even if this switch is set to `T` all force contributions are only calculated if a structural optimization of the atom positions is activated. The directions for which the relaxations are to be performed are selected in `atomGroups/atomGroup/force/@relaxXYZ` by three logical values for the x, y, and z direction. An example `force` tag used for the relaxation of the z coordinate only is presented below. Such a relaxation is often used when calculations for thin film systems have to be performed.

```
<force calculate="T" relaxXYZ="FFT"/>
```

Different iterative optimization schemes can be used to find the atom positions featuring vanishing forces. In each force iteration they generate a new displacement of the atom positions from the original input file based on the position-dependent forces calculated up to that point. A very efficient optimization scheme for this purpose is the BFGS algorithm. But also a straight linear mixing may be useful in certain situations. These optimization schemes are parametrized by a mixing parameter α_{force} and convergence criteria relating to the maximal change of the positions between the last two force iterations and the maximal forces.

`calculationSetup/geometryOptimization/@forcemix` parameter. The default `BFGS` entry selects the BFGS algorithm, `straight` selects a straight linear mixing. The mixing parameter is set in `calculationSetup/geometryOptimization/@forcealpha`, the convergence criterion for the change of the atom displacements in `calculationSetup/geometryOptimization/@epsdisp`, and the convergence criterion for the forces in `calculationSetup/geometryOptimization/@epsforce`.

After selecting and parametrizing the force optimization scheme. The only missing step is the activation of the force optimization procedure.

`calculationSetup/geometryOptimization/@l_f` switch. An example for the `geometryOptimization` tag with activated force optimization is provided below.

```
<geometryOptimization l_f="T" forcealpha="1.00" forcemix="BFGS"
                      epsdisp=".00001" epsforce=".00001"/>
```

After activating the force optimization starting Fleur yields an SCF calculation. When this calculation reaches the convergence criterion for the minimal distance between the charge densities the forces are calculated and a new atom displacement is calculated.

positions provided in the Fleur input file in the file `relax.xml`. By default this file is included into the Fleur input file in terms of an XML include tag. If no such file is present in the working directory or the inlude tag has been removed no displacements of the atom positions are taken into account. The file includes a list of diplacements for the representative atoms of each atom group in `relaxation/displacements/displace`. The order of displacements has to be consistent with the order of the respective atom groups. For every force relaxation step the file also contains a `relaxation/relaxation-history/step` section. This section provides the total energy related to the SCF run for the respective atom positions in `relaxation/relaxation-history/step/@energy` and also for each atom group a `relaxation/relaxation-history/step/posforce` element containing the x, y, z coordinates of the representative atom followed by the forces on this atom in x, y, and z direction. The `relaxation/relaxation-history` is considered for the mixing part of the chosen force optimization algorithm. After performing the force optimization the final atom positions can also easily be extracted from this section.

Performing a force optimization of the atom positions for a given unit cell shape therefore mainly reduces to invoking Fleur several times with activated geometry optimization. In each force calculation step the atom displacements as well as the relaxation history are updated. They are automatically included in the following Fleur run.

If Fleur is started with an old charge density file in the working directory the starting density is taken from this file, even if the atom positions have changed. This may be a very good starting point if the change of the atom positions is small. In such a situation the number of required iterations to reach a self-consistent density is strongly reduced. If the shifts in the atom positions is large this approach can be a catastrophy. It may lead to a slower convergence or even to an error in the calculation because the potential obtained from the stored density is too inconsistent to the actual atom positions. If a problem like this is observed it may be a good idea to delete the charge density file and automatically construct a new starting density for the current atom positions.

It can also be that the relaxation-history gives rise to large atom displacements. This is because the model assumption in optimization algorithms typically is that the parameters of the function to be optimized are near an optimum and that the function can be approximated by a parabolic model near this optimum. If this is not fulfilled the hints extracted from the relaxation-history may be contradictory ending in an extreme shift of the next atom position. In such a situation the advanced user can delete parts of the relaxation-history by hand or switch to a straight linear optimization scheme until the parabolic model assumptions are better fulfilled.

Besides these challenges during the force optimization the choice of the initial atom positions is also crucial. As already mentioned it is of benefit to start from atom positions that are already near their optimum. Beyond this the user should pay attention that

the starting atom positions do not preserve symmetries that are not featured by the possible optimized positions. Otherwise one may end up with vanishing forces because the initial positions mark a local maximum in the energy landscape, not because they mark a minimum.

Note that forces are defined as the negative variation of the total energy with respect to the atom positions. This implies that the demands for high-quality force calculations are stricter than for the simple calculation of total energy differences. Especially due to the atom position dependence of the LAPW basis and the partitioning of the unit cell care has to be taken to ensure that the forces are carefully converged with respect to the calculation setup. This implies that the need to describe semicore states by LOs as valence electrons is increased. Also the demands for the cutoff parameters of the LAPW basis may be increased.

4.6 Usage of the LDA+U approach

It is possible to perform Fleur calculations incorporating Hubbard U parameters for certain orbitals at certain atoms. The implementation of the approach complies with the publication *Shick et al., Implementation of the LDA+U method using the full-potential linearized augmented plane-wave basis, Phys. Rev. B 60, 10763 (1999).* This implies that the U parameter actually doesn't act on certain Kohn-Sham orbitals but on those parts of the LAPW basis functions featuring the angular momentum quantum number for which the U parameter is specified.

parameter for a certain l quantum number in the MT sphere is straight forward. For this an `ldaU` tag has to be inserted below the `energyParameters`, `electronConfig`, and `nocoParams` tags and above the `lo` tags. The `ldaU` tag includes specifications for the U parameter in `ldaU/@U`, and specifications for the J parameter in `ldaU/@J`. Both parameters have to be provided in eV. Additionally it has to be specified in `ldaU/@l_amf` whether the U is treated in terms of the around-mean-field limit (`T`) or in terms of the atomic limit (`F`). For each atom species up to 4 different U parameters can be specified, each within a separate `ldaU` tag. An example for the specification of a U parameter of 8 eV and a J of 0.9 eV for $l = 2$ is shown below.

```
<ldaU l="2" U="8.0" J="0.9" l_amf="F"/>
```

In DFT+U calculations the additional Hamiltonian contributions due to the U parameter depend on the density matrix for the considered angular momentum quantum number in the respective MT sphere. Similar to the density this matrix also has to be determined self-consistently.

specified in the separate `calculationSetup/ldaU` tag. One can choose to either perform a linear mixing by setting `calculationSetup/ldaU/@l_linMix` to `T` or to perform the mixing consistently with the density by setting it to `F`. For linear mixing the mixing parameter is defined in `calculationSetup/ldaU/@mixParam`. An example for the setup of a `calculationSetup/ldaU` tag is provided below.

```
<ldaU l_linMix="F" mixParam="0.05" spinf="1.00"/>
```

MT sphere the $(2l + 1) \times (2l + 1)$ density matrix (one entry for each m, m' combination) is written out to the file `n_mmp_mat_out`. The format of the matrices are 7×7 blocks of space separated complex numbers where each line of the matrix stretches out to two lines in the file. The real and imaginary part of the entries are also space separated. If a file `n_mmp_mat` is provided in the working directory the density matrix provided in that file is taken as an input and the file is directly renamed into `n_mmp_mat_old` so that it is not read in again in the next iteration. Advanced users may define density matrix starting points in this way. The density matrix is also written to the `out.xml` file within `densityMatrixFor` sections. The attributes of these tags uniquely identify a density matrix by providing spin, atom type, index of the U parameter, l quantum number, U parameter, and J parameter.

Note: If a U parameter and a local orbital (LO) are defined for the same atom species and l quantum number, Fleur will write out `LO and LDA+U for same l not implemented`. Nevertheless you can use Fleur in this way. What is not implemented is the effect of the U parameter on the extra radial function used for the LO. This is not problematic if the LO is used to represent a semicore state and the U parameter is supposed to have an effect on higher lying valence or conduction band states. In fact, in such a situation this code behavior is explicitly wanted. On the other hand if the LO and the U are made for the same states in mind the calculation will not work as intended.

The user should also be aware that implementations of the LDA+U approach in FLAPW typically only affect the MT spheres. This implies that depending on the localization of the states adequate U parameters may slightly depend on the MT sphere radii. In general U parameters used for calculations with different DFT implementations are not necessarily quantitatively comparable.

4.7 Plotting densities and potentials

The FLEUR code offers several options of plotting the densities and potentials. The feature is MPI and OMP parallelized and can be applied to systems of reasonable size to gain insight about their electronic structure. However, care needs to be taken when

choosing the number of MPI processes. It should be a divisor of the number of grid points in z-direction.

The corresponding switches can be found in the `output` section of the Fleur input file under

```
<plotting iplot="0" polar="F" format='1'>
</plotting>
```

With the above information, no plot will be created yet. To do so, first run a normal self-consistency calculation and then modify the inp.xml by setting `iplot` to a particular integer. Another iteration will be executed during which several quantities can be plotted out. The new density from this iteration is not written into the density file. The plottable quantities are:

- The input density as generated by the density file from the self-consistent run [id=1, name=denIn].
- The total potential calculated from the input density that goes into the Kohn-Sham-Equations [id=2, name=vTot].
- The Coulomb potential calculated from the input density [id=3, name=vCoul].
- The exchange-correlation potential [id=4, name=vXc].
- The output density calculated before it is mixed with the input for the next iteration [id=5, name=denOutWithCore].
- The mixed density calculated for the next iteration [id=6, name=denOutMixWithCore].

To control the options via a single integer switch, `iplot` is calculated by binary addition from the plot ids given above. For example, if you want to plot the input density and the resulting potential, then iplot=2^{1+2}2=2+4=6. Optionally, `iplot=1` or any odd number will plot all options coded into FLEUR. The `polar` switch will, when set to true, generate additional plots in the non-collinear case, id est for example the magnetization density components in x-, y- and z-direction will be supplemented by the absolute values |m| and the angles θ and ϕ, giving the polar representation. `format` can either be set to 1 or 2. 1 gives .xsf files to be used with XCrySDen and 2 will print out all available information and coordinates as a tabloid.

To generate plots, another tag needs to be added into the inp.mxl file in the plotting section:

```
<plot cartesian="F" TwoD="T" grid="30 30 30" vec1="1.0 0.0 0.0" vec2="0.0 1.0 0.0" vec
```

There can be multiple tags of this form to make plots with different characteristics. If at least one tag is put in, a plot will be made, otherwise setting `iplot` to a non-zero integer will have no effect. The various plotting options can now be specified. Note, that not all particular switches need to be given, as defaults exist. The examplary tag contains the default options coded into FLEUR. The tag contains the following options:

- cartesian: Are the following options given in internal or cartesian (Bohr radii) units?
- TwoD: Determines whether a 2-dimensional or 3-dimensional density plot is created.
- grid: Number of grid points in each direction.
- vec: Vectors spanning the part of the unit cell that should be plotted.(*)
- zero: Origin shift of the plot.
- file: Name of the plot. This will be [part of] the filename for non-xsf or vector plots and an identifying tag in xsf-datagrids otherwise.
- onlyMT: When set to true, all gridpoints not in the Muffin Tins will have zero density to better visualize the atomic densities.
- typeMT: More specific than the last switch, this one will set everything to zero except for a single type of atoms specified by an integer n.

Note, that cartesian units within FLEUR are done in units if the Bohr radius a_0, while XCrySDen operates in Angstrom. All output files will have their coordinates written in the latter.

To better understand which files are written, here are some examples for the case of a single 3D plot with `iplot='2'`, `polar='F'` and `format='1'` in the `plotting` tag and `TwoD='F'` for the different levels of magnetism:

- non-spinpolarized: `denIn.xsf` with a single `DATAGRID` called "plot".
- collinear: `denIn_f.xsf` and `denIn_g.xsf` each with a single `DATAGRID` called "plot". These are the total and magnetization density.
- non-collinear: `denIn_f.xsf`, `denIn_A1.xsf`, `denIn_A2.xsf`, `denIn_A3.xsf` each with a single `DATAGRID` called "plot". These are the total density and the three components of the magnetization density.
- non-collinear with `polar=T`: `denIn_f.xsf`, `denIn_A1.xsf`, `denIn_A2.xsf`, `denIn_A3.xsf`, `denInAabs.xsf`, `denInAtheta.xsf`, `denInAphi.xsf` each with a single `DATAGRID` called "plot". These are the total density, the three components of the magnetization density and their polar representation.
- non-collinear with `vecField=T`: `denIn_f.xsf`, `denIn_A1.xsf`, `denIn_A2.xsf`, `denIn_A3.xsf` and `denIn_A_vec_plot.xsf`

The descriptors in the filenames refer to the representations as scalar fields $f(\vec{r})$ and $g(\vec{r})$ or the components of a vector field $\vec{A}(\vec{r})$. For the potential, these quantities correspond to the effective scalar potential and the additional exchange-correlation magnetic field in the spin-polarized case.

Using the representations of forces in XCrySDen, plots of the vectorial quantities can also be made. For this, set `vecField=T` in a non-collinear calculation (for collinear spin-polarized calculations please use the above example and view the _f and _g quantities). The vector for each grid point will be written into a new file as an atom of type X at coordinates x, y, z, with the associated "Forces" Ax, Ay and Az. In the file, manually set the first number under `PRIMCOORD` from n up to n plus the total number of gridpoints

specified. Dummy atoms will then be displayed in the XCrySDen visualization and the vectors can be shown by going to `Display` and checking `Forces`. In the `Modify` section, the representation of these vectors can then be controlled. Note, that it is quite hard to neatly visualize the vectors, as the length and thickness needs to be chosen appropriately and the atomic radius of the dummies needs to be set very small. The visualization can easily become cluttered.

The following shows two plots created for a converged calculation on collinear ferromagnetic Fe_2N. They were obtained in XCrySDen by reading in the `denIn_f.xsf` and `denIn_g.xsf` to obtain the total density and magnetization density respectively.

The first plot shows an isosurface for a certain charge density value on top of the structure extended to two unit cells in y- and z-direction.

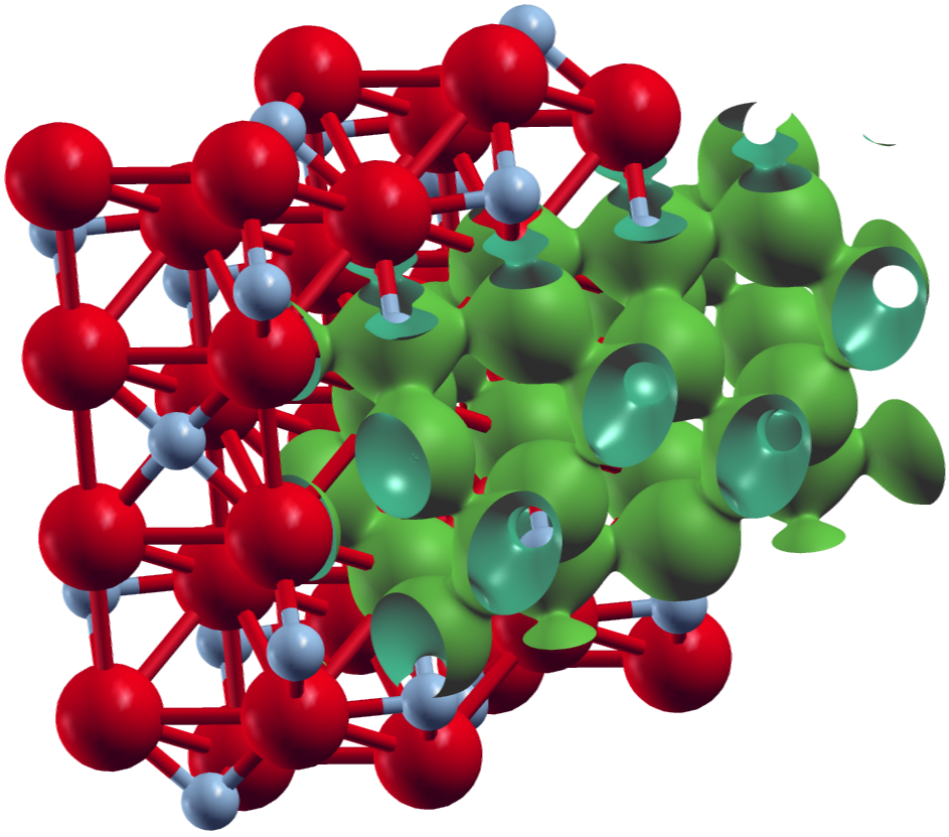

Figure 4.5: XCrySDen plot of an isosurface for the total charge density for Fe_2N.

The second plot shows an isosurface indicating a vanishing magnetization density and a color-coded visualization of the magnetization density on a plane, both on top of the underlying structure.

The following input was used:

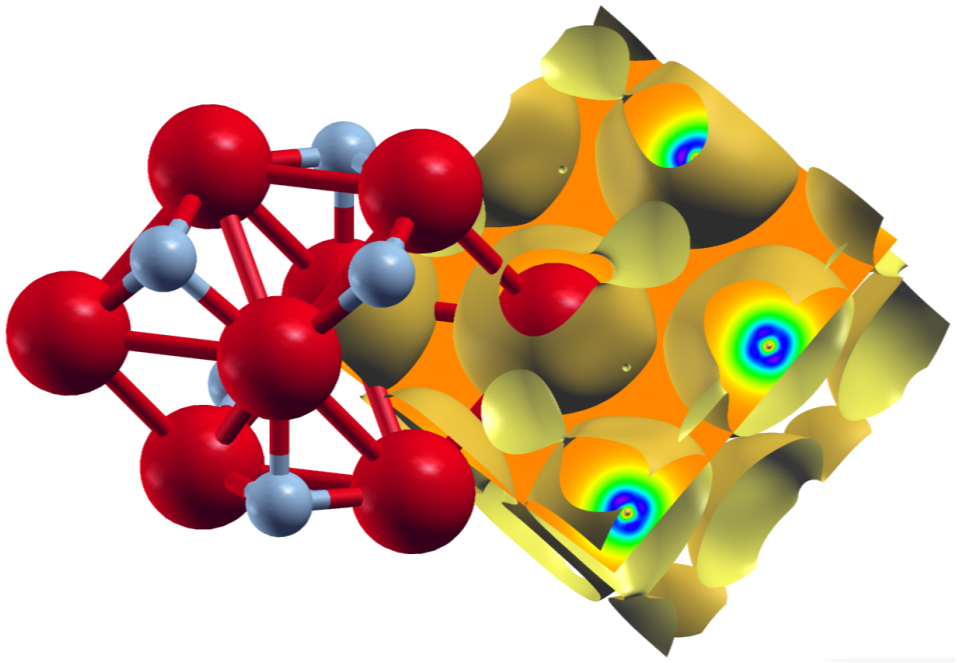

Figure 4.6: XCrySDen plot of the magnetization density for collinear ferromagnetic Fe_2N.

```
<plotting iplot="2" polar="F" format='1'/>
   <plot cartesian="F" TwoD="F" grid="110 138 121" vec1="1.0 0.0 0.0" vec2="0.0 1
</plotting>
```

Note that the grid sizes in each direction differ. This has been done to achieve approximately cubic voxels in the datafile: The underlying unit cell is orthorombic. The offset defined by the `zero` in the tag is clearly visible in the plots above.

It is also possible to only plot the density contribution related to certain states which can be selected by different criteria. In Fleur this feature is called "charge density slicing". For this one first has to converge a self-consistent density and then prepare the plot by creating a charge density file that only contains the density due to the selected states. For the latter part of this procedure one has to set `output/@slice` to `T` and select the states in the `output/chargeDensitySlicing` tag.

It allows to specify the criteria by the following attributes:

- `minEigenval`: The minimal eigenvalue of the states to be selected. Investigate the eigenvalue in the `out.xml` file to select this value.
- `maxEigenval`: The maximal eigenvalue of the states to be selected. Investigate the eigenvalue in the `out.xml` file to select this value.
- `pallst`: Mnemonic for "plot all states", setting this boolean switch to `T` will also include the unoccupied states into the plot.
- `nnne`: This is used to specify a certain eigenstate index. If it is set to `0` the criterion is not applied.

- `numkpt`: This is used to specify a certain **k** point to be considered. If it is set to `0` the criterion is not applied.

Performing a Fleur calculation with such a parametrization will produce the desired charge density file `cdn_slice.hdf` or `cdn_slice` if Fleur is used without HDF5 support. Once this file is available one additionally sets `output/plotting/@iplot` to 2 and procedes with the plotting as before. The correct charge density file for the plotting will be chosen as long as `output/@slice="T"`.

To illustrate this feature the following plot shows the density due to the defect states of an A-nitrogen center defect in the diamond bandgap at the Γ point. A $4 \times 4 \times 4$ supercell was chosen for this calculation and the two nitrogen atoms are located on two opposing corners of the supercell. The plot visualizes an isosurface for some value of the density and an isolines-enhanced color-coding of the density on a plane on one side of the unit cell. Note that the color-plane is not parallel to the nitrogen bond.

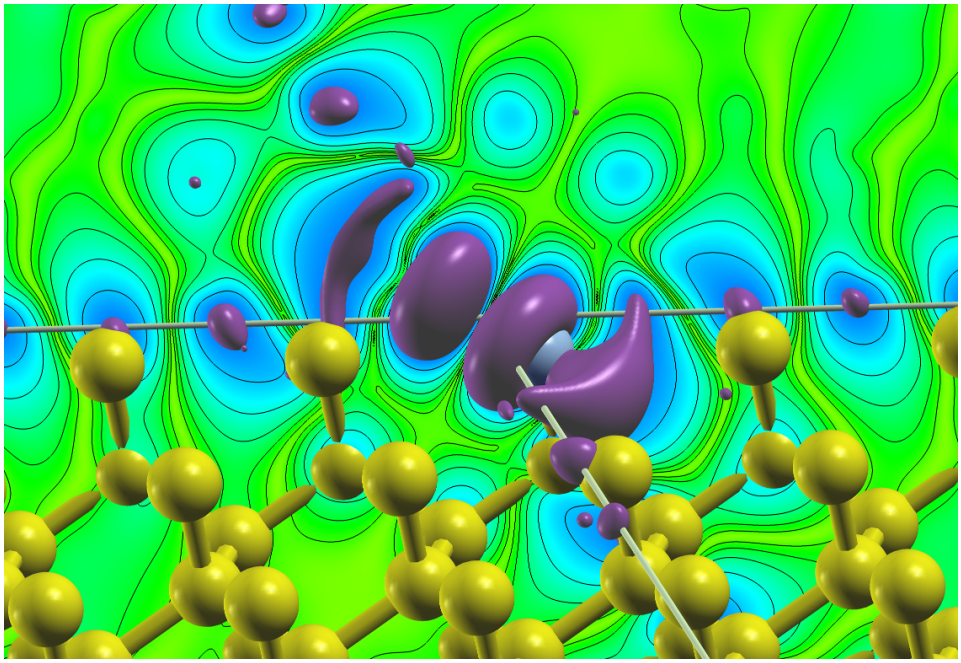

Figure 4.7: Density due to defect states of an A-nitrogen center defect in diamond.

4.8 Band unfolding

Fleur includes the ability to unfold the band structure (see e.g. *O. Rubel, A. Bokhanchuk, S. J. Ahmed and E. Assmann, Unfolding the band structure of discordered solids: from bound states to high-mobility Kane fermions, Physical Review B 90, 115202 (2014)*) of a supercell calculation to the one of a primitive unit cell. This feature is controlled by a seperate flag within the output section. In this section the size of the supercell has to be

specified (in multiples of primitive unit cells). Unfolding the band structure can only be used when the flag for a band structure calculation (`/output/@band`) is set. When the boolean flag `/output/unfoldingBand/@unfoldBand` is set to `T`, two band structures are calculated. The band structure of the provided cell (supercell) and a band structure for the primitive cell, meaning the backfolded band structure, is written out. Fleur then creates additional output files (`band_sc.1`, `band_sc.2` and `band_sc.gp`) with the same usage as described for the normal band structure calculation. The gnuplot file can be used directly to plot the backfolded band structure.

Additionally the information is also written to the `banddos.hdf` file. There it can be used for other plotting tools. This allows for additional analysis, for example the combination of band unfolding and band character weight is then possible to plot.

points for the band structure that is a multiple of the number of mpi threads. This is checked at the beginning of the calculation, if the associated error message is displayed, please adapt your parallelisation or number of **k** points accordingly.

It follows an example for a $4 \times 4 \times 4$ diamond supercell with a single A-nitrogen-center defect. The band unfolding specification in the input file for such a calculation is:

```
<unfoldingBand unfoldBand="T" supercellX="4" supercellY="4" supercellZ="4"
```

In the figure below the supercell band structure together with a visualization of the weights obtained for the backfolded band structure are shown along the **k**-point path for the primitive unit cell. The visualization of the weights makes the backfolded band structure clearly identifiable while the supercell band structure is very faint. In addition to this visualization in the band gap the two defect states due to the A-nitrogen-center are highlighted in yellow.

Additionally the ability to use a rotated supercell is available. This can be useful for surfaces. The full tranformation matrix between the primitive cell and the supercell can be given below the unfolding input (example: 120° rotation of hexagonal lattice). It has to consist of integer values for the unfolding to work.

```
<unfoldingBand unfoldBand="T" supercellX="1" supercellY="1" supercellZ="1">
  <transMat>
    <row-1>-1  2 0</row-1>
    <row-2> 2 -1 0</row-2>
    <row-3> 0  0 1</row-3>
  </transMat>
</unfoldingBand>
```

It is important to note that the k-point path is not adapted automatically to the supercell when using the full transformation matrix. This means the path has to be given in terms of the supercell corrdinates.

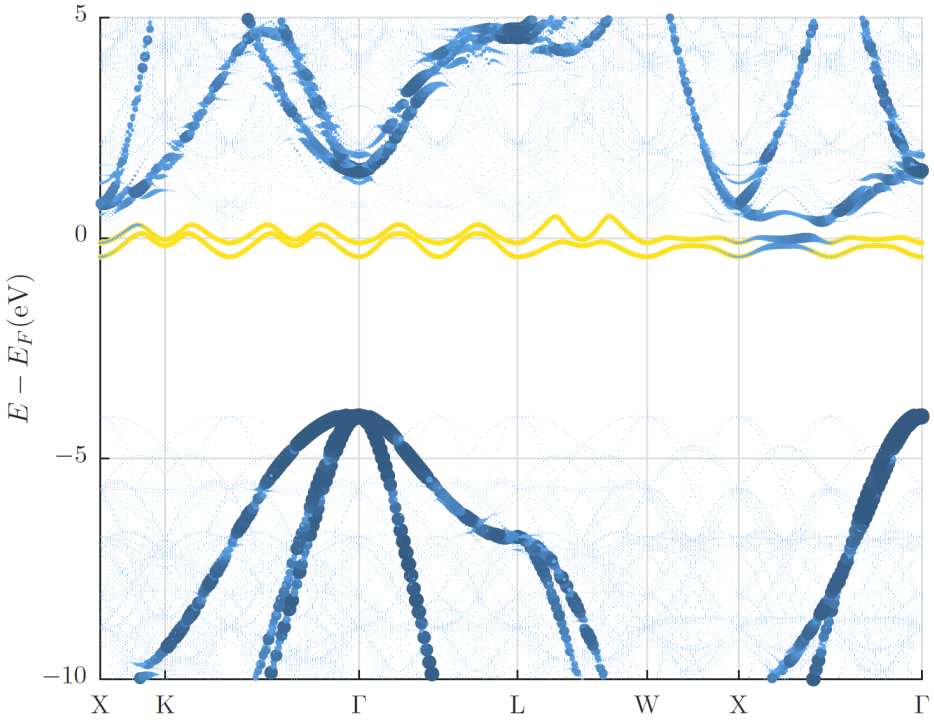

Figure 4.8: Diamond band structure with A-nitrogen-center defect.

In the case of using the entries `supercellX`, `supercellY`, and `supercellZ` it is not necessary to adapt the k-point path to the supercell, rather the path of the primitive cell can be given as an input. If the supercell has the same ratio of basis vectors as the primitive cell the path is found automatically by the input generator.

It is also possible to use the unfolding method for the case of spin-orbit coupling in 2nd variation (`l_soc="T"`). In this case a second variation eigenproblem is solved and the resulting wavefunctions are used. The switch `useOlap` allows to control whether the overlap matrix for the basis functions is used or not. In the case of unfolding for the second variation wavefunctions this switch has to be set to false (`useOlap="F"`).

```
<unfoldingBand unfoldBand="T" supercellX="4" supercellY="4" supercellZ="4" useOlap=
```

The unfolding method is also of high interest for the calculation of film systems and surface states. In this case the unfolding is either performed purely in the z-direction or combined with a transormation matrix to allow the setup of an efficient surface cell. To simplify the setup the flag `&expert primCellZ=6.854 /` in the input generator can be used. This flag has two different functionalities. In the case of a true film system `film=T` the vacuum is set accordingly, so that the unfolding works (an integer multiple of the primitive cell is required). For the case of a pseudo film system (3D periodic system with additional vacuum) the input generator checks if the cell is a multiple of the size of the primitive cell in z-direction. The size of the primitive cell is to be given in atomic units.

4.9 Usage of the magnetic force theorem

Fleur has a special inbuilt calculation mode for evaluating some magnetic properties using the magnetic force theorem. The fundamental idea of these calculations is to use a converged charge density of a reference setup and then to estimate the change of the energy of a modified setup by simply comparing the sum of eigenvalues of the different setups. So instead of performing full self-consistency calculations for all setups one needs to perform a self-consistent density for a single setup only. More detailed information on the theoretical background of the approach can be found for example here.

The general procedure for the usage of the force-theorem mode is:

1. Select a suitable reference system and obtain a converged charge density.
2. Use that charge density and the input-switches described below to evaluate the eigenvalue sum for different setups.

warning Choice of reference setup
Of course the proper choice ofthe reference system is of paramount importance in the application of this method. In general one should apply the force theorem method only for the investigation of small changes. You should investigate the influence of the reference state carefully and make sure you understand the dependence of your results to be able to estimate the applicability of your calculations. It is also important that the configuration of the reference setup is compatible to all subsequent force theorem calculations. For example the selected symmetries should not clash with the parameters set for one of the force theorem setups.

In Fleur we currently have implemented the calculation of the following properties using a force-theorem approach:

- Magnetic anisotropy
- Spin-spiral dispersion
- Dzyaloshinskii Moriya interaction
- Heisenberg exchange interaction

`force-theorem` section in the input-file that has to be added after the `output` section. They are activated after the self-consistency loop.

For example:

```
<forceTheorem>
   <MAE theta="0.0 0.1*Pi" phi="0.0 0.2*Pi" />
</forceTheorem>
```

4.9.1 Magnetic anisotropy energy

The magnetic anisotropy energy of a magnet can be calculated in this mode. Please be aware of the fact that this mode of course will not include any contribution from the shape-anisotropy. Basically, you compare the eigenvalue sums for different directions of the magnetisation. Hence, you specify for the mode a tag like:

```
<MAE theta="0.0 0.1*Pi" phi="0.0 0.2*Pi" />
```

different angles. The number of angles in both must be the same and in the example above, two calculations, one with `theta=0.0`, `phi=0.0` and a second with `theta=0.1*Pi`, `phi=02.Pi` will be performed.

You will find in the out.xml file a summary section listing the resulting eigenvalue sums. For example you will find for the two angle pairs defined above something like:

```
<Forcetheorem_MAE Angles="2">
   <Angle theta="0.0000000" phi="0.0000000" ev-sum="-16.7314156"/>
   <Angle theta="0.3141593" phi="0.0628319" ev-sum="-16.7314090"/>
</Forcetheorem_MAE>
```

The value in `ev-sum` is the eigenvalue sum for the respective calculation.

Please note, that this mode requires a collinear reference state.

4.9.2 Spin-spiral dispersion

The spin-spiral dispersion mode evaluates the eigenvalue sum of a series of spin-spirals of different q-vector. Hence, you have to provide a list of q-vectors as in the example:

```
<spinSpiralDispersion>
   <q> 0.0 0.0 0.0 </q>
   <q> 0.1 0.0 0.0 </q>
   <!--   ...   -->
</spinSpiralDispersion>
```

Please note, that the first q-vector should correspond to the q-value of the reference state.

4.9.3 Dzyaloshinskii Moriya Interaction(DMI)

This mode can be used to estimate the effect of the DMI by calculating the changes of the energy induced in spin-spiral states by the spin-orbit interaction (SOC). The generalized Bloch theorem employed in spin-spiral calculations is fundamentally incompatible with SOC, hence this mode is actually quite different from the other modes discussed here as it employs first order perturbation theory to estimate the effect of spin-orbit coupling on a spin-spiral calculation.

The input can be understood as a combination of the MAE-mode and the Spin-spiral dispersion mode. For example:

```
<DMI theta="0.0 0.5*Pi 0.5*Pi" phi="0.0 0.0 0.5*Pi" >
   <qVectors>
      <q> 0.0 0.0 0.0 </q>
      <q> 0.1 0.0 0.0 </q>
   </qVectors>
</DMI>
```

This would provide results for six setups: two q-vectors for each of the three sets of angles for the SOC.

4.9.4 Heisenberg exchange interaction (Jij)

The mode to calculate data suitable to determine the Heisenberg exchange interactions determines energies for special spin-spiral states. Starting from a collinear reference state these are constructed as follows: for all pairs of magnetic atoms the magnetisation of the atoms is tilded by an angle '`thetaj` and a spin-spiral is superimposed. The corresponding input will specify the q-vectors and the tilting angle.

```
<Jij thetaj="0.1*Pi" >
   <qVectors>
      <q> 0.0 0.0 0.0 </q>
      <q> 0.1 0.0 0.0 </q>
   </qVectors>
</Jij>
```

The number of force-theorem eigenvalue sums to be evaluated in this mode can be quite significant. In general a maximum of $N = M^2_{\text{Number of magnetic atoms}} \cdot Q_{\text{Number of q-vectors}} + 1$ different setups might be calculated in this mode.

4.10 Applying external fields

It is possible to apply electric and magnetic fields to a material investigated by Fleur. This is done by adding a `fields` element in the `calculationSetup` section of the input file.

4.10.1 Applying magnetic fields

For the application of magnetic fields the `fields` element in the `calculationSetup` section looks like:

```
<fields b_field="0.1"/>
```

Specifying a **B** field in this way applies it to the whole unit cell. But it is also possible to only apply a **B** field to a certain atom. For this a `special` element has to be inserted into the respective `atomSpecies` section directly below the `prodBasis` and `energyParameters` entries. Such an element looks like:

```
<special b_field_mt="0.1"/>
```

If you are applying **B** fields in Fleur you should also be aware that:

- The implementation actually only adds simple Zeeman terms to the potential, not proper **B**-fields.
- The fields are not taken into account when calculating the density-potential integrals which enter the total energy..
- Magnetic fields can be applied in film and bulk setups.

4.10.2 Applying electric fields

There are two basic ways of specifying electric fields:

1. The values of the sheets of charge for external electric fields is set by requiring charge neutrality. Thus, most easily you can add or subtract a fractional charge by changing the number of valence electrons. The resulting field is shown in the **external electric field** section of the **out** file.

```
<bzIntegration valenceElectrons="8.00100000" ...
```

Note that adding too much charge to the valence electrons will result in a Fleur error.

2. For film unit cells more complex settings are possible using the `<fields>` tag in the `calculationSetup` section. This is done by placing a sheet of charge on either side of the film in the vacuum. In consequence an electric field perpendicular to the film surface is generated. To conserve charge neutrality of the unit cell the total charge added in this way has to be subtracted from the number of `valenceElectrons` in the film.

   In detail the setup for the application of an electric field in this way looks like:

```
<fields zsigma="0.0" sig_b_1="0.0" sig_b_2="0.0" autocomp="T"
        dirichlet="F" eV="F">
   <shape> shapestring </shape>
</fields>
```

The following attributes are provided:

- `zsigma`: The position of the sheets of charge relative to the vacuum boundary (set by default to 10 a.u. (= 5.291772 Å)).
- `sig_b_1` and `sig_b_2`: for charges on the upper and lower plates (default 0.0). Setting these to different values enables to place an asymmetric field.
- `autocomp`: This switch makes sure that overall charge neutrality is automatically calculated.
- `dirichlet`: This switch enables the use of Dirichlet boundary conditions.
- `eV`: This switch is used to modify units in the `dirichlet="T"` case.
- In addition an (unlimited) number of `<shape>` tags can be given to specify inhomogenous fields.

Inhomogeneous fields can also be generated:

- 'sig_b_1/2' contain the additional (homogeneous) charge for the top and bottom sheet. By default, excess (positive/negative) charge of the film is compensated by equally charging the charge sheets; if 'autocomp' is false, the user has to do this manually. Note: Fleur requires an overall (film plus top plus bottom sheet) charge neutrality.
- The inhomogeneous charge can be placed using the in which strings are supplied. These strings specify the inhomogenous charges using the key-words **rect**, **circ**, **rectSinX**, **polygon**, and **datafile**. Their detailed syntax is:

```
rect <sheet> <x>,<y> <width>,<height> <charge> [options]

circ <sheet> <x>,<y> <radius> <charge> [options]

rectSinX <sheet> <x>,<y> <width>,<height>  <amplitude> <n> <delta> [options]

polygon <sheet> <n_points> <x1>,<x2> ... <x_n>,<y_n> <charge> [options]
```

```
datafile <filename> [zero_avg] [options]
```

Note that all positions and lengths are currently relative values (i.e. between 0 and 1). The sheet to be used can be set using , which can be either **top**, **bot** or **topbot**/**bottop**. Options are: **add** (default) to add the charge to the charge of previous tags or **replace** to use the new charge instead; **zero** to place the charge only to areas which were before zero, **nonzero** to place it at areas which where before nonzero or **all** (default) to place it in the whole area covered by the tag.

Note: The regions can exceed the unit cell plane and then cut off, e.g.

```
circ top 0,0  0.25  0.5
```

places half an electron in a quarter circle with origin (0,0). Note, however, that **circ** creates a perfect circle only on the grid; this generates a circle and not an ellipse only if the ''k1d''/''k2d" ratio matches the crystal's ''a''/''b" ratio.

rectSinX creates a sinodal potential in ''x" direction (constant in ''y" direction for any ''x" value), i.e. $V(x,y) = A\sin(2\pi nx + 2\pi\delta)$, where ''A" is the amplitude; however, the argument in **apwefl** is not ''A" directly but $A' = AL_z$, where $\{L_z\}$ is the number of points in ''z" direction. Contrary to **circ** and **rect**, charges are masked out without being redistributed to non-masked positions. It is $\int_0^{2\pi} A|\sin(x)|\mathrm{d}x = 4A'$, **n** is the order and **delta** (δ) the offset.

polygon creates a polygon-shaped charge distribution; note that the currently used algorithm does not always give the perfectly shaped polygon - and the edge points are not always included in the polygon.

The **datafile** reads the data from a file; if a **zero_avg** has been given, the charge is averaged to zero, i.e. only the inhomogeneous contributions are taken into account. The option **replace**/**add** is supported, but **zero**/**not_zero** is not. The syntax of the data file itself is as follows. First line: number of ''x" and ''y" points; second line: charge for point (x=1,y=2), third: (x=1,y=2) etc. The number of points must be ***3k1d and 3*k2d**.

Example 1: To have two top plates (segments):

```
rect top 0,   0  0.5,1.0  0.2
rect top 0.5,0  0.5,1.0 -0.2
```

Example 2: To have a charged ring with 0.2e and -0.2e of charge evenly distributed outside this ring:

```
circ topBot 0.5,0.5 0.2  1           ! Create temporarily an inner ring
circ topBot 0.5,0.5 0.3  0.2 zero    ! Create outer ring
circ topBot 0.5,0.5 0.2  0   replace ! Delete inner ring
rect topBot 0,0     1,1 -0.2 zero    ! Place smeared opposite charge
```

4.11 Core spectrum calculations for EELS

In the `output` section of the Fleur input file it can be specified that a core spectrum has to be calculated. For this `output/@coreSpec` has to be set to `T` and in the `output` section the optional element `coreSpectrum` has to be defined in analogy to the following example:

```
<coreSpectrum eKin="300.0" atomType="1" lmax="2" edgeType="L"
              eMin="-1.0" eMax="15.0" numPoints="17" nqphi="10"
              nqr="10" alpha_Ex="0.024" beta_Ex="0.050"
              I_initial="155" verbose="T">
   <edgeIndices> 3 </edgeIndices>
</coreSpectrum>
```

Attribute	Description
eKin	The kinetic energy of the incoming electron in units of keV. The relativistic correction term, which was shown to be important (see PhD thesis of Kevin Jorissen), is proportional to eKin.
atomType	This is the index of the atom group for which the EELS is to be calculated.
lmax	Maximum value of the angular momentum to be considered in the EELS calculation. By setting it to a reasonably low value (e.g. 2 or 3), the calculation is significantly faster in comparison with no lmax restriction.
edgeType	The edge selection: K, L, M, ...
eMin	The bottom edge of the energy interval with respect to the Fermi energy for which the EELS is calculated.
eMax	The top edge of the energy interval with respect to the Fermi energy for which the EELS is calculated.
numPoints	Number of points in the [eMin,eMax] interval for which the EELS is calculated..
nqphi	Number of angular divisions of the q-mesh, optional, standard value: 10
nqr	Number of radial divisions of the q-mesh, optional, standard value: 10
alpha_Ex	Experimental convergence semi-angle in rad, optional, standard value: 0.024
beta_Ex	Experimental collection semi-angle in rad, optional, standard value: 0.05

Attribute	Description
I_initial	Initial intensity of the incoming electron beam, optional, standard value: 155
verbose	Verbose output is produced, optional, standard value: F

In the `edgeIndices` element a list of space separated indices has to be provided.

4.12 Employing Wannier functions

1. eig66: Keep the gauge consistent by storing eig.hdf
2. Formatted vs. unformatted IO of files such as WF1.mmn, WF1.uHu, WF1.mmn0,...
3. MLWFs in nonmagnetic systems without SOC
4. MLWFs in magnetic systems without SOC
5. MLWFs with SOC and MLWFs for noncollinear magnets
6. Tutorial Examples
7. Wannier-related input and output files

5 Expert knowledge and Troubleshooting

Contents

Users of a DFT code, and especially users of an FLAPW code like Fleur, experience a steep learning curve when they start to use such a program. To simplify this learning in the following sections experience from expert users will be directly provided.

5.1 Choosing good parallelization schemes

The efficient usage of Fleur on modern (super)computers is ensured by a hybrid MPI/OpenMP parallelization. The **k**-point loop and the eigenvector problem are parallelized via MPI (Message Passing Interface). In addition to that, every MPI process can be executed on several computer cores with shared memory, using either the OpenMP (Open Multi-Processing) interface or multi-threaded libraries.

In the following the different parallelization schemes are discussed in detail and the resulting parallelization scaling is sketched for several example systems.

5.1.1 MPI parallelization

- The **k**-point parallelization gives us increased speed when making calculations with large **k**-point sets.
- The eigenvector parallelization gives us an additional speed up but also allows to tackle larger systems by reducing the amount of memory each MPI process uses.

Depending on the specific architecture, one or the other or both levels of MPI parallelization can be used.

5.1.1.1 k-point parallelization

This type of parallelization is always chosen if the number of **k**-points (K) is a multiple of the number of MPI processes (P). If K/P is not an integer, a mixed parallelization will be attempted and M MPI processes will work on a single k-point, so that $K \cdot M/P$ is an integer. This type of parallelization can be very efficient because the three most time-consuming parts of the code (Hamiltonian matrix setup, diagonalization and generation of the new charge density) of different **k** points are independent of each other and there is no need to communicate during the calculation. Therefore this type of parallelization is very beneficial, even if the communication between the nodes/processors is slow. The drawback of this type of parallelization is that the whole matrix must fit in the memory

available for one MPI process, i.e., on each MPI process sufficient memory to solve a single eigenvalue-problem for a single **k** point is required. The scaling is good as long as many **k** points are calculated and the potential generation does not become a bottleneck. The superideal scaling in the following figure is caused by caching effects.

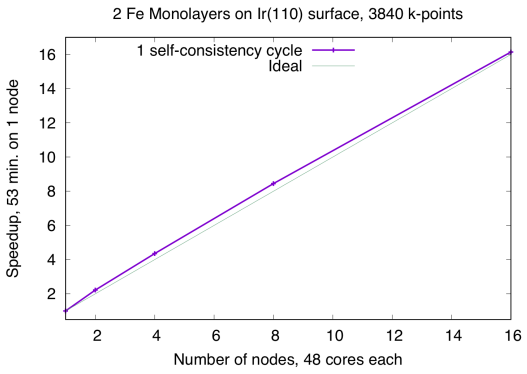

Figure 5.1: Typical speedup of the k-point parallelization. Hardware: Intel Skylake, 2.1 GHz (CLAIX 2018).

5.1.1.2 Eigenvector Parallelization

If the number of **k** points is not a multiple of the number of MPI processes, every **k** point will be parallelized over several MPI processes. It might be necessary to use this type of parallelization to reduce the memory usage per MPI process, i.e. if the eigenvalue-problem is too large. This type of parallelization depends on external libraries which can solve eigenproblems on parallel architectures. The Fleur code contains interfaces to ScaLAPACK, ELPA and Elemental. Furthermore, for a reduction of the memory footprint it is also possible to use the HDF5 library to store eigenvector data for each **k** point on the disc. However, this implies a performance drawback.

5.1.2 OpenMP parallelization

Modern HPC systems are usually cluster systems, i.e., they consist of shared-memory computer nodes connected through a communication network. It is possible to use the distributed-memory paradigm (MPI parallelization) also inside the node, but in this case the memory available for every MPI process will be considerably smaller. Imagine to use a node with 24 cores and 120 GB memory. Starting a single MPI process will make all 120 GB available to this process, two will only get 60 GB each and so on, if 24 MPI processes are used, only 5 GB of memory will be available for each of them. The intra-node parallelism can be utilized more efficiently when using the shared-memory programming

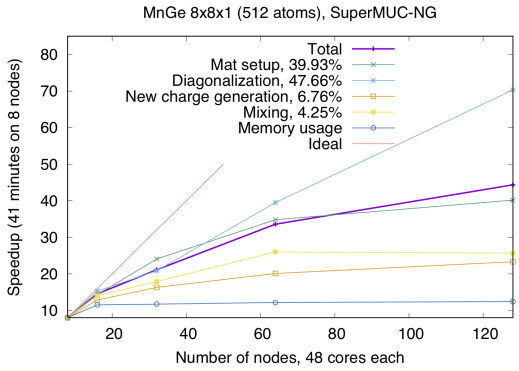

Figure 5.2: Example of an EV (eigenvector) parallelization. Calculation were performed on the SuperMUC-NG (Intel Skylake). The test case: MnGe supercell (512 atoms) with non-collinear magnetic structure, spin-orbit interactions included.

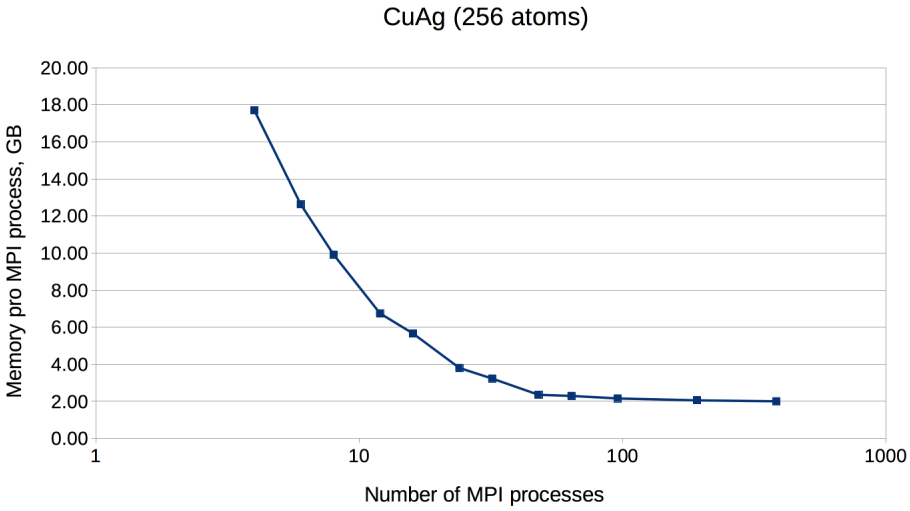

Figure 5.3: An example of FLEUR memory requirements depending on the amount of MPI ranks. Test system: CuAg (256 atoms, 1 **k** point). Memory usage was measured on the CLAIX supercomputer (Intel E5-2650V4, 2.2 GHz, 128 GB per node).

paradigm, for example through the OpenMP interface. In the Fleur code the hybrid MPI/OpenMP parallelization is realised by directly implementing OpenMP pragmas and the usage of multi-threaded libraries. Note that in the examples above OpenMP parallelization was used together with the MPI parallelization. To strongly benefit from this type of parallelization, it is crucial that Fleur is linked to efficient multithreaded software libraries. A good choice is the ELPA library and the multithreaded MKL library. The following figure shows the pure OpenMP parallelization scaling on a single compute node.

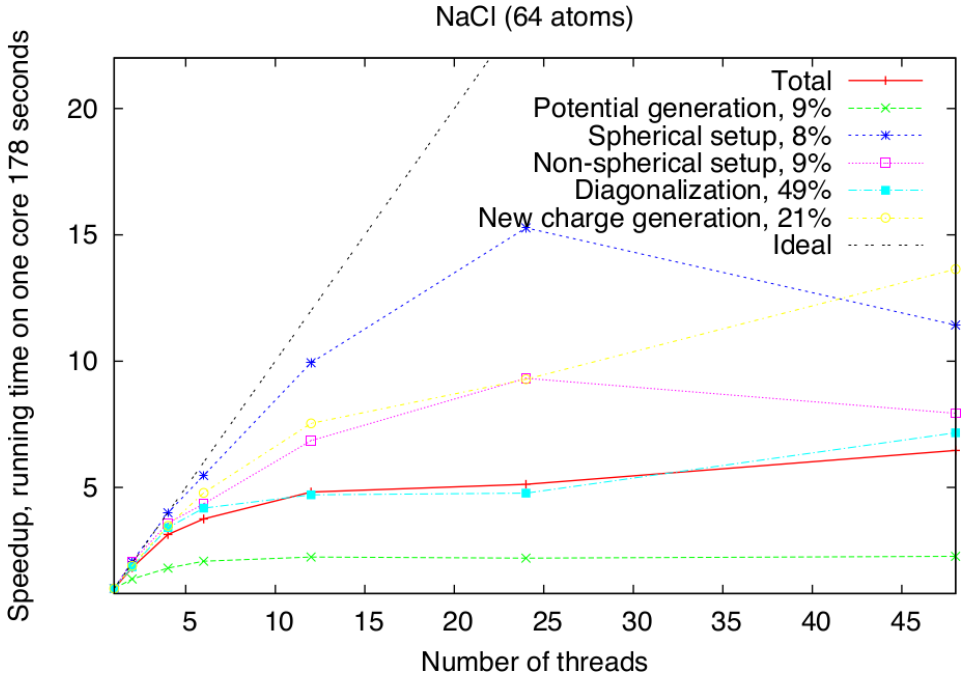

Figure 5.4: Pure OpenMP parallelization scaling for NaCl (64 atoms) on a single node (Intel Skylake Platinum 8160, 2x24 cores).

5.1.3 Parallel execution: best practices

Since there are several levels of parallelization available in Fleur: **k**-point MPI parallelization, eigenvalue MPI parallelization, and multi-threaded parallelization, it is not always an easy decision how to use the available HPC resources in the most effective way: How many nodes are needed, how many MPI processes per node, how many threads per MPI process. A good choice for the parallelization is based on several considerations.

First of all, you need to estimate a minimum amount of nodes. This depends strongly on the size of the unit cell and the memory size of the node. In the table below some numbers for a commodity Intel cluster with 120 GB and 24 cores per node can be found - for comparable unit cells (and hardware) these numbers can be a good starting point for choosing a good parallelization. The two numbers in the "#nodes" column show the

range from the "minimum needed" to the "still reasonable" choice. Note that these test calculations only use a single **k** point. If a simulation crashes with a run-out-of-memory-message, one should try to double the requested resources. The recommended number of MPI processes per node can be found in the next column. If you increasing #k-points or creating a super-cell and you have a simulation which for sure works, you can use that information. For example, if you double #k-points, just double #nodes. If you are making a supercell which contains N times more atomes, than your matrix will be N-square times bigger and require about N-square times more memory.

Best values for some test cases. Hardware: Intel Broadwell, 24 cores per node, 120 GB memory.

Name	# k-points	real/complex	# atoms	Matrix size	LOs	# nodes	# MPI per node
NaCl	1	c	64	6217	-	1	4
AuAg	1	c	108	15468	-	1	4
CuAg	1	c	256	23724	-	1 - 8	4
Si	1	r	512	55632	-	2 - 16	4
GaAs	1	c	512	60391	+	8 - 32	2
TiO2	1	c	1078	101858	+	16 - 128	2

The next question, how many MPI processes? The whole amount of MPI parocesses is #MPI per node times #nodes. If the calculation involves several **k** points, the number of MPI processes should be chosen accordingly. If the number of **k** points (K) is a multiple of the number of MPI processes (P) then every MPI procces will work on a given **k** point alone. If K/P is not an integer, a mixed parallelization will be attempted and M MPI processes will work on a single **k** point, so that $K \cdot M/P$ is an integer. This means for example that if the calculation uses 48 **k** points, it is not a good idea to start 47 MPI processes.

As for the number of OpenMP threads, on the Intel architectures it is usually a good idea to fill the node with threads (i.e. if the node consist of 24 cores and 4 MPI processes are used, each MPI process should spawn 6 threads), but not to use the hyper-threading.

5.1.4 Example: Choosing the right parallelization on a single node

On a single Node using as much k-point parallelization is usually the most efficient parallelization. To get this you need to get the number of k-points in your calculation from your inp.xml or kpts.xml. Here we have a calculation with 20 k-points:

```
<kPointList name="default" count="20" type="tria-bulk">
```

We also need to know how many cores are on our node. In this example we will assume 12 cores. Then the number of MPIs is given by the greatest common denominator of 20 and 12:

```
n_mpi = gcd(12,20) = 4
n_openmp = n_cores / n_mpi = 12/4 = 3
```

Therefore we can start our calculation with:

```
export OMP_NUM_THREADS=3
mpirun -np 4 <insert your fleur binary here>
```

5.1.5 Further reading

- Parallelization and performance optimization: *U. Alekseeva, G. Michalicek, D. Wortmann, and S. Blügel, Hybrid Parallelization and Performance Optimization of the FLEUR Code: New Possibilities for All-Electron Density Functional Theory. In: Aldinucci M., Padovani L., Torquati M. (eds) Euro-Par 2018: Parallel Processing. Euro-Par 2018. Lecture Notes in Computer Science, vol 11014. Springer, Cham.*
- Parallelization of Hybrid functionals calculations with Fleur: *M. Redies, G. Michalicek, J. Bouaziz, C., M. S. Müller, S. Blügel,and D. Wortmann, Fast All-Electron Hybrid Functionals and Their Application to Rare-Earth Iron Garnets, Front. Mater., 21 March 2022*

5.2 Ghost bands

The separation of core electrons from valence electrons in the FLAPW method relies on the orthogonality of the LAPW basis to the core electron states. This is only given if the core electron states vanish in value and slope at the MT sphere boundary. In reality this is only approximately fulfilled and if the system under investigation strongly deviates from this assumption one obtains (bad) representations of core electron states with the LAPW basis. These representations are bad since the energy parameters are typically far from the core electron eigenenergies. They are called ghost bands. An example for such a ghost band can be seen in figure 1.

The example shows the band structure of bcc Vanadium. In the calculation all electrons up to the $3p$ states were treated as core electrons, all other electrons as valence electrons. Between 10 and 15 eV above the Fermi energy a strange band with nearly no dispersion can be observed. Since the energy of this band shows sudden variations with the k point it is clear that the respective representation is rather bad. These sudden variations are due to small differences in the basis set size for the different k points. The effect of these

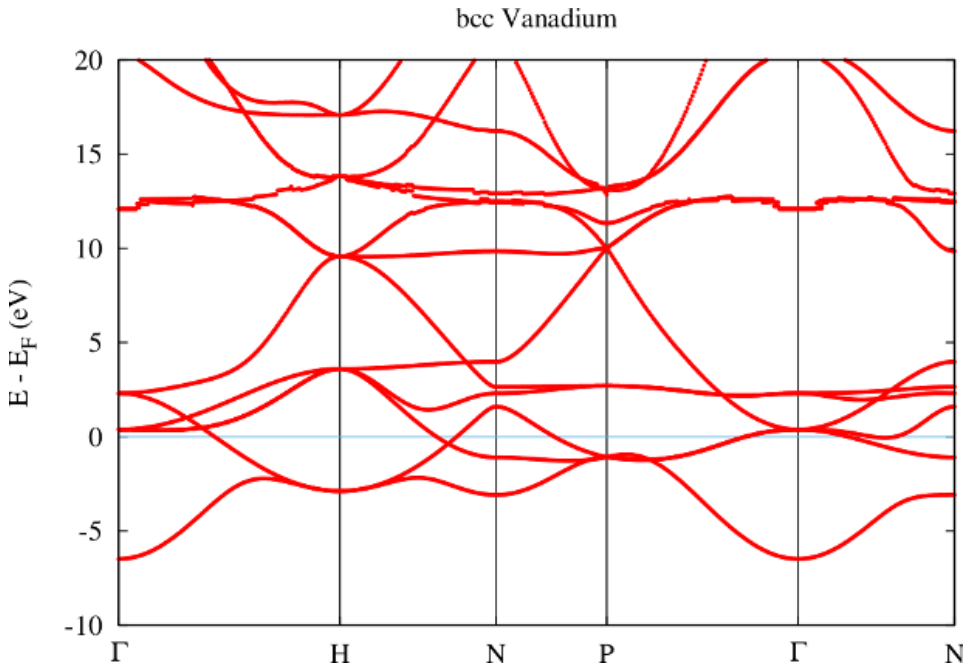

Figure 5.5: Ghost band in Vanadium. $R_{\mathrm{MT}} = 2.20\ a_0$, core electrons reaching out of MT: 0.086

variations can only be so strong if the representation of the band leaves a lot of room for improvement. The vanishing dispersion of the band is a typical property of ghost bands.

Figure 2 shows the effect of reducing the MT radius in such a case. The obvious effect of such a change is the increase of the number of core electrons reaching out of the MT sphere. This implies that the orthogonality assumption is even less fulfilled. It also implies that the linearization error in the MT sphere becomes smaller since the part of the basis functions that is constructed in with the energy parameters fills a smaller fraction of space.

In the band structure the reduction of the MT radius shows up in a lowering of the energy of the ghost band to about 6 eV above the Fermi energy.

The effects of a further reduction of the MT radius are shown in figure 3. Now the ghost band is only about 4 eV above the Fermi energy.

Reducing the MT radius even further results in a crash of the calculation. This is because the ghost band starts to become occupied in the construction of the charge density and therefore contributes to the potential. Since the respective state is already occupied in terms of core electrons this leads to a double-occupation resulting in a messed up potential. The potential then can not be used any more to determine meaningful the energy parameters and/or core electron states. Fleur detects this and therefore stops the calculation. In this example the error message indicates that the Dirac equation can not

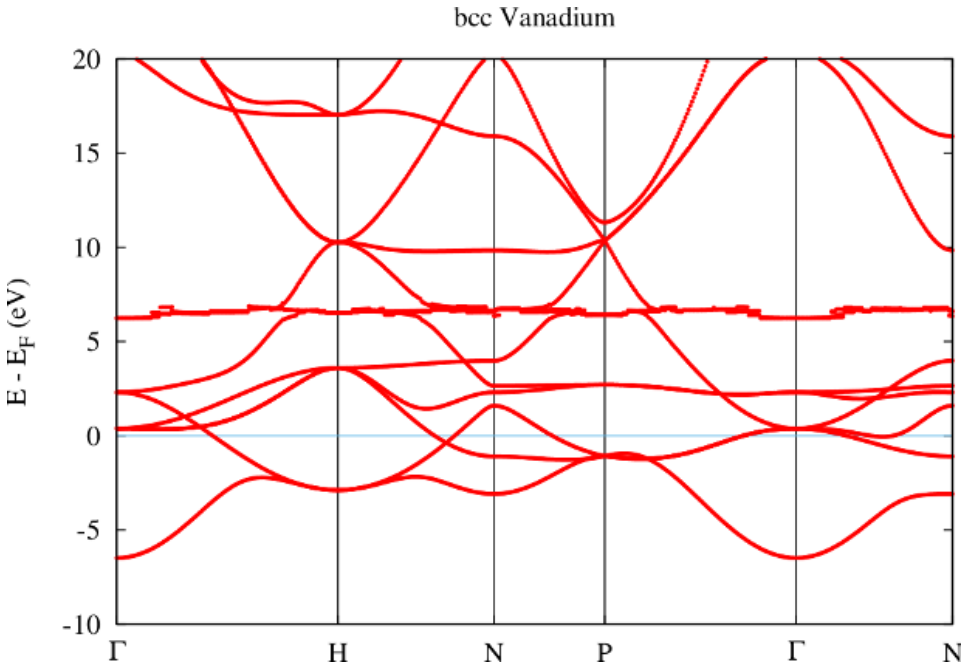

Figure 5.6: Ghost band in Vanadium. $R_{\mathrm{MT}} = 2.17\ a_0$, core electrons reaching out of MT: 0.109

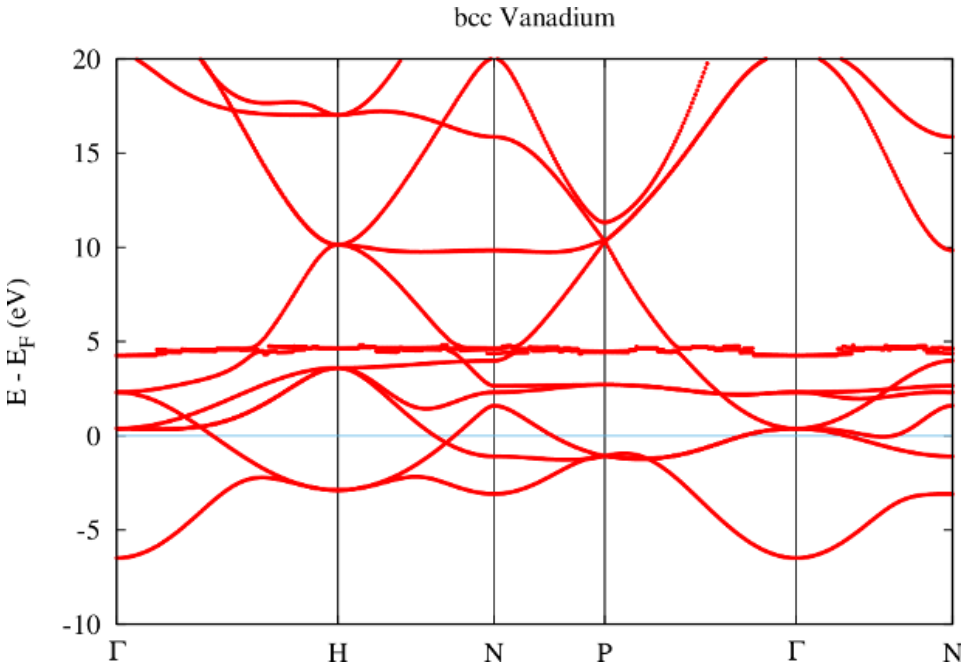

Figure 5.7: Ghost band in Vanadium. $R_{\mathrm{MT}} = 2.16\ a_0$, core electrons reaching out of MT: 0.112

be solved any more, but other error messages are also possible and this error can also have other causes. It may also be that no error message appears and only the result is wrong.

The number of core electrons reaching out of the MT sphere is a good indicator to judge whether there may be problems due to ghost states in the calculation. For SCF calculations numbers below 1% are typically safe territory, for numbers above 4% the validity of the calculation should strongly be investigated in detail.

can be extracted from the out.xml file by invoking `grep lostElectrons out.xml`.

Note that certain calculations may need a high-quality representation of the unoccupied states, too. In such a case the orthogonality requirements may have to be fulfilled to an even higher degree.

The common approach to deal with semicore states leading to ghost bands is to remove them from the core electrons and treat them as valence electrons. The first step in this procedure is the identification of the responsible states. A good indicator for this are the energy eigenvalues of the core electron states. Those states featuring the highest eigenenergies most probably reach farthest out of the MT sphere.

core electrons for each atom type are provided within the respective `coreStates` element. Also the number of electrons lost from the core are provided here. An example for this output for the discussed Vanadium is shown below. The $3p1/2$ and $3p3/2$ core electron states feature the highest eigenenergies.

```
<coreStates atomType="     1" atomicNumber=" 23" spin="1"
            kinEnergy="    941.0636778793" eigValSum="   -550.9805159504"
            lostElectrons=" 0.111893">
   <state n="1" l="0" j="0.5" energy="-195.9388108744" weight="2.000"/>
   <state n="2" l="0" j="0.5" energy="-21.4169509005" weight="2.000"/>
   <state n="2" l="1" j="0.5" energy="-17.9790788510" weight="2.000"/>
   <state n="2" l="1" j="1.5" energy="-17.7189379876" weight="4.000"/>
   <state n="3" l="0" j="0.5" energy="-1.8913495403" weight="2.000"/>
   <state n="3" l="1" j="0.5" energy="-0.9625902671" weight="2.000"/>
   <state n="3" l="1" j="1.5" energy="-0.9318007834" weight="4.000"/>
</coreStates>
```

For the representation of the semicore electron as a valence electron also the representation of the respective state by the valence basis has to be improved. This is done by adding a local orbital (LO) for the representation of this state to the LAPW basis. The outcome of the treatment of the $3p$ electrons as valence electrons with an adapted LO is shown in figure 4.

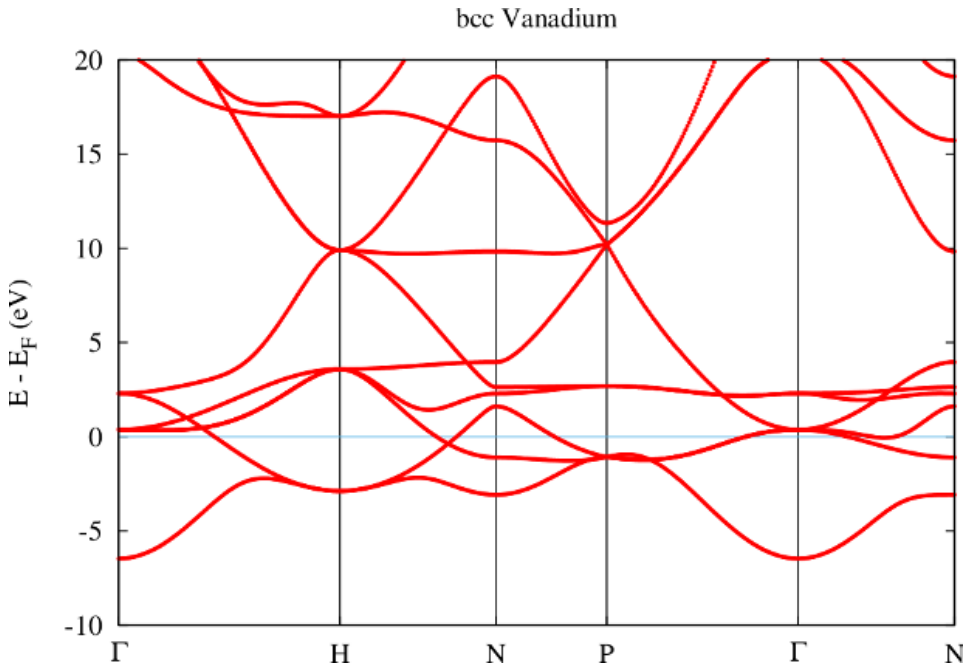

Figure 5.8: No more ghost band in Vanadium. $R_{\mathrm{MT}} = 2.16\ a_0$, core electrons reaching out of MT: 0.012, semicore LO for $3p$ state

From the band structure obtained with the semicore LO it is directly visible that the ghost band is gone. With the LO the representation of the semicore state is good enough to put it energetically to the place where it belongs: Far below the valence band.

If the extension of the basis with LOs is not an option one may also change the MT radii such that less electrons reach out. However, this is cumbersome since the effect is small and one may end up with overlapping MT spheres, which is not allowed.

5.3 Parameter convergence

To obtain high-quality DFT calculations, their parametrization carefully has to be converged. In this section the dependence of Fleur calculations on the most-relevant parameters is discussed. Note that the convergence studies focus on the total energy. This may not be the quantity of relevance for the calculations a user has in mind. Due to fortunate error cancellations total energy differences converge much faster than total energies themselves. Thus, even if the default parametrization mentioned in the following discussions may seem to lead to underconverged results, for most quantities this is not of much relevance. However, it implies that the user has to ensure that the calculations he wants to compare feature analogous parametrizations. Otherwise differences may be dominated by unintended parameter changes.

Note that there is no optimal parameter set that works for all materials. The demands on the parameters depend on the actual physics in the material and are therefore material-dependent.

5.3.1 $K_{\max}$ convergence

The example discussed in this section is an fcc Cu Crystal with a lattice constant of 6.82 a_0. The default parametrization generated by the input generator involved a MT radius of $R_{\mathrm{MT}} = 2.35\ a_0$, a reciprocal plane wave cutoff for the LAPW basis of $K_{\max} = 3.4\ a_0^{-1}$, and and angular momentum cutoff of $l_{\max} = 8$. For such a setup the black curve in the following figure shows the dependence of the calculated total energy on an increasing $K_{\max}$ cutoff. Note that $G_{\max}$ and $G_{\max}^{\mathrm{XC}}$ have manually been set to large enough values for all calculations. All energy values are provided relative to an arbitrary energy cutoff E_{offset}.

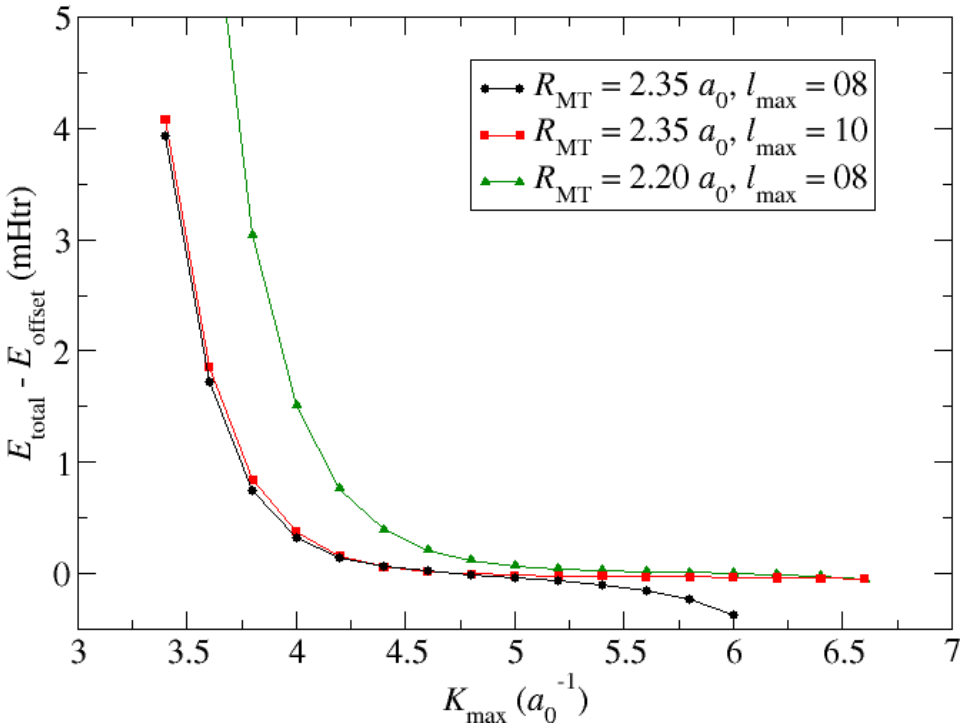

Figure 5.9: $K_{\max}$ convergence for Cu

It can be seen that increasing the $K_{\max}$ cutoff to values up to about $K_{\max} = 4.8\ a_0^{-1}$ leads to less and less changes in the total energy. Considering the fine energy scale one can assume that the total energy is converged for almost all relevant applications. However, beyond $K_{\max} = 4.8\ a_0^{-1}$ the total energy shows an accelerating trend to ever lower values. Results for $K_{\max}6.0\ a_0^{-1}$ are not provided since the respective calculations crash.

What is observed here is the property of the LAPW basis to be nearly linearly dependent. Plane waves over the whole unit cell form, of course, an orthogonal basis set. But in the LAPW basis the plane waves only cover the interstitial region and in the MT spheres they are replaced up to some $l_{\max}$ by numerical radial functions. Beyond the $l_{\max}$ cutoff there is no replacement for the plane waves in the MT sphere. Considering the Rayleigh expansion of plane waves into spherical Harmonics on a sphere it is obvious that for larger $K_{\max}$ cutoffs larger l components become more and more important. Neglecting them in FLAPW yields the convergence behavior discussed above.

Fortunately there are obvious approaches to overcome such problems. One could either increase the $l_{\max}$ cutoff as is shown by the red curve in the figure or reduce the MT radius as it has been done for the green curve.

Increasing the $l_{\max}$ helps directly. The total energy becomes stable up to values of $K_{\max}$ far beyond $K_{\max} = 6.0\ a_0^{-1}$. However, since the numerical radial function is not the same as a spherical Bessel function that would actually perfectly replace the plane wave in the MT sphere, this approach might not always help. If it doesn't help one can still reduce the MT sphere radius. The respective curve also shows an increased stability up to very large $K_{\max}$ values. But it is also obvious that a reduction of the MT radius implies the requirement to use larger $K_{\max}$ parameters to obtain converged results. The product $K_{\max} \cdot R_{\mathrm{MT}}$ may actually be seen as a better convergence measure than $K_{\max}$ alone.

Note that calculations performed with different MT radii might not yield the same results, even if they are carefully converged with respect to all other parameters. This is because of the relativistic treatment in the MT spheres which is not performed in the interstitial region. Also core electron states reaching out of the MT sphere may lead to a dependence of results on the MT sphere radius. Finally some output quantities are only evaluated in the spheres.

In the evaluation of the balance between these parameters one should also keep in mind that the number of LAPW basis functions N_{bas} is proportional to $K_{\max}^3$ and the runtime of an SCF iteration is proportional to N_{bas}^3. This implies that the runtime is actually proportional to $K_{\max}^9$. For complex unit cells it is therefore advisable to choose a setup featuring already good convergence for small $K_{\max}$ parameters.

5.3.2 l_{max} and l_{nonsph} convergence

In contrast to increasing $K_{\max}$ increasing $l_{\max}$ is not problematic and will only yield more precise total energies. No calculation crashes or convergence problems due to too high $l_{\max}$ parameters are known. However, the user should be aware that the $l_{\max}$ requirements depend on the chosen $K_{\max}$ parameter. A common way to choose $l_{\max}$ is by using the rule of thumb

$$l_{\max}^{\alpha} \approx K_{\max} \cdot R_{\mathrm{MT}}^{\alpha}.$$

Increasing l_{nonsph} is also not problematic and will only yield more precise total energies. The difference to l_{max} is that the requirements on l_{nonsph} are related to the potential in the respective MT sphere, while the l_{max} requirements are due to the need to evaluate the kinetic energy of the Kohn-Sham orbitals with high precision and to avoid a linear dependence of the LAPW basis. This allows to choose smaller l_{nonsph} than l_{max}. Typically they are chosen according to the rule

$$l^{\alpha}_{\text{nonsph}} = \min(8, l^{\alpha}_{\text{max}} - 2). \tag{5.1}$$

Both, l_{max} and l_{nonsph} have to be systematically converged for every material under investigation. For most demands the default parameters from the input file generator are already good choices. Note that increasing these parameters is computationally expensive.

5.3.3 The linearization error

Within the MT spheres the LAPW basis features a linearized representation of the Kohn-Sham orbitals around the energy parameters defined in the input file. Of course, this implies that for energies different from the energy parameters there exists a linearization error. It depends quadratically on the energy difference between the energy parameters and the Kohn-Sham states.

For many structures the linearization error for the representation of the valence states is tiny and neglectable, but there are also setups, calculations, and quantities for which care has to be taken to overcome the linearization error and obtain high-quality results. Different strategies exist to eliminate the error. One approach is to shrink the MT sphere in which the error occurs. Another approach is to extend the basis in the respective MT sphere by local orbitals tailored to reduce the error.

The following figure shows the band structure for an fcc Ne crystal obtained with the conventional LAPW basis and with an LAPW basis extended by local orbitals for higher energies in the range of the unoccupied bands (HELOs). The basic setup for these calculations involve the determination of energy parameters for the $2s$, $2p$, $3d$, and $4f$ states and a MT radius of 2.8 a_0. For the HELO extended calculation a set of HELO type local orbitals for the $3s$, $3p$, $4d$, and $5f$ states were added.

It is directly visible that on the energy scale measurable in the plot the representation of the valence states by the conventional LAPW basis is already excellent: The bands calculated with the two setups lie on top of each other. But the LAPW basis functions are only constructed for the valence states. For the unoccupied bands in this example the representation is bad. Adding local orbitals to the basis overcomes this problem and a precise DFT bandgap can be extracted from the figure. Note, however, that Ne belongs to the most extreme examples for the effect of the linearization error on the predicted

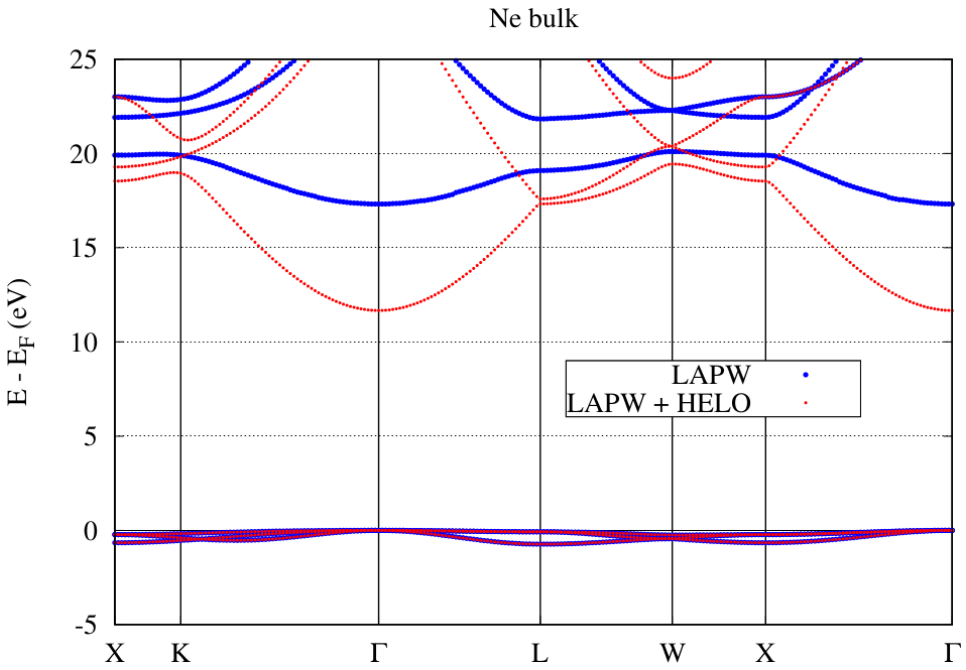

Figure 5.10: Linearization error for the Ne band structure.

bandgap. To overcome the error it is also possible to strongly reduce the MT sphere radius, but this is computationally more expensive.

The demonstration of the effect of the linearization error on the unoccupied states may seem to have limited relevance, but certain calculations also rely on precise representations of these states. For example, GW calculations on top of DFT calculations require unoccupied states up to very high energies above the Fermi level as part of their input. For such calculations it may even be required to add multiple sets of HELOs to cover the whole energy range of relevance. For an analysis of the relevance of the linearization error on GW calculations see *C. Friedrich et al., Band convergence and linearization error correction of all-electron GW calculations: The extreme case of zinc oxide, Phys. Rev. B 83, 081101(R) (2011).*

There are also cases for which the linearization error for the representation of the valence states is relevant. The consequence of this error may be a dependence of calculated quantities on the MT radius. The figure below sketches such an example. It displays the dependence of the calculated equilibrium lattice constant of KCl on the MT radius. The MT radii for both atom species were chosen to be identical.

The figure compares for this dependence the behavior for the conventional LAPW basis with that of an LAPW basis extended by a set of local orbitals (s, p, d, f states) constructed with the second energy derivative of the solution to the radial Schrödinger equation (HDLOs). It can be seen that the extended basis yields results that are stable against variations of the MT radius. This is the wanted behavior. Note that the lattice constant of KCl is especially sensitive to the linearization error due to the large MT

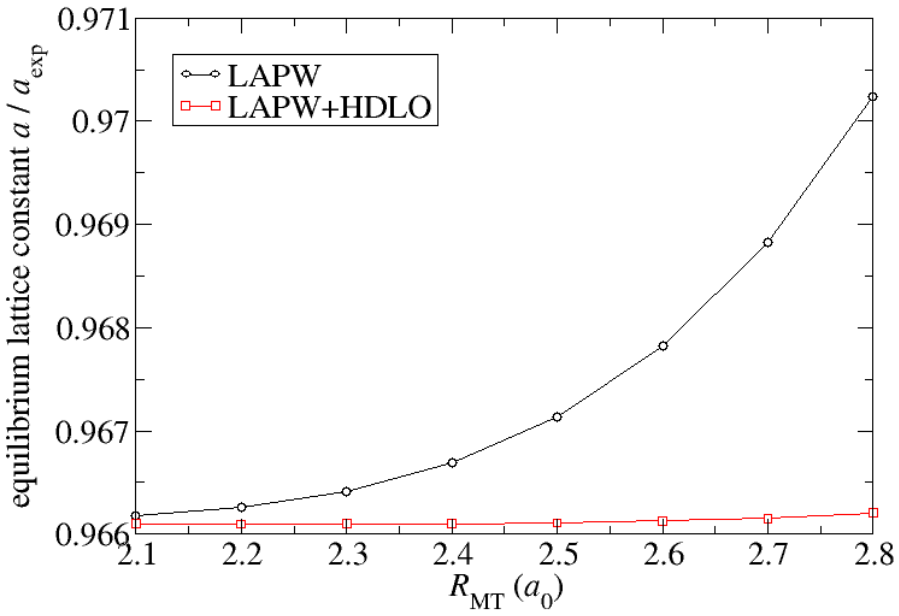

Figure 5.11: MT radius dependence due to the linearization error for the KCl lattice constant.

spheres and its small bulk modulus. A more detailed analysis of the linearization error is performed in *G. Michalicek et al., Elimination of the linearization error and improved basis-set convergence within the FLAPW method, CPC 184, 2670 (2013).*

5.3.4 k-point set convergence

An aspect of the parametrization that is not limited to the FLAPW method is the choice of the **k**-point set used to sample the Brillouin zone in the construction of the charge density. For the Cu example discussed above the Fleur input generator yields an $8 \times 8 \times 8$ **k**-point mesh. The following figure shows how the results for the total energy change when finer **k**-point meshes are selected

For this example the changes in the total energy are very small and for most purposes the default **k**-point set is sufficient. For other unit cells and for certain target quantities this may be different and the **k**-point mesh has to be chosen much finer. But even here it is important to choose analogous **k**-point sets if different structures have to be compared, i.e., the **k**-point density should be as similar as possible.

The plot also demonstrates an even-odd behavior: The convergence behavior differs between even and odd **k**-mesh sizes N. Such a behavior can also be observed in some

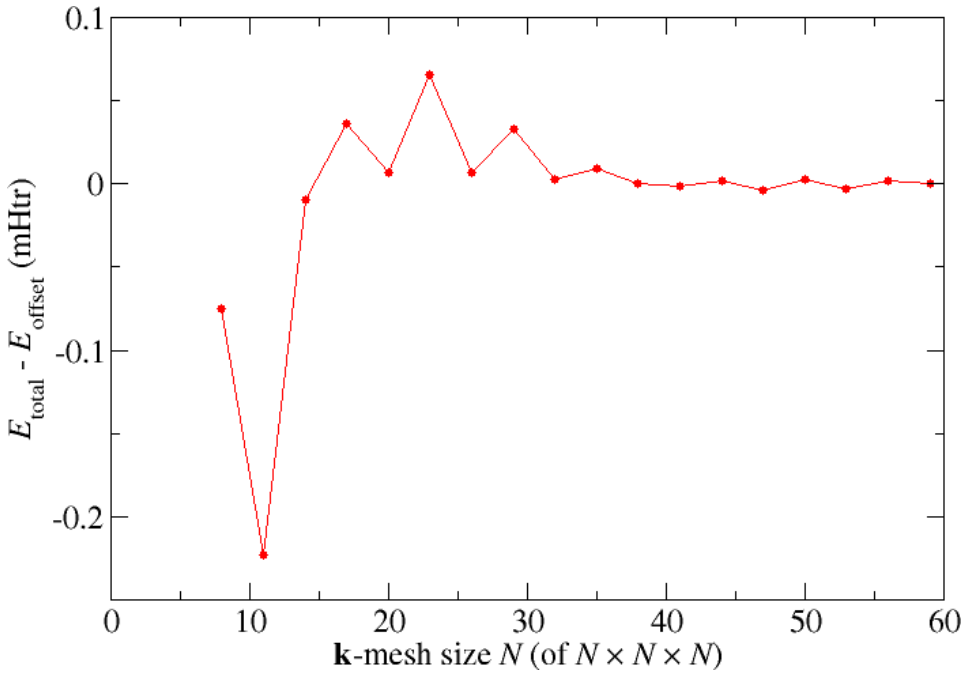

Figure 5.12: **k**-point mesh convergence for Cu.

other systems. It is related to the presence or absence of certain **k** points in the set.

In general the demands on the **k**-point set depend on the complexity of the electronic structure in the vicinity of the Fermi energy and the sensitivity of the investigated quantity on the details of the electronic structure in this energy range. This implies that for insulators rather coarse meshes are sufficient and for metals finer meshes are needed. Also the required mesh size is reciprocal to the unit cell size: The **k**-point density should be chosen to be the same for primitive unit cells and for supercells.

5.3.5 numbands convergence for spin-orbit coupling in 2nd variation

When performing calculations involving spin-orit coupling (SOC) in second variation the number of bands to be taken into account for the second variation step (`numbands`) also is a convergence parameter that has to be considered. In the following figure the respective total energy convergence for such calculations on elemental bulk Pb is shown. The default number of bands taken into account for this system is 24. Increasing the parameter yields the expected behavior, i.e., one obtains lower and lower total energies until some minimal value is reached.

Note that the convergence observed in the second variation step also relies on the LAPW basis. A change in the LAPW basis may lead to a different behavior, e.g., one needs more bands but also reaches lower total energies. This is especially relevant since the radial functions in the MT spheres are not constructed for the description of spin-orbit

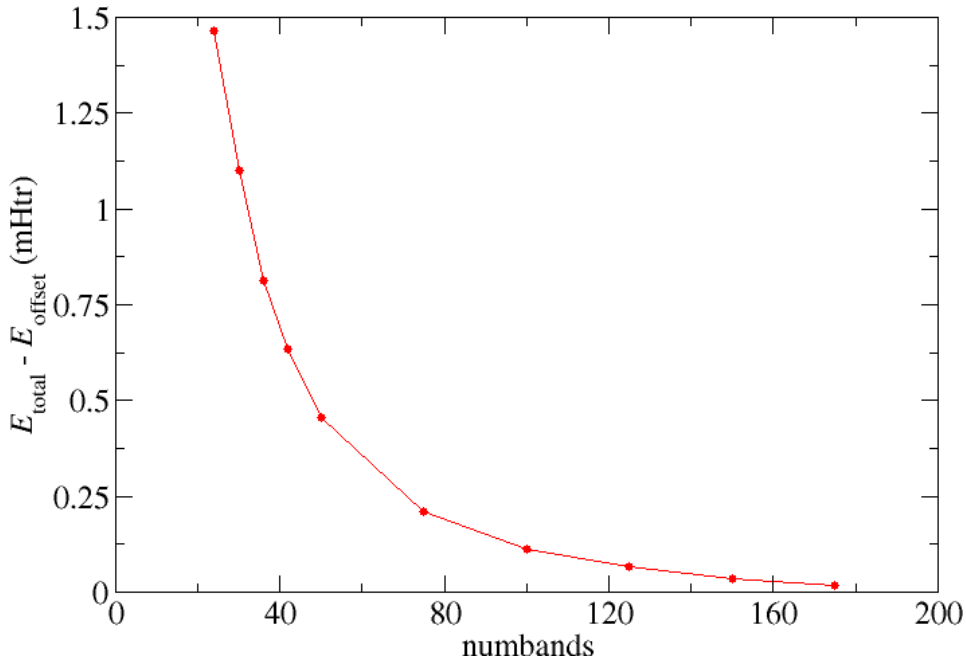

Figure 5.13: numbands convergence for SOC calculations on Pb.

coupling and it is hard to systematically converge the MT basis to obtain a high-quality representation for this purpose. The results of second variation SOC calculations therefore are approximative, though the approximation is good and allows quantitative predictions for many quantities.

5.4 Describing semicore states with local orbitals

If the treatment of a semicore state as a core electron leads to a ghost band the user typically resolves this issue by switching the treatment of these respective electrons to a valence electron treatment. This procedure involves several steps that have to be performed in a consistent way.

The starting point is the identification of the responsible semicore states. This is done by identifying those core electron states with the highest eigenenergies.

states are provided in the `coreStates` elements of the `out.xml` file. The number of core electrons lost from the respective MT sphere is also listed in this tag. An example for such an output is provided below. It features $3p1/2$ and $3p3/2$ states with very high lying eigenenergies.

```
<coreStates atomType="     1" atomicNumber=" 23" spin="1"
            kinEnergy="    941.0636778793" eigValSum="   -550.9805159504"
```

```
            lostElectrons=" 0.111893">
   <state n="1" l="0" j="0.5" energy="-195.9388108744" weight="2.000"/>
   <state n="2" l="0" j="0.5" energy="-21.4169509005" weight="2.000"/>
   <state n="2" l="1" j="0.5" energy="-17.9790788510" weight="2.000"/>
   <state n="2" l="1" j="1.5" energy="-17.7189379876" weight="4.000"/>
   <state n="3" l="0" j="0.5" energy="-1.8913495403" weight="2.000"/>
   <state n="3" l="1" j="0.5" energy="-0.9625902671" weight="2.000"/>
   <state n="3" l="1" j="1.5" energy="-0.9318007834" weight="4.000"/>
</coreStates>
```

After identifying the core electron states to be moved to the valence description the number of electrons in these states has to be counted. This is done by multiplying for each state the electrons in it by the number of atoms in the respective atom group and adding these numbers up for all considered states of all considered atom groups.

> electrons per atom is provided in `coreStates/state/@weight`. In general these are 2 electrons for s states, 6 electrons for p states, 10 electrons for d states, and 14 electrons for f states if spin-orbit splitting is neglected.

To move the description of the semicore electrons from the core electrons to the valence electrons the respective core electron states have to be removed in the input file and the number of valence electrons has to be increased.

> `electronConfig` tag the respective states listed in `atomSpecies/species/electronConfig/coreConfig` have to be moved directly to the section of the valence electrons in `atomSpecies/species/electronConfig/valenceConfig`. The number of valence electrons is specified in `calculationSetup/bzIntegration/@valenceElectrons`. It has to be adapted even if the electron configuration is specified directly.

The last step is the extension of the LAPW basis by local orbitals (LOs). For this you have to consider the main quantum number of the semicore states and the orbital character.

tag has to be inserted in the `atomSpecies/species` section. All of these tags have to be at the end of the section. The tag involves the specification of the LO type in `atomSpecies/species/lo/@type`. For the description of semicore states this has to be set to `SCLO`. It specifies details of the LO energy parameter calculation procedure. `SCLO` extrapolates the spherical effective MT potential by a confining potential outside the MT sphere, considers an atomic problem with this potential, and uses the eigenenergy related to the specified main quantum number and angular momentum quantum number as LO energy parameter. The main quantum number is specified in `atomSpecies/species/lo/@n` and the angular momentum quantum number in `atomSpecies/species/lo/@l`. The last parameter that has to be specified in this tag is the degree of the energy derivative of the solution to the atomic problem. This is specified in `atomSpecies/species/lo/@eDeviv`. For the most common usage of the function $u_l^\alpha(r_\alpha, E_l^{\text{lo}})$ this has to be set to 0. If higher order energy derivatives of the function have to be used the respective degree of the derivative has to be specified here. An example for the specifiifaction of $3p$ semicore LOs is shown below.

```
<lo type="SCLO" l="1" n="3" eDeriv="0"/>
```

5.5 Error messages

FLAPW calculations and DFT calculations in general are very complex. They require complex input, rely on complex files on the disc and are performed on complex computers. Therefore it is not surprising that at some point one will encounter error messages when working with such codes. To simplify the diagnosis of error messages related to using the Fleur code in this section the most important Fleur error messages will be discussed.

There are different types of error messages:

- **`juDFT-Error`**: This kind of error is thrown if a Fleur run enters a problematic state from which no recovery is possible. It is most likely due to unfortunate or erroneous calculation setups. But program bugs are also possible causes.
- **`juDFT-Warning`**: A warning is thrown if a Fleur rund enters a problematic state from which automatic recovery is unlikely. It is also thrown when a calculation is about to be performed that uses experimental features or features requiring detailed user insights to avoid unintended results. Fleur can be started such that the program flow does not stop when a warning is thrown.
- **Status messages**: Fleur writes out some messages which might look like errors to new users but are actually only notes that certain situations occured during a Fleur run. The program flow is unaffected by such messages.

- **Errors in linked software libraries**: A crash in a linked library can also trigger an error message. These messages look different than other Fleur errors and can have many different causes.

In general the most important advice is to actually read occuring error messages and also the last lines in the `out` file. This may already provide enough hints to overcome the respective problem. For more complex errors it may be a good idea to investigate the `out` in more detail and check whether certain output values are unexpected. Note: Depending on the used compiler the Fleur console output may be redirected to the `out` file.

5.5.1 Succesful Fleur run with multiple MPI processes

Even if you are performing a succesful Fleur calculation with multiple MPI processes you may encounter error messages. An example for such a case is provided below.

```
 ****************************************
 Run finished successfully
 Stop message:
   all done
 ****************************************
Rank:0 used    2.695GB/  2826072 kB
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   784  100    40  100   744    123   2293 --:--:-- --:--:-- --:--:--  2289
 Usage data send using curl: usage.json
OK

===================================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 0 PID 30853 RUNNING AT iff691
=   KILLED BY SIGNAL: 9 (Killed)
===================================================================================
```

At first the output says that the `Run finished successfully` then you receive a message about the reporting of the usage metrics and finally the output compains about a `BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES`. Note that the final error message may actually differ depending on the machine you use for the calculation.

If you see such an output everything is ok. The error message is thrown because whenever Fleur ends in a controlled way the MPI process that reaches the end point aborts all other MPI processes. This is done because in certain calculations only a single MPI process actually processes the code part leading to the ending subroutine. If it would

not be done in such a way other MPI processes may not end at all but remain running as zombie processes. Do not worry about whether all MPI processes actually did what they have to do. This is taken care of in the code.

Note that there are still situations in which zombie processes may remain after Fleur ends. This is the case if Fleur comes to an uncontrolled end, i.e., it crashes without providing a Fleur error message. You may still obtain an error message related to some software library. If you are unsure whether the Fleur ending was actually controlled check with `top` whether there are still Fleur processes running and terminate them manually with the `kill` command.

5.5.2 XML document file not parsable: inp.xml

This error indicates that the Fleur input file is ill-formed. This is probably due to an incomplete manual modification of the input file. You may obtain additional hints above this error message.

5.5.3 XML document cannot be validated against Schema.

This error indicates that at least one of the parameters in the Fleur input file has an illegal value. You may obtain additional hints above this error message. For example an error output like

```
inp.xml:16: element kPointCount: Schemas validity error : Element 'kPointCount',
            attribute 'count': '-5' is not a valid value of the atomic
            type 'xs:nonNegativeInteger'.
**************juDFT-Error*****************
Error message:XML document cannot be validated against Schema.
Error occurred in subroutine:xmlValidateDoc
Hint:See hints in lines directly above this error message.
Error from PE:0/1
****************************************
 Last kown location:
 Last timer:r_inpXML
 Timerstack:
 Timer:Initialization
 Timer:Total Run
 ****************************************
```

indicates that in line 16 of the input file the parameter `count` of the XML element `kPointCount` was set to a negative value, which is not allowed. Note that the additional hints above the error message are written out by the used libxml2 library. They may

vary on different machines and may even be missing for certain machine configurations. There also exist illegal parameter values that are not detected on this stage of the input parsing. In such a case one will obtain a different error message.

5.5.4 I/O warning : failed to load external entity "relax.xml"

This is not an error at all. When performing force relaxations Fleur writes out the relaxation history and the atom displacements for the next calculation to the file `relax.xml`. When starting Fleur this file is automatically included into the `inp.xml` file, even if no force relaxations are performed. Whenever this file is not present one obtains the warning that it could not be loaded. The consequence is that there will not be any displacements of the atom positions defined in the input file. The Fleur run will continue as intended.

5.5.5 Multiple differing error messages with MPI parallelization

If you start an MPI parallelized Fleur calculation you may obtain multiple differing error messages. These may or may not include XML validation and parsing errors, HDF5 errors, and different Fleur errors. In such a situation it is most likely that the different Fleur processes were actually not started such that they are connected by MPI. In consequence the different Fleur processes running in the same working directory may harm each other. For example one process may write a file while another process already reads it.

There are two possible reaons why such a situation occurs. The first case is that MPI is configured correctly but it is used on a Fleur executable that has not been compiled with MPI support. The other case is that a Fleur executable with MPI support is used but the MPI setup is wrong. In the most simple case this may be due to a missing command line option in the call of the MPI executable.

5.5.6 differ

The differ error comes in the two flavors `differ 1` and `differ 2`. As the error message says it is related to problems in solving the Dirac equation. The Dirac equation is solved for two purposes:

- The determination of the core electron states and energies.
- The determination of the energy parameters for the LAPW basis and the LOs.

The solver for the Dirac equation assumes that the eigenenergies are found within a certain energy window. If a differ error is thrown this means that the respective eigenenergy could not be found in that region. The reason for such a situation is typically a messed-up potential, for example due to ghost bands, a linear dependence of the LAPW basis,

an inconsistent Fleur input or other causes. Resolving this problem therefore is often dependent on the experience of the user. The general advice is to investigate the out file for possible hints on the cause, e.g., the identification of the affected atom and state, and also to check whether possible issues can identified in the Fleur input. In the end the input may have to be changed to overcome the problem.

5.5.7 E-field too big or No. of e- not correct

The efield error typically arises whenever the user sets up a charged unit cell. Charge neutrality is crucial when bulk systems are investigated. The most likely cause of this problem is an erroneous shifting of core electrons to the valence window. Typical mistakes here are due to a wrong setting of the new number of core states or a wrong calculation of the new number of valence electrons.

5.5.8 Error checking M.T. radii

MT radii checks fail whenever the MT spheres of two different atoms overlap. This can be due to a manual change in the MT radii or due to changes of in the unit related to structural relaxations or lattice constant calculations. A solution may be to slightly reduce the MT radii. If this error is due to a massive change in the unit cell the initial setup should also be adapted. Within force relaxations there may be an overshooting of the correction to the atom positions. In such a case it might be a good idea to erase the relaxation history and start with other initial displacements.

5.5.9 Too low eigenvalue detected

Fleur complains about too low eigenvalues in the calculation of the Fermi energy whenever there exists a valence electron energy eigenvalue that is far below the lowest energy parameter. Similar to the differ error the cause for this problem typically is a messed-up potential and the advice to the user is again to investigate the output and double-check the input. Note that this complaint only is a warning. Fleur can be started such that warnings are only written out but the program is not stopped. However, in most cases ignoring this warning in such a way will yield a differ error.

5.5.10 U+SS+EV-PAR and U+LO+EV-PAR

Certain features are not compatible to each other. The error message `U+SS+EV-PAR` indicates that LDA+U is used together with spin-spiral calculations and eigenvector parallelization. One has to get rid of one of these aspects, for example by using a different

parallelization scheme. The error message `U+LO+EV-PAR` indicates the simultaneous usage of LDA+U, local orbitals, and eigenvector parallelization. This is also not possible.

5.5.11 Coretail option cannot be used!!!

The rigorous treatment of the core elctron tails outside the originating MT sphere is not implemented for calculations employing non-collinear magnetism. If `calculationSetup/magnetism/@l_noco` and `calculationSetup/coreElectrons/@ctail` are both set to `T` Fleur complains that the coretail option cannot be used. In such a case one has to set `calculationSetup/coreElectrons/@ctail` to `F`.

5.5.12 e>vz0 and e >= vz0

Whenever film calculations are performed energy parameters also have to be defined or determined for the vacuum regions. The errors `e>vz0` and `e >= vz0` indicate that the energy parameters are above the vacuum potential at an infinite distance from the film. Such a situation does not yield a state bound to the film. A vacuum function for the LAPW basis therefore cannot be calculated. The solution is to modify the vacuum energy parameters in the Fleur input file such that they are below the vacuum potential at infinity.

5.5.13 Determination of fermi-level did not converge

Whenever a complex electronic structure in the vicinity of the Fermi energy is sampled by a very coarse **k**-point mesh and a very small `FermiSmearingEnergy` is used, the determination of the Fermi energy may not be possible. Such situations can be resolved by increasing the **k**-point density and/or increasing the smearing for the Fermi distribution used to determine the occupation numbers for the Kohn-Sham orbitals.

5.5.14 No input file found

If there exists no input file in the working directory, i.e., no `inp.xml` file , Fleur has no way of finding out what it is supposed to do. It therefore complains.

6 Reference

Contents

This chapter provides a short reference on the Fleur and inpgen input files and also a list of articles and books that may be of interest to users of the Fleur code.

6.1 The Fleur input generator

The basic input of Fleur is the inp.xml file . As it does not only contain switches to control the calculation but also a detailed setup of the system including e.g. its symmetries or the atomic parameters it is hard to set it up by hand. Hence, an input-file generator is provided.

The `inpgen` executable takes a simplified input file and generates defaults for:

> **Note** Migrating to newer FLEUR versions
> In some cases theinput-generator might help you to adjust older inp.xml files to the current version. See the section on modifying inp.xml for more details.

- the symmetry information
- the atom species and the symmetry equivalent atoms (atom groups)
- muffin-tin radii, l-cutoffs, and radial grid parameters for the atom species
- plane-wave cutoffs ($K_{\max}$, $G_{\max}$, $G_{\max}^{\mathrm{XC}}$).
- (in film calculations) the vacuum distance (D) and d-tilda ($\tilde{D}$)
- many more specialized parameters ...

In general the input generator does not know:

- is your system magnetic? If some elements (Fe,Co,Ni...) are in the unit cell the program sets `jspins=2` and puts magnetic moments. You might like to change jspins and specify different magnetic moments of our atom types.
- how many **k** points will you need? For metallic systems it might be more than for semiconductors or insulators. In the latter cases, also the mixing parameters might be chosen larger.
- is the specified energy range for the valence band ok? Normally, it should, but it's better to check, especially if LO's are used.

You have to modify your inp.xml file accordingly. Depending on your demands, you might want to change something else, e.g. the XC-functional, the switches for relaxation, use LDA+U etc.

6.1.1 Running inpgen

To call the input generator you tyically do

```
inpgen -f simple_file
```

input from the standard input.

The `inpgen` executable accepts a few command-line options. In particular you might find usefull

Option	Description
`-h`	list off all options
`-kpt string`	run the k-point generator
`-f filename`	name of simplified input file

6.1.2 Basic input

Your input should contain (in this order):

1. A title
2. Optionally: input switches (e.g. whether your atomic positions are in internal or scaled Cartesian units)
3. Lattice information (either a Bravais-matrix or a lattice type and the required constants (see below); in units of Bohr radii)
4. Atom information (an identifier (maybe the nuclear number) and the positions (in internal or scaled Cartesian coordinates)).
5. Optionally: for spin-spiral calculations or in case of spin-orbit coupling we need the **q** vector or the spin-quantization axis to determine the symmetry.
6. Optionally: Preset parameters (atoms/general)

6.1.2.1 Title

Your title cannot begin with an `&` and should not contain an `!`. Apart from that, you can specify any 80-character string you like.

6.1.2.2 Input switches

The namelist input should start with `&input` and end with `/`. Possible switches are:

switch	description
film=[t,f]	if true, assume film calculation (not necessary if dvac is specified)
cartesian=[t,f]	if true, input is given in scaled Cartesian units, if false, it is assumed to be in internal (lattice) units
cal_symm=[t,f]	if true, calculate space group symmetry, if false, read in space group symmetry info (file 'sym')

switch	description
checkinp=[t,f]	if true, program reads input and stops
inistop=[t,f]	if true, program stops after input file generation (not used now)
symor=[t,f]	if true, largest symmorphic subgroup is selected
oldfleur=[t,f]	if true, only 2D symmetry elements (+I,m_z) are accepted

6.1.2.3 An example (including the title):

```
3 layer Fe film, p(2x2) unit cell, p4mg reconstruction

&input symor=t oldfleur=t /
```

6.1.2.4 Lattice information

There are two possibilities to specify the lattice information: either you specify the Bravais matrix (plus scaling information) or the Bravais lattice and the required information (axis lengths and angles).

First variant:

The first 3 lines give the 3 lattice vectors; they are in scaled Cartesian coordinates. Then an overall scaling factor (aa) is given in a single line and independent (x,y,z) scaling is specified by scale(3) in a following line. For film calculations, the vacuum distance dvac is given in one line together with a3.

Example: tetragonal lattice for a film calculation:

```
1.0  0.0  0.0        ! a1
0.0  1.0  0.0        ! a2
0.0  0.0  1.0  0.9   ! a3 and dvac
4.89                 ! aa (lattice constant)
1.0  1.0  1.5        ! scale(1),scale(2),scale(3)
```

The overall scale is set by aa and scale(:) as follows: assume that we want the lattice vectors to be given by

```
a_i = ( a_i(1) xa , a_i(2) xb , a_i(3) xc )
```

then choose aa, scale such that: xa = aa · scale(1), etc. To make it easy to input sqrts, if scale(i) < 0, then scale = $\sqrt{|\text{scale}|}$ Example: hexagonal lattice

```
a1 = ( sqrt(3)/2 a , -1/2 a , 0.      )
a2 = ( sqrt(3)/2 a ,  1/2 a , 0.      )
a3 = ( 0.          , 0.     , c=1.62a )
```

You could specify the following:

```
 0.5  -0.5  0.0     ! a1
 0.5   0.5  0.0     ! a2
 0.0   0.0  1.0     ! a3
 6.2                ! lattice constant
-3.0   0.0  1.62    ! scale(2) is 1 by default
```

Second variant:

Alternatively, you may specify the lattice name and its parameters in a namelist input, e.g.

```
&lattice latsys='tP' a=4.89 c=6.9155 /
```

The following arguments are implemented: `latsys`, `a0` (default: 1.0), `a`, `b` (default: `a`), `c` (default: `a`), `alpha` (90 degree), `beta` (90 degree), `gamma` (90 degree). Hereby, `latsys` can be chosen from the following table (intended to work for all entries, up to now not all lattices work). `a0` is the overall scaling factor.

full name	No	short	other possible names	Description	Variants
simple-cubic	1	sc	cP, sc	cubic-P	
face-centered-cubic	2	fcc	cF, fcc	cubic-F	
body-centered-cubic	3	bcc	cI, bcc	cubic-I	
hexagonal	4	hcp	hP, hcp	hexagonal-P	(15)
rhombohedral	5	rho	hR, r, R	hexagonal-R	(16)
simple-tetragonal	6	tP	tP, st	tetragonal-P	
body-centered-tetragonal	7	bct	tI, bct	tetragonal-I	
simple-orthorhombic	8	orP	oP	orthorhombic-P	
face-centered-orthorhombic	9	orF	oF	orthorhombic-F	
body-centered-orthorhombic	10	orI	oI	orthorhombic-I	
base-centered-orthorhombic	11	orC	oC, oS	orthorhombic-C, orthorhombic-S	(17,18)
simple-monoclinic	12	moP	mP	monoclinic-P	
centered-monoclinic	13	moC	mC	monoclinic-C	(19,20)
triclinic	14	tcl	aP		

full name	No	short	other possible names	Description
hexagonal2	15	hdp		hexagonal-2 (60 degree angle)
rhombohedral2	16	hR2	hR2, r2, R2	hexagonal-R2
base-centered-orthorhombic2	17	orA	oA	orthorhombic-A (centering on
base-centered-orthorhombic3	18	orB	oB	orthorhombic-B (centering on
centered-monoclinic2	19	moA	mA	monoclinic-A (centering on A)
centered-monoclinic3	20	moB	mB	monoclinic-B (centering on B)
monoclinic-I	21	moI	mI	monoclinic-I

You should give the independent lattice parameters `a,b,c` and angles `alpha,beta,gamma` as far as required.

6.1.2.5 Atom information

First you give the number of atoms in a single line. If this number is negative, then we assume that only the representative atoms are given; this requires that the space group symmetry be given as input (see below).

Following are, for each atom in a line, the atomic identification number and the position. The identification number is used later as default for the nuclear charge (Z) of the atom. (When all atoms are specified and the symmetry has to be found, the program will try to relate all atoms of the same identifier by symmetry. If you want to manipulate specific atoms later (e.g. change the spin-quantization axis) you have to give these atoms different identifiers. Since they can be non-integer, you can e.g. specify 26.01 and 26.02 for two inequivalent Fe atoms, only the integer part will be used as Z of the atom.)

The input of the atomic positions can be either in scaled Cartesian or lattice vector units, as determined by the logical `cartesian` (see above). For supercells, it is sometimes more natural to specify positions in scaled Cartesian units.

A possible input (for CsCl) would be:

```
 2
55 0.0 0.0 0.0
17 0.5 0.5 0.5
```

or, for a p4g reconstructed Fe trilayer specifying the symmetry:

```
-2
26 0.00 0.00 0.0
26 0.18 0.32 2.5
```

```
 &gen          3

   -1    0    0         0.00000
    0   -1    0         0.00000
    0    0   -1         0.00000

    0   -1    0         0.00000
    1    0    0         0.00000
    0    0    1         0.00000

   -1    0    0         0.50000
    0    1    0         0.50000
    0    0    1         0.00000 /
```

Here, `&gen` indicates, that just the generators are listed (the 3×3 block is the rotation matrix [only integers], the floating numbers denote the shift); if you like to specify all symmetry elements, you should start with `&sym`. You have furthermore the possibility to specify a global shift of coordinates with e.g.

```
 &shift 0.5 0.5 0.5 /
```

or, to introduce additional scaling factors

```
 &factor 3.0 3.0 1.0 /
```

by which your atomic positions will be divided (the name "factor" is thus slightly counterintuitive).

6.1.2.6 Ending an input file

If `inpgen` should stop reading the file earlier (e.g. you have some comments below in the file) or if `inpgen` fails to recognize the end of the input file (which happens with some compilers), one can use the following line:

```
 &end /
```

6.1.2.7 Special cases

Film calculations

In the case of a film calculation, the surface normal is always chosen in z-direction. A two-dimensional Bravais lattice then corresponds to the three-dimensional one according to the following table:

lattice	description
square	primitive tetragonal
primitive rectangular	primitive orthorhombic
centered rectangular	base centered orthorhombic
hexagonal	hexagonal
oblique	monoclinic

The z-coordinates of all atoms have to be specified in Cartesian units (a.u.), since there is no lattice in the third dimension, to which these values could be referred. Since the vacuum boundaries will be chosen symmetrically around the $z = 0$ plane (i.e. -dvac/2 and dvac/2), the atoms should also be placed symmetrically around this plane.

The initial values specified for `a3` and `dvac` (i.e. the third dimension, see above) will be adjusted automatically so that all atoms fit in the unit cell. This only works if the atoms have been placed symmetrically around the $z = 0$ plane.

Spin-spiral or SOC

If you intend a spin-spiral calculation, or to include spin-orbit coupling, this can affect the symmetry of your system:

- a spin spiral in the direction of some vector $\mathbf{q}$ is only consistent with symmetry elements that operate on a plane perpendicular to $\mathbf{q}$, while
- (self-consistent) inclusion of spin-orbit coupling is incompatible with any symmetry element that changes the polar vector of orbital momentum L that is supposed to be parallel to the spin-quantization axis (SQA)

Therefore, we need to specify either $\mathbf{q}$ or the SQA, e.g.:

```
&qss 0.0 0.0 0.1 /
```

(the 3 numbers are the x,y,z components of $\mathbf{q}$) to specify a spin-spiral in z-direction, or

```
&soc 1.5708 0.0 /
```

(the 2 numbers are θ and ϕ of the SQA) to indicate that the SQA is in x-direction.

Be careful if symmetry operations that are compatible with the chosen $\mathbf{q}$ vector relate two atoms in your structure, they also will have the same SQA in the muffin-tins!

6.1.2.8 Preset parameters

Atoms

After you have given general information on your setup, you can specify a number of parameters for one or several atoms that the input-file generator will use while generating the inp file instead of determining the values by itself.

The list of parameters for one atom must contain a leading `&atom` flag and end with a `/`. You have to specify the atom for which you set the parameters by using the parameter `element`. If there are more atoms of the same element, you can specify the atom you wish to modify by additionally setting the `id` tag.

You might encounter the situation that different atoms of the same element need to be treated differently. For example, atoms might be in different chemical surounding or have different magnetic properties. You might also be in a situation in which you ant to break the symmetry between different atoms on purpose. In this cases an additional `id` tag should be supplied to the element. For example you could have not only Silicon as `&atom element=14 /` but also `&atom element=14 id=14.1/` in your list of atoms.

> **note** Use of the id-flag
> You should be aware of the followingfeature regarding the `id` flag: a) It breaks the symmetry of the system, so if you specify atoms with different ids, they are no longer symmetry equivalent, even if they might still belong to the same species in the resulting `inp.xml`. b) If you want to modify the muffin-tin radius, this only works if you specify the `id` flag, not with the default element settings.

All parameters available are

parameter	description
element=[name of the element]	identifies the atom to modify by its element name. This is a string. You must specify this.
id=[atomic identification number]	identifies the atom you wish to modify.
z=[charge number]	specifies the charge number of the atom.
rmt=[muffin-tin radius]	specifies a muffin-tin radius (only if id is specified!).
dx=[log increment]	specifies the logarithmic increment of the radial mesh.
jri=[# mesh points]	specifies the number of mesh points of the radial mesh.
lmax=[spherical angular momentum]	specifies the maximal spherical angular momentum.
lnonsph=[nonspherical angular momentum]	specifies the maximal angular momentum up to which non-spherical potentials are included.

parameter	description
ncst=[number of core state]	specifies the number of states you wish to include in the core.
econfig=[core states|valence states]	specifies, which states of the atom to modify are put into the core and which are treated as valence states. This is a string. You can use `[element name of noble gas]` to shorten the list. d and f states will be filled preferring magnetization.
bmu=[magnetic moment]	specifies the magnetic moment.
lo=[list of local orbitals]	specifies, which states shall be treated as local orbitals. This is a string.

General

You might also want to set more general parameters like the choice of the exchange-correlation potential or the desired reciprocal grid in the Brillouin zone beforehand. Those parameters can be given as a namelist using the `&comp`, `&exco`, `&film` and/or `&kpt` flag. The corresponding line in the input-file for the input-file generator has to end with a `/`. All parameters available are, sorted by their affiliation

&comp:

parameter	description
jspins=[number of spins]	specifies the number of spins for the calculation.
frcor=[frozen core?]	specifies whether or not the frozen-core approximation is used.
ctail=[core-tail correction?]	specifies whether or not the core-tail correction is used.
kcrel=[fully-magnetic dirac core?]	specifies whether or not the core is treated fully-relativistic.
gmax=[dop PW-cutoff]	specifies the plane-wave cutoff for the density and potential Fourier expansion.
gmaxxc=[xc-pot PW-cutoff]	specifies the plane-wave cutoff for the exchange-correlation potential Fourier expansion.
kmax=[basis set size]	specifies the cutoff up to which plane-waves are included into the basis.

&exco:

parameter	description
xctyp=[xc-potential]	specifies the choice of the exchange-correlation potential. This is a string.
relxc=[relativistic?]	specifies whether or not relativistic corrections are used.

&film:

parameter	description
dvac=[vacuum boundary]	specifies the vacuum boundary in case of film calculations.
dtild=[z-boundary for 3D-PW box]	specifies the z-boundary for the 3D plane-wave box in case of film calculations.

&kpt:

parameter	description
nkpt=[number of k-pts]	specifies the number of k-points in the IBZ to be used.
div1=[number of k-pts x-direction]	specifies the exact number of k-points to be used in the full BZ zone along x-direction (equidistant mesh).
div2=[number of k-pts y-direction]	specifies the exact number of k-points to be used in the full BZ zone along y-direction (equidistant mesh).
div3=[number of k-pts z-direction]	specifies the exact number of k-points to be used in the full BZ zone along z-direction (equidistant mesh).
tkb=[smearing parameter]	specifies a smearing parameter for Gauss- or Fermi-smearing method.
tria=[triangular method]	specifies whether or not triangular method shall be used.

Here is an example of an inpgen input file for Europium Titanate in which local orbitals are used for the $5s$ and $5p$ states of Europium and the muffin-tin radius of one Oxygen atom is manually set. Also, the exchange-correlation potential is chosen to be that of Vosko, Wilk, and Nusair and a **k**-point mesh is defined for the Brillouin-zone.

```
Europium Titanate Perovskite Structure

 &input cartesian=t inistop=t oldfleur=f /

 &lattice latsys='sc' a= 7.38 a0= 1.0 /

   5
   63     0.000  0.000  0.000
```

```
  08.01  0.000  0.500  0.500
  08.02  0.500  0.000  0.500
  08.03  0.500  0.500  0.000
  22     0.500  0.500  0.500

&atom element="eu" lo="5s 5p" econfig="[Kr] 4d10|4f7 5s2 5p6 6s2" /
&atom element="o" id=08.03 rmt=1.17 /
&exco xctyp='vwn' /
&kpt div1=5 div2=5 div3=5 /
```

6.1.3 Modifying inp.xml

The input generator can also read an existing `inp.xml` file. This is achieved by using `inpgen -inp.xml` as the command. At the moment two use cases exist for this feature.

Some old versions of an `inp.xml` can be read and converted to the current version. We are trying to improve this feature to make sure that at least inp.xml files of older releases can be converted. However, not all features might be included and not all switches or settings will be transfered. So please check the resulting file carefully.

An existing `inp.xml` can also be read to modify the k-points as discussed in the next section.

6.1.4 K-Point Generator

The generation of K-Points is now always performed by `inpgen`. The `fleur` executable no longer contains code for k-point generation and a list of K-Points now has to be provided in the `inp.xml` file.

Beside specifying k-points in the simple input for `inpgen` as outlined above. You can explicitely specify options for generating K-Points on the command line of 'inpgen'. For example you might want to call

```
inpgen -inp.xml -kpt testset#gamma@grid=10,10,10
```

Here we use an already existing inp.xml file (by specifying `-inp.xml`) and invoke the k-point generator with the `-k` option. This option expects a string specifying the k-point set to generate. The general form of this string is:

```
name#modifier@mode=details
```

- name: specifies a name for the set. (Optional)

- modifier: some modes of k-point generation can be modified with options (see below)
- mode: `inpgen` supports several different modes of k-point generation (see below)
- details: each mode needs parameters that are described below.

Possible modifier currently include `gamma` to indicate that the Gamma-Point should be included in the k-point set. The modifiers `gauss`, `tria`, `tetra` change the default integration method from the standard method. And the 'soc' modifier indicated that a k-point set compatible with spin-orbit coupling should be generated.

6.1.4.1 Grid mode

This mode is choosen by using `grid` as the mode in the k-point string. It expects the dimension of the grid as three integers separated by ',' in the details part of the string. The grid can be modified by all modifiers. For example you will use this mode as

```
inpgen -inp.xml -kpt gamma@grid=10,10,10
```

6.1.4.2 Density mode

This mode is activated by the `den` keyword for the mode. The requested density of k-points must be given as real number. Again, all modifiers are supported. Example:

```
inpgen -inp.xml -kpt testname#den=0.1
```

6.1.4.3 Number mode

This mode is activated by the `nk` keyword. It expects the approximate number of k-points to be used and is compatible with all modifiers.

```
inpgen -inp.xml -kpt nk=10
```

6.1.4.4 Band structure mode

This is a more complex mode generated by the using `band` as a mode. It expects the total number of k-points to generate and does not support any of the modifiers. In the most simple form you can use it as

```
inpgen -inp.xml -kpt band=100
```

to generate a bandstructure with 100 k-points.

In this most simple form, the code will try to determine a 'standard path' for your structure connecting a set of special k-points. However, you might not find this path appropriate or your structure might not be recognized. Hence, there is an additional option to overwrite the default set of special points. For example you can specify

```
-kpt band=100 -kptsPath "gamma=0,0,0;x=0.5,0.5,0.5"
```

Please note:

- The string to specify the special points should contain a list of such points (at least two to have a single line).
- The different points are given in the form `name=kx,ky,kz`.
- As the different points must be separated by `;` you have to set the string in quotation marks to avoid shell confusion.

6.2 The Fleur input file

Fleur expects all input (execpt some computational settings specified on the command line) in the `inp.xml` file in the current working directory. This file is usually produced by the input-generator 'inpgen' but Fleur workflows require adjustments to this file.

6.2.1 Example inp.xml file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<fleurInput fleurInputVersion="0.35">
   <comment>
      Si bulk
   </comment>
   <calculationSetup>
      <cutoffs Kmax="3.70" Gmax="11.10" GmaxXC="9.20" numbands="0"/>
      <scfLoop itmax="15" minDistance=".00001" maxIterBroyd="15"
```

```
            imix="Anderson" alpha=".05" precondParam="0.0" spinf="2.0"/>
   <coreElectrons ctail="T" frcor="F" kcrel="0" coretail_lmax="0"/>
   <magnetism jspins="1" l_noco="F" swsp="F" lflip="F"/>
   <soc theta=".00000000" phi=".00000000" l_soc="F" spav="F"/>
   <expertModes gw="0" secvar="F"/>
   <geometryOptimization l_f="F" forcealpha="1.00" forcemix="BFGS"
                         epsdisp=".00001" epsforce=".00001"/>
   <ldaU l_linMix="F" mixParam=".050000" spinf="1.000000"/>
   <xcFunctional name="pbe" relativisticCorrections="F"/>
</calculationSetup>
<cell>
   <bzIntegration valenceElectrons="8.00000000" mode="hist"
                  fermiSmearingEnergy=".00100000">
      <kPointListSelection listName="default-1"/>
      <kPointLists>
         <kPointList name="default-1" count="8" nx="2" ny="2" nz="2" type="mesh">
            <kPoint weight="  1.00"> -1.00/4.00  -1.00/4.00  -1.00/4.00</kPoint>
            <kPoint weight="  1.00">  1.00/4.00  -1.00/4.00  -1.00/4.00</kPoint>
            <kPoint weight="  1.00"> -1.00/4.00   1.00/4.00  -1.00/4.00</kPoint>
            <kPoint weight="  1.00">  1.00/4.00   1.00/4.00  -1.00/4.00</kPoint>
            <kPoint weight="  1.00"> -1.00/4.00  -1.00/4.00   1.00/4.00</kPoint>
            <kPoint weight="  1.00">  1.00/4.00  -1.00/4.00   1.00/4.00</kPoint>
            <kPoint weight="  1.00"> -1.00/4.00   1.00/4.00   1.00/4.00</kPoint>
            <kPoint weight="  1.00">  1.00/4.00   1.00/4.00   1.00/4.00</kPoint>
         </kPointList>
      </kPointLists>
   </bzIntegration>
   <symmetryOperations>
      <symOp>
         <row-1>1 0 0 .0000000000</row-1>
         <row-2>0 1 0 .0000000000</row-2>
         <row-3>0 0 1 .0000000000</row-3>
      </symOp>
   </symmetryOperations>
   <bulkLattice scale="1.0000000000" latnam="any">
      <bravaisMatrix>
         <row-1>.0000000000 5.1306085335 5.1306085335</row-1>
         <row-2>5.1306085335 .0000000000 5.1306085335</row-2>
         <row-3>5.1306085335 5.1306085335 .0000000000</row-3>
      </bravaisMatrix>
   </bulkLattice>
</cell>
<atomSpecies>
   <species name="Silicon (Si)" element="Si" atomicNumber="14">
```

```
        <mtSphere radius="2.16000000" gridPoints="521" logIncrement=".02200000"/
        <atomicCutoffs lmax="8" lnonsphr="6"/>
         <electronConfig>
           <coreConfig>(1s1/2) (2s1/2) (2p1/2) (2p3/2) </coreConfig>
           <valenceConfig>(3s1/2) (3p1/2) (3p3/2)</valenceConfig>
           <stateOccupation state="(3p1/2)" spinUp=".33333333" spinDown=".33333
           <stateOccupation state="(3p3/2)" spinUp=".66666667" spinDown=".66666
        </electronConfig>
        <energyParameters s="3" p="3" d="3" f="4"/>
     </species>
  </atomSpecies>
  <atomGroups>
     <atomGroup species="Silicon (Si)">
        <relPos label="        1">1.000/8.000 1.000/8.000 1.000/8.000</relPos>
        <force calculate="T" relaxXYZ="TTT"/>
     </atomGroup>
     <atomGroup species="Silicon (Si)">
        <relPos label="        2">-1.000/8.000 -1.000/8.000 -1.000/8.000</relPos
        <force calculate="T" relaxXYZ="TTT"/>
     </atomGroup>
  </atomGroups>
  <output dos="F" band="F" slice="F" >
     <checks vchk="F" cdinf="F"/>
     <bandDOS minEnergy="-.50" maxEnergy=".50"
                     sigma=".0150"/>
     <vacuumDOS layers="0" integ="F" star="F" nstars="0" locx1=".00000"
                locy1=".00000" locx2=".00000" locy2=".00000" nstm="0"
                tworkf=".00000"/>
     <unfoldingBand unfoldBand="F" supercellX="1"
                    supercellY="1" supercellZ="1"/>
     <plotting iplot="0" />
     <chargeDensitySlicing numkpt="0" minEigenval=".00000"
                           maxEigenval=".00000" nnne="0" pallst="F"/>
     <specialOutput eonly="F" bmt="F"/>
     <magneticCircularDichroism mcd="F" energyLo="-10.0000" energyUp=".0000"/>
  </output>
  <!-- We include the file relax.inp here to enable relaxations
       (see documentation) -->
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
              href="relax.xml">
     <xi:fallback/>
  </xi:include>
</fleurInput>
```

Being an XML file `inp.xml` starts with some general XML information in line 1. The rest of the file is enclosed in the `<fleurInput>` element that carries as an attribute the version number of the input file format. Within `<fleurInput>` the input file consists of several sections to be discussed in detail below.

6.2.2 Comment and Constants sections

```
<comment>
   Si bulk
</comment>
```

The comment section is optional. It encloses a simple line of text that is written out as part of the `inp.xml` into the `out.xml` file.

```
<constants>
   <constant name="myConst" value="5.1673552752"/>
</constants>
```

The constants section is optional and not part of the example `inp.xml` file shown above. It can be used to define constants that can then be used in other parts of the XML input file, e.g., the lattice setup or the declaration of the atom positions. The constants element may enclose multiple constant definitions. Each one has to provide the name and value of the respective constant.

6.2.3 Calculation Setup section

```
   <calculationSetup>
      <cutoffs Kmax="3.70" Gmax="11.10" GmaxXC="9.20" numbands="0"/>
      <scfLoop itmax="15" minDistance=".00001" maxIterBroyd="99"
               imix="Anderson" alpha=".05" precondParam="0.0" spinf="2.0"/>
      <coreElectrons ctail="T" frcor="F" kcrel="0" coretail_lmax="0"/>
      <magnetism jspins="1" l_noco="F" swsp="F" lflip="F"/>
      <soc theta=".00000000" phi=".00000000" l_soc="F" spav="F"/>
      <expertModes gw="0" secvar="F"/>
      <geometryOptimization l_f="F" forcealpha="1.00" forcemix="BFGS"
                            epsdisp=".00001" epsforce=".00001"/>
      <ldaU l_linMix="F" mixParam=".050000" spinf="1.000000"/>
      <xcFunctional name="pbe" relativisticCorrections="F"/>
   </calculationSetup>
```

The calculation setup section covers the input of general numerical parameters controlling the Fleur calculation.

Tag	Attribute	Description
cutoffs		The main cutoff parameters for the calculation
	Kmax	The cutoff for the basis functions $K_{\max} = \|\mathbf{k}+\mathbf{G}\|_{\max}$
	Gmax	The cutoff for the density
	GmaxXC	The cutoff used for the potential when the XC functional is calculated
	numbands	The number of eigenvalues to be calculated at each k point. A value of 0 is converted to a default value.
scfLoop		Parameters controlling the number of SCF loop iterations and the mixing scheme
	itmax	The number of SCF loop iterations to be performed by Fleur
	maxIterBroyd	The number of iterations to be taken into account by Broyden-like mixing schemes. The mixing history is periodically deleted with the periodicity specified by this parameter.
	imix	The mixing scheme. This can be one of "straight", "Broyden1", "Broyden2", and "Anderson"
	alpha	The mixing factor
	precondParam	The preconditioning parameter for bulk metals. Typical value: 0.8. Choose higher mixing parameter, e.g. 0.25.
	spinf	The spin mixing factor enhancement
coreElectrons		Parameters for the treatment of the core electrons
	ctail	Use core tail corrections.
	frcor	The frozen core approximation can be activated here.
	kcrel	If true fully relativistic core routines are used, otherwise only spin-polarized routines.
	coretail_lmax	Cutoff for the expansion of the core-tail into other MT spheres. Also relevant for initial charge generation.
magnetism		Parameters for controlling the degree of magnetism considered in the calculation
	jspins	The number of spins to be considered: 1 for nonmagnetic calculations and 2 for calculations incorporating magnetism.
	l_noco	Set this to true to consider not only collinear but also non-collinear magnetism See the section on the non-collinear magnetism setup for details.
	swsp	If true generate spin-polarized from unpolarized density.
	lflip	If true flip spin directions for each atom with set flipSpin flag.
soc		Parameters needed for calculations with spin-orbit coupling

Tag	Attribute	Description
	theta	The spin quantization axis is given by the theta and phi angles.
	phi	The spin quantization axis is given by the theta and phi angles.
	l_soc	This switch is used to toggle the consideration of spin-orbit coupling in the respective calculation .
	spav	Construct spin-orbit operator from spin-averaged potential.
expertModes		Parameters for the control of certain advanced Fleur calculation modes
	gw	The different output modes for GW approximation calculations are set here.
	secvar	Treat the nonspherical part of the Hamiltonian in second variation.
geometryOptimization		Parameters required for force calculations and the optimization of atom positions
	l_f	l_f is used to switch on the calculation of forces.
	forcealpha	The mixing parameter for the forces when performing structural optimizations.
	forcemix	The mixing procedure for the forces. Choose one of "BFGS" and "straight"
	epsdisp	The convergence criterion for atom displacements.
	epsforce	The convergence criterion for the forces. For further options on relaxation..
ldaU		Optional parameters for the density matrix mixing in LDA+U calculations. See the LDA+U setup for details.
xcFunctional		see below

The exchange correlation functional section consists of a single xml element with the two attributes `name` and `relativisticCorrections`. The XC functional is specified by the name attribute:

LDA functionals	
x-a	
wign	
mjw	
pz	The functional by Perdew and Zunger
vwn	The functional by Vosko, Wilk, and Nusair
bh	The functional by Barth and Hedin

GGA functionals	
pw91	The functional by Perdew and Wang
pbe	The functional by Perdew, Burke, and Ernzerhof
rpbe	The revPBE functional by Zhang and Yang
Rpbe	The RPBE functional by Hammer, Hansen, and Nørskov
wc	The functional by Wu and Cohen

Tag	Attribute	Description
	relativisticCorrections	A boolean switch used to activate optional relativistic corrections according to MacDonnald-Vosko.

6.2.4 Cell section

```
<cell>
   <bzIntegration valenceElectrons="20.00000000" mode="hist"
                  fermiSmearingEnergy=".00100000">
      <kPointListSelection listName="default-1"/>
      <kPointLists>
         <kPointList name="default-1" count="8" nx="2" ny="2" nz="2" type="mes
            <kPoint weight="  1.00"> -1.00/4.00  -1.00/4.00  -1.00/4.00</kPoi
            <kPoint weight="  1.00">  1.00/4.00  -1.00/4.00  -1.00/4.00</kPoi
            <kPoint weight="  1.00"> -1.00/4.00   1.00/4.00  -1.00/4.00</kPoi
            <kPoint weight="  1.00">  1.00/4.00   1.00/4.00  -1.00/4.00</kPoi
            <kPoint weight="  1.00"> -1.00/4.00  -1.00/4.00   1.00/4.00</kPoi
            <kPoint weight="  1.00">  1.00/4.00  -1.00/4.00   1.00/4.00</kPoi
            <kPoint weight="  1.00"> -1.00/4.00   1.00/4.00   1.00/4.00</kPoi
            <kPoint weight="  1.00">  1.00/4.00   1.00/4.00   1.00/4.00</kPoi
         </kPointList>
      </kPointLists>
   </bzIntegration>
   <symmetryOperations>
      <symOp>
         <row-1>1 0 0 .0000000000</row-1>
         <row-2>0 1 0 .0000000000</row-2>
         <row-3>0 0 1 .0000000000</row-3>
      </symOp>
   </symmetryOperations>
   <bulkLattice scale="1.0000000000" latnam="any">
      <bravaisMatrix>
```

```
        <row-1>.0000000000 5.1306085335 5.1306085335</row-1>
        <row-2>5.1306085335 .0000000000 5.1306085335</row-2>
        <row-3>5.1306085335 5.1306085335 .0000000000</row-3>
      </bravaisMatrix>
    </bulkLattice>
  </cell>
```

The cell section covers the declaration of the k-points, of the symmetry operations available in the unit cell and the definition of the lattice vectors. For details please see the sections on the kpoint setup, Bravais lattice setup and the unit cell symmetry setup .

6.2.5 Atom species section

```
  <atomSpecies>
    <species name="Silicon (Si)" element="Si" atomicNumber="14">
      <mtSphere radius="2.16000000" gridPoints="521" logIncrement=".02200000"/>
      <atomicCutoffs lmax="8" lnonsphr="6"/>
       <electronConfig>
        <coreConfig>(1s1/2) (2s1/2) (2p1/2) (2p3/2) </coreConfig>
        <valenceConfig>(3s1/2) (3p1/2) (3p3/2)</valenceConfig>
        <stateOccupation state="(3p1/2)" spinUp=".33333333" spinDown=".33333333"/>
        <stateOccupation state="(3p3/2)" spinUp=".66666667" spinDown=".66666667"/>
      </electronConfig>
      <energyParameters s="3" p="3" d="3" f="4"/>
    </species>
  </atomSpecies>
```

The `atomSpecies` section is a tool to set identical numerical parameters for the atoms of different atom groups without introducing redundancy. For this several species with a unique name can be defined in the section. In the following `atomGroups` section each atom group is associated to one of the species.

Tag	Attribute	Description
species		The element defining a species. There can be multiple of these elements in this section.
	name	A name for the species. This should be unique within the set of species.
	element	The abbreviation of the chemical element.
	atomicNumber	The atomic number of the chemical element.

A species element contains other elements to determine its numerical parameters. Please

note that the order of these elements in the input file is predefined:

Tag	Attribute	Description
mtSphere		This element is used to define the properties of the muffin-tin spheres.
	radius	The radius of the MT sphere.
	gridPoints	The number of grid points on the radial mesh for this MT sphere.
	logIncrement	The logarithmic increment of the radial mesh.
atomicCutoffs		This element covers the l-cutoffs.
	lmax	The general l-cutoff for all atoms of the species.
	lnonsphr	The reduced l-cutoff for the calculation of contributions originating from non-spherical part of the potential
	lmaxAPW	If present the APW+lo approach will be used. This is the cutoff defining the lmax up to which LAPWs are used if no APW+lo local orbital is defined for the respective l. See also the respective article by G.K.H. Madsen.
energyParameters		This element sets the energy parameters.
	s	The main quantum number for the valence s electrons.
	p	The main quantum number for the valence p electrons.
	d	The main quantum number for the valence d electrons.
	f	The main quantum number for the valence f electrons.
special		
	socscale	A float in range from 0.0 to 1.0. Scales the magnitude of SOC at the species. socscale="0.0" switches SOC off.
	vca_charge	A float to specify an extra charge for calculations in the virtual crystal approximation for this species.
	lda	Logical switch to use LDA for this atom.
	b_field_mt	Zeeman field applied to this atom.
electronConfig		See the description of the electron configuration setup for details.
ldaU		Up to one ldaU element can be present to define a U parameter for this atom and a certain l channel.
	l	This is the l channel the U is supposed to affect.
	U	This is the U parameter in eV.
	J	This is the J parameter in eV.
	l_amf	If true the around mean field limit is employed, otherwise the atomic limit. For a more detailed description have a look at [[xmlLDAUSetup

Tag	Attribute	Description
lo		This element is used to introduce local orbitals to each of the associated atoms. It can be present multiple times. See the section on the local orbital setup for a more profound discussion.
	type	The type of the LO. This can be SCLO for semicore LOs or HELO for LOs at higher energies
	l	The angular momentum quantum number belonging to this LO
	n	The main quantum number for this LO
	eDeriv	The energy derivative of the additional radial function introduced with this LO. This is by default 0 to obtain conventional LOs. For HDLOs it has to be set to a finite positive integer value.

6.2.6 Atom groups section

```
<atomGroups>
   <atomGroup species="Silicon (Si)">
      <relPos label="        1">1.000/8.000 1.000/8.000 1.000/8.000</relPos>
      <force calculate="T" relaxXYZ="TTT"/>
   </atomGroup>
   <atomGroup species="Silicon (Si)">
      <relPos label="        2">-1.000/8.000 -1.000/8.000 -1.000/8.000</relPos>
      <force calculate="T" relaxXYZ="TTT"/>
   </atomGroup>
</atomGroups>
```

The atom groups section covers parameters relevant for each group of symmetry equivalent atoms. In the example the Fleur input file has been generated without symmetry detection. The two Si atoms are thus placed in different atom groups.

Tag	Attribute	Description
atomGroup		There is at least one atom group. Each atom in the unit cell has to be in one.
	species	The name of the species of this group's atoms.

Each atom group element encloses certain other elements:

Tag	Attribute	Description
relPos		See below
filmPos		See below
force		Some switches associated to force calculations.
	calculate	This boolean switch determines whether forces are calculated for the atoms of this group.
	relaxXYZ	Three boolean switches used declare in which directions the atom position may be optimized in force relaxation calculations.
nocoParams		See the description of the non-collinear magnetism setup for details.

The atom positions are defined within each atomGroup of symmetry equivalent atoms in the atom groups section of the input file. They can be provided as relative or film positions:

6.2.7 Relative positions

```
<atomGroup species="W-1">
   <relPos label="   1">.000000 1.0/2.0 .060000</relPos>
   <relPos label="   2">1.0/2.0 .000000 -.060000</relPos>
   <force calculate="T" relaxXYZ="TTT"/>
</atomGroup>
```

Typically for bulk materials the atom positions are provided in relative coordinates as fractions of the three lattice vectors. For this the `relPos` tag is used. In the example the atom group consists of two atoms at two different positions. The first one is the representative atom. As shown the relative coordinates are provided as three numbers within the `relPos` element. Note that each coordinate can also be provided by a short mathematical expression that does not contain any spaces, e.g., 1.0/4.0.

Each atom can be associated to a `label` that can be specified in the inpgen input. If there is no explicit labeling in the inpgen input the labels just associate a unique number with each atom. This labeling is only meant as a help to the user to keep an overview for setups of very large unit cells.

6.2.8 Film positions

```
<atomGroup species="W-2">
   <filmPos label="   1">1.0/2.0 1.0/2.0 -12.0314467594</filmPos>
   <filmPos label="   2">1.0/2.0 1.0/2.0 12.0314467594</filmPos>
```

```
      <force calculate="T" relaxXYZ="TTT"/>
   </atomGroup>
```

For calculations on films the atom positions are provided within the filmPos element. Here, the first two of the coordinates are relative, while the third coordinate in the direction normal to the film plane is absolute and in atomic units (Bohr radii).

6.2.9 Output section

```
<output dos="F" band="F" slice="F" >
   <checks vchk="F" cdinf="F"/>
   <bandDOS minEnergy="-.50" maxEnergy=".50"
                    sigma=".0150"/>
   <vacuumDOS layers="0" integ="F" star="F" nstars="0" locx1=".00000"
              locy1=".00000" locx2=".00000" locy2=".00000" nstm="0"
              tworkf=".00000"/>
   <unfoldingBand unfoldBand="F" supercellX="1"
                  supercellY="1" supercellZ="1"/>
   <plotting iplot="0" />
   <chargeDensitySlicing numkpt="0" minEigenval=".00000"
                         maxEigenval=".00000" nnne="0" pallst="F"/>
   <specialOutput eonly="F" bmt="F"/>
   <magneticCircularDichroism mcd="F" energyLo="-10.0000" energyUp=".0000"/>
</output>
```

The output section is optional. It covers parameters relevant for the generation of special output.

Tag	Attribute	Description
output		
	dos	A boolean switch that determines whether a density of states has to be calculated.
	band	A boolean switch that determines whether a band structure calculation should be performed.
	slice	A boolean switch controlling whether a slice has to be calculated. The associated parameters are set in the chargeDensitySlicing element.

If an attribute of the output element is set to true the associated enclosed element has to be present:

Tag	Attribute	Description
checks		The checks element covers several switches to perform and write out certain tests.
	vchk	Continuity checks of the potential at the MT and vacuum boundaries.
	cdinf	Calculation of partial charges and the continuity of the density.
	disp	Calculation of the distance between the in- and output potential.
bandDOS		Select the DOS calculation mode. For details have a look at the respective section .
vacuumDOS		For details have a look at the respective section .
unfoldingBand		
	unfoldBand	A boolean switch that defines if unfolding is used and additional weights are written.
	supercellX	The size of the supercell (in units of simple unit cells) (iteger value) in X direction.
	supercellY	The size of the supercell (in units of simple unit cells) (iteger value) in Y direction.
	supercellZ	The size of the supercell (in units of simple unit cells) (iteger value) in Z direction.
plotting		The plotting element groups several switches to plot the density and the potential.
	iplot	Calculate a charge density plot. For details see the respective section .
	score	If true the core charge is excluded from the plot.
	plplot	This switch allows the plotting of the potential from its respective files.
chargeDensitySlicing		
	numkpt	This is the number of the k-point which is used for the slice (k=0 : all k-points are used)
	minEigenval	This is the lower boundary for eigenvalues in the slice.
	maxEigenval	This is the upper boundary for eigenvalues in the slice.
	nnne	The number of eigenvalues used for the slice (nnne=0 : all eigenvalues between boundaries are taken into account)
	pallst	Set this to true if states above the Fermi level are plotted.
coreSpectrum		See the section on core spectrum calculations for details.
magneticCircularDichroism		See the section on magnetic circular dichroism calculations for details.

6.2.10 Force therorem section

```
<forceTheorem>
   <MAE theta="0.0 0.1*Pi" phi="0.0 0.2*Pi" />
</forceTheorem>
```

It is possible to apply the magnetic force theorem to perform efficient approximative calculations on the magnetocrystalline anisotropy energy (MAE), spin-spiral dispersions, the Dzyaloshinskii Moriya interaction (DMI), and the Heisenberg exchange interaction. To do so a `forceTheorem` section has to be inserted directly below the `output` section. For details see the section on the respective calculations .

6.2.11 Inclusion of the relax.xml file

```
<!-- We include the file relax.inp here to enable relaxations
   (see documentation) -->
   <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
               href="relax.xml">
      <xi:fallback/>
   </xi:include>
```

It is possible to automatically include other xml files into the Fleur input file by using the respective XInclude feature. By default this is done for the `relax.xml` file. The syntax above also provides a fallback that is used when the `relax.xml` file is not present: In that case it is just not included, but a message about this is written out to the terminal.

When starting new Fleur calculations the `relax.xml` file is typically not present. It is automatically created when geometry optimizations for the atom positions are performed. Its contents are the atom displacements calculated so far. These automatically modify the atom positions provided in the Fleur input file. Furthermore a history of atom positions and related forces for the previous optimization steps is found in the `relax.xml` file. This is needed for the determination of the next displacements for ongoing atom position optimization calculations.

Details on how geometry optimizations are performed are found in the respective section .

6.2.12 k-point set setup

note Creation of k-point sets
While you can use thedocumentation provided here to generate you own **k**-point. It is usually recommended and much simpler to use input generator for this task.

In the FLEUR inp.xml file you typically find a series of XML-tags like this the `cell` section:

```
<bzIntegration valenceElectrons="20.00000000" mode="hist" fermiSmearingEnergy=".0
   <kPointListSelection listName="default-1"/>
   <kPointLists>
      <kPointList name="default-1" count="2" nx="2" ny="2" nz="2" type="mesh">
         <kPoint weight="   2.00">  1.00/4.00   1.00/4.00   1.00/4.00</kPoint>
         <kPoint weight="   6.00">  1.00/4.00   2.00/4.00   2.00/4.00</kPoint>
      </kPointList>
   </kPointLists>
</bzIntegration>
```

Alternatively, the `kPointLists` might be put into a separate `kpts.xml` file and in the `inp.xml` you find the corresponding include.

```
<bzIntegration valenceElectrons="20.00000000" mode="hist" fermiSmearingEnergy=".0
   <kPointlistselection Listname="default-1" />
   <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="kpts.xml">
</bzIntegration>
```

6.2.12.1 Brillouin zone integration

The highest level tag to be found is the `bzIntegration` tag:

Attribute	Desciption
valenceElectrons	The total number of valence electrons in your system
mode	Method to determine the Fermi level. Supported are: hist - Use the histogram mode. This is the default. gauss - Use Gaussian smearing. tria - Use the tetrahedron method.
fermiSmearingEnergy	The Fermi smearing can be parametrized by this energy in Hartree.
fermiSmearingTemp	As an alternative to fermiSmearingEnergy a Fermi smearing temperature can be set in Kelvin.

6.2.12.2 Selecting a k-point set

Then, the `Kpointlistselection` tag with a single attribute `Listname` is used to select the k-point set from the List provided. The `Listname` has to correspond to a `Name`

attribute of one of the KpointLists.

6.2.12.3 The List of k-points

You might have different k-point sets in your setup, these KpointList XML elements are included into an outer KpointLists tag. Usually, the inpgen will generate multiple sets of k-points initially. With further calls to inpgen you might create further sets.

A Kpointlist is defined with the following attributes:

Attribute	Desciption
Name	Name of the k-point set to be used in Kpointlistselection. (required)
Count	Number of k-points in the set. (optional)
type	Description of the type: at present one of mesh,path,tria-bulk,tria,SPEX-mesh,unspecified (optional)
nx,ny,nz	Size of the mesh in three spatial directions. (optional)
nkq_pairs	Pairs of k,q vectors in a hybrid calculation. (optional)

Each kPoint element features the attribute weight and three numbers. The weight is the weight of the **k** point in the Brillouin zone integration. Each of the three numbers is divided by the value of the posScale attribute of the kPointList element to obtain the coordinates of the **k** point. In addition a label can be provided for use in bandstructure calculations.

6.2.13 Definition of the Bravais lattice

In the XML input file lattices for bulk or film unit cells can be defined in the cell section. The type of unit cell is selected by using either the BulkLattice or the FilmLattice XML elements. The definition of the details of the lattice is in both cases similar. The lattice vectors are given in atomic units (bohr radii). For the bulk lattice the following examples illustrate the different options to declare the shape of the unit cell:

```
<bulkLattice scale="1.00">
   <bravaisMatrix>
      <row-1>.0000000000 5.1673552752 5.1673552752</row-1>
      <row-2>5.1673552752 .0000000000 5.1673552752</row-2>
      <row-3>5.1673552752 5.1673552752 .0000000000</row-3>
```

```
   </bravaisMatrix>
</bulkLattice>
```

A simple way to define the lattice is to just provide the lattice vectors in the 3x3 Bravais matrix. The attribute scale of the bulkLattice element is a factor the matrix is multiplied with. It can be used to change the size of the unit cell, e.g., to find the optimized lattice constant from a sequence of DFT calculations. #### Film calculations

For film lattices several additional attributes have to be set. The attribute dVac defines the length of the unit cell in z-direction. dTilda defines the length of an artificial, virtual, extended unit cell used to determine the set of LAPW basis functions. This is required as the LAPWs for a given unit cell are adapted to periodic problems. For the direction normal to the film plane this periodicity is not present.

The lattice itself can again be declared by providing the Bravais matrix as it is done in the example. However, in this case the entries of the matrix in the third row and in the third column are ignored.

The vacuum energy parameters are also defined in the filmLattice XML element. These are specified by up to two vacuumEnergyParameters elements: For both vacua energy parameters can be provided for two spins. In nonmagnetic systems only the spinUp entry is considered. The entries define the energy parameters relative to the vacuum potential zero, i.e., the potential infinitely far away from the film.

```
<filmLattice scale="1.00"  dVac="47.66" dTilda="51.37">
   <bravaisMatrixFilm>
      <row-1>6.0157233797 .0000000000 </row-1>
      <row-2>.0000000000 6.0157233797 </row-2>
   </bravaisMatrixFilm>
   <vacuumEnergyParameters vacuum="1" spinUp="-.25" spinDown="-.25"/>
</filmLattice>
```

`bravaisMatrixFilm` tag with a 2D Bravais matrix.

6.2.14 Setup of the unit cell symmetry

The symmetry of the system is another important input to Fleur. It is specified by providing a list of symmetry operations. Usually, these operations are generated by the input-generator by inspection of the cell and atomic input.

warning Adjusting the symmetry
For certain calculations youmight want to modify the operations and for example remove symmetry operations. While this is easily possible by removing operations from the list in `inp.xml` you should be careful not to remove operations that map equivalent atoms onto each other within an atom group. If you remove such operations you have to adjust the assignment of atoms to groups.

The usual way to specify symmetry operations is by using the corresponding '' XML tag.

```
<symmetryOperations>
   <symOp>
      <row-1>1 0 0 .0000000000</row-1>
      <row-2>0 1 0 .0000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>-1 0 0 .0000000000</row-1>
      <row-2>0 1 0 .0000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>1 0 0 .0000000000</row-1>
      <row-2>0 -1 0 .0000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>-1 0 0 .0000000000</row-1>
      <row-2>0 -1 0 .0000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>0 -1 0 .5000000000</row-1>
      <row-2>-1 0 0 .5000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>0 -1 0 .5000000000</row-1>
      <row-2>1 0 0 .5000000000</row-2>
      <row-3>0 0 1 .0000000000</row-3>
   </symOp>
   <symOp>
      <row-1>0 1 0 .5000000000</row-1>
```

```
   <row-2>-1 0 0 .5000000000</row-2>
   <row-3>0 0 1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>0 1 0 .5000000000</row-1>
   <row-2>1 0 0 .5000000000</row-2>
   <row-3>0 0 1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>1 0 0 .5000000000</row-1>
   <row-2>0 1 0 .5000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>-1 0 0 .5000000000</row-1>
   <row-2>0 1 0 .5000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>1 0 0 .5000000000</row-1>
   <row-2>0 -1 0 .5000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>-1 0 0 .5000000000</row-1>
   <row-2>0 -1 0 .5000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>0 -1 0 .0000000000</row-1>
   <row-2>-1 0 0 .0000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>0 -1 0 .0000000000</row-1>
   <row-2>1 0 0 .0000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
   <row-1>0 1 0 .0000000000</row-1>
   <row-2>-1 0 0 .0000000000</row-2>
   <row-3>0 0 -1 .0000000000</row-3>
</symOp>
<symOp>
```

```
        <row-1>0 1 0 .0000000000</row-1>
        <row-2>1 0 0 .0000000000</row-2>
        <row-3>0 0 -1 .0000000000</row-3>
      </symOp>
    </symmetryOperations>
```

The symmetryOperations element allows to specify each symmetry operation directly. Each symmetry operation is given by a matrix of three rows and four columns, where the last column is a translation vector needed for nonsymmorphic symmetries. If the input file generator is invoked with the -explicit command line switch this form of declaring the symmetry operations is used in the inp.xml file.

note Including the sym.xml file
As this list can be long itmight be desired to provide the symmetry operations in a separate file. You can use the x-include option for this purpose.

6.2.15 Local orbital Setup

In Fleur a local orbital (LO) is given by an energy parameter, an angular momentum quantum number, and a definition of the kind of radial function used to construct the LO. Within the `inp.xml` file LOs are defined for certain species within the `atomSpecies` section. Some examples for such definitions are:

```
<lo type="SCLO" l="1" n="3" eDeriv="0"/>
```

An LO definition like this is typically used to define a local orbital used to represent semi-core states within the valence electron framework in an FLAPW calculation. Sometimes it is even better to add another LO to describe such a state as the energy parameter might not be perfectly adjusted. In such a case one might add an LO with the same parameters except the degree of the energy derivative (`eDeriv`) which would be 1.

```
<lo type="HELO" l="2" n="4" eDeriv="0"/>
```

An LO definition typically used to define local orbitals with radial functions at energy parameters in the range of the unoccupied states. Such LOs are typically used whenever the performed calculation explicitly considers the unoccupied states, e.g., in calculations employing the GW approximation to many-body perturbation theory. Another use for such LOs is the elimination of the linearization error within the FLAPW method.

```
<lo type="SCLO" l="0-3" n="4,4,3,4" eDeriv="2"/>
```

A definition of a set of local orbitals for the angular momentum quantum numbers 0 to 3 and corresponding main quantum numbers 4,4,3, and 4. Each of the LOs uses the second energy derivative of the solution to the radial scalar-relativistic approximation (SRA) to the Dirac equation as third radial function. Such sets of LOs can be used to overcome the linearization error in all relevant l channels. However, one has to be careful not to obtain a numerically singular overlap matrix for the radial functions in one of the l channels. If energy parameters for unoccupied states are used, this way of defining sets of LOs is very practical to cover a large range of energy and l quantum numbers in only a few lines in the input file.

In detail, the energy parameter for the LO is given by the LO type and the main quantum number. The main quantum number `n` defines the number of nodes (n-l-1) of the additional radial function constructed for the LO. The energy parameter is then obtained by solving the radial problem under certain boundary conditions defined by the type attribute:

LO-Type	Description
SCLO	A semicore local orbital. The spherical part of the potential in the MT sphere is extended by an artificial confining potential outside the MT sphere. The energy parameter then is the eigenenergy belonging to the solution to this problem with the given l and n quantum numbers.
HELO	A higher energy local orbital. Here the SRA to the radial Dirac equation is solved for different test energies as a differential equation outwards starting at the atomic nucleus. The energy parameter then is that energy whose solution yields the correct number of nodes and a logarithmic derivative of -(l+1) at the MT boundary. It is found by a bisection search algorithm.

The angular momentum quantum number l and the main quantum number n are defined by the associated attributes of the lo XML element. The entries for these values can either be single integer values or sequences of values. For the l quantum number these sequences can be defined by two numbers and a "-" in between or by comma separated values. For the n quantum number only comma separated values are allowed. Note that l and n quantum numbers are used in pairs: The i-th l quantum number together with the i-th n quantum number are used to define the i-th local orbital.

Note that if an enpara file is present the energy parameters defined in that file override the definitions in the inp.xml file. If the energy parameters are to be obtained by the energy center of mass (ECM) method, this additional file has to be used.

The kind of the additional radial function is given by the `eDeriv` attribute. If this is set to 0 the solution of the SRA to the radial Dirac equation at the given energy parameter is used. If it is set to finite positive integers the energy derivative of this solution of degree eDeriv is used to construct an HDLO (higher derivative local orbital).

Further reading

- Local orbitals for the representation of semicore states have originally been introduced by Singh et al.
- In the context of GW calculations local orbitals employing higher energy derivatives have been introduced by Friedrich et al.
- For the representation of unoccupied states local orbitals on the basis of the HELO -(l+1) criterion have been used by Betzinger et al.
- An analysis about the usefulness of different types of local orbitals to eliminate the linearization error for the representation of valence electrons has been performed by Michalicek et al.

6.2.16 Specifying an electron configuration

Within each species in the `atomSpecies` section an `electronConfig` element can be defined. This is used to declare which electron states are to be treated within the core electron framework and which have to be considered in the valence electron framework. Occupations for the different electron states are also set here. The `electronConfig` element is optional and only required if a setup differing from the default for the respective atom shall be considered. The following example demonstrates how an electron configuration is set:

```
<electronConfig>
   <coreConfig>[Xe] (4f5/2) (4f7/2)</coreConfig>
   <valenceConfig>(6s1/2) (5d3/2) (5d5/2) (6p1/2) (6p3/2)</valenceConfig>
   <stateOccupation state="(6p3/2)" spinUp="1.00" spinDown="1.00"/>
</electronConfig>
```

The `electronConfig` encloses the XML elements `coreConfig`, `valenceConfig` and possibly multiple `stateOccupation` elements.

The `coreConfig` element is used to provide a space separated list of electron states to be treated by the core framework. Certain subsets can be set in terms of noble gas configurations. The electron states together with the noble gas configurations are

noble gas configuration	electron states
[He]	(1s1/2)
[Ne]	(2s1/2) (2p1/2) (2p3/2)
[Ar]	(3s1/2) (3p1/2) (3p3/2)
[Kr]	(4s1/2) (3d3/2) (3d5/2) (4p1/2) (4p3/2)
[Xe]	(5s1/2) (4d3/2) (4d5/2) (5p1/2) (5p3/2)
[Rn]	(6s1/2) (4f5/2) (4f7/2) (5d3/2) (5d5/2) (6p1/2) (6p3/2) (7s1/2) (5f5/2) (5f7/2) (6d3/2) (6d5/2)

In the table each noble gas configuration incorporates the electron states that are stated in the same line and the lines above.

In the `valenceConfig` element a similar list of electron states has to be provided to declare the occupied valence states. Here noble gas configurations are not allowed.

For each of the listed states that is not fully occupied a `stateOccupation` element has to be set. In it the state attribute selects the respective states. The spinUp and spinDown attributes are used to define the number of electrons in the two spin channels.

6.2.17 LDA+U setup

To amend the description of electron correlations in local and semilocal XC functionals, up to 4 Hubbard U parameters can be defined for each species in the atom species section. For this optional `ldaU` XML elements have to be inserted into the respective section below the `energyParameters`, `electronConfig`, and `nocoParams` entries and above the `lo` entries. The following example demonstrates how an `ldaU` element looks like:

```
<ldaU l="2" U="8.0" J="0.9" l_amf="F"/>
```

Attribute	Description
l	The angular momentum quantum number of the orbital for which the U parameter is set.
U	The U parameter in eV.
J	The J parameter in eV.
l_amf	This logical switch determines whether the "around mean field" limit (if true) or the atomic limit (if false) is used.

6.2.17.1 Mixing of the density matrix

Whenever a Hubbard U parameter is added to an atom not only the density has to be part of the mixing from iteration to iteration but the density matrix, too. For this additional parameters can be set in an optional `ldaU` XML element (different from the one above) in the `calculationSetup` section. Such an element looks like:

```
<ldaU l_linMix="F" mixParam="0.05" spinf="1.00"/>
```

Attribute	Description
l_linMix	This switch determines whether a straight mixing algorithm is applied to the density matrix (if true) or the mixing of the density matrix will be performed like the mixing of the density (if false). The switch is optional and set to false by default.
mixParam	This is the optional mixing parameter that is used for the straight mixing of the density matrix. By default this parameter is 0.05.
spinf	Optional, default ist 1.0.

If the `ldaU` element in the `calculationSetup` section is not present all parameters that can be set in it have their default values.

6.2.17.2 Further reading

- The LDA+U method has been developed by Anisimov et al.
- The implementation of LDA+U in Fleur is similar to the one proposed by Shick et al.
- A comparison between the around mean field limit and the atomic limit is available in an article by Petukhov et al.

6.2.18 Non-collinear magnetism setup

To configure a Fleur calculation incorporating non-collinear magnetism, some parameters have to be set in the `calculationSetup` section and further parameters have to be set for each `atomGroup` in the `atomGroups` section. Templates with default parameters are generated by using the input generator with the `-explicit` or `-noco` command line options.

An example for the magnetic input elements in the calculation setup section is:

```
<magnetism jspins="1" l_noco="F" l_ss="F" lflip="F">
   <qss>.0000000000 .0000000000 .0000000000</qss>
   <mtNocoParams l_mperp="F" l_mtNocoPot="F" l_relaxSQA="F" mag_mixing_scheme="0" mix_Rel
   <sourceFreeMag l_sourceFree="F" l_scaleMag="F" mag_scale="1.00000000"/>
</magnetism>
```

The following attributes can/have to be modified here:

Tag	Attribute	Description
magnetism	jspins	Number of spins to consider, i.e. jspins="1" for non-magnetic, and ="2" for magnetic calculations

Tag	Attribute	Description
	l_noco	This boolean switch activated non-collinear calculations
	l_ss	This boolean switch is used to activate spin-spiral calculations.
	lflip	Flip the magnetisation. See section below for details
qss		In case of a spin-spiral calculation you have to give the q-vector of the spiral here
mtNocoParams		In this tag flags controlling the treatment of magnetism in the MT-spheres can be set.
	l_mperp	Here the output of the magnetization perpendicular to the chosen axis can be activated.
	l_mtNocoPot	Switching this to "T" will allow a fully unconstrained calculation in which the MT spin-offdiagonal potential is calculated and use in the SCF. (Needs l_mperp="T")
	l_relaxSQA	Adjust the local spin-quantization axis in the sphere in each iteration to the direction of the local magnetic moment. Needed for some features like LDA+U in a l_mtNocoPot="t" calculation.
	mag_mixing_scheme	In case of l_relaxSQA="T" different preconditioners can be tried out to accelerate the convergence of the magnetic direction. Currently values of 1-3 are implemented
	mix_RelaxWeightOffD	Weight used to increase the local off-diagonal magnetisation in the mag_mixing_scheme not "0" cases. Should be >1 to have an effect.
	l_constrained	Perform a calculation in which an additional constraining field is used to fix the direction of the moments. Typical value "0.5"
	mix_constr	Mixing factor used to determine the constraining field self-consistently
sourceFreeMag		Switches to use a source-free correction to the XC-B-Field. (Experimental)

In addition to these global switches there is the `nocoParams` tag in each `atomGroup` to control magnetic calculations.

```
<nocoParams alpha="Pi/2.0" beta="Pi/4.0"
            l_constrained="F" l_mtNocoPot="F" l_relaxSQA="F"/>
```

The following attributes have to be set here:

Attribute	Description
alpha	The 1st angle that determines the magnetic structure. It is equal to φ in spherical coordinates.
beta	The 2nd angle that determines the magnetic structure. It is equal to ϑ (measured from the z axis) in spherical coordinates.

Attribute	Description
	Please note, that in a l_relaxSQA="T" calculation, these are only used in constructing the starting charge as the correspong angles are determined from the local magnetic moment
l_constrained	These switches overwrite the corresponding global switches (optional).
l_mtNocoPot	These switches overwrite the corresponding global switches (optional).
l_relaxSQA	These switches overwrite the corresponding global switches (optional).

6.2.19 Manipulating the magnetism

If you set the `lfip="T"` switch in the `magnetism` tag, you can manipulate the magnetisation in the spheres. To do so you need to specify a `modInitDen` flag in the `atomGroup` tag. Please be careful about the workflow as you do not want to do this contineously.

```
<modInitDen flipSpinTheta="0.0" flipSpinPhi="0.0" flipSpinScale="F">
```

Attribute	Description
flipSpinTheta	Rotate the magnetisation by this angle around the local "y" axis. This is done first.
spinFlipPhi	Rotate the magnetisation by this angle around local "z" axis. Done after "theta" rotation.
flipSpinScale	If "T" flip the local spin.
beta	The 2nd angle that determines the magnetic structure. It is equal to ϑ (measured from the z axis) in spherical coordinates.
magMom	(Experimental)

6.3 References

Approaches, methods, developments, and investigations in the environment of the FLAPW method and also implementations in the Fleur code have been published in many journal articles. If your publication relies on such work please study it to learn about the details and remember to cite the respective articles. It follows a list of references that may be of relevance here:

- The Fleur code: *www.flapw.de*

- The LAPW method: *O.K. Andersen, Linear methods in band theory, Phys. Rev. B 12, 3060 (1975)*
- Early usage of LAPW: *D. D. Koelling and G. O. Arbman, Use of energy derivative of the radial solution in an augmented plane wave method: application to copper, Phys. F: Metal Phys. 5, 2041 (1975)*
- LAPW for thin films: *H. Krakauer, M. Posternak, and A. J. Freeman, Linearized augmented plane-wave method for the electronic band structure of thin films, Phys. Rev. B 19, 1706 (1979)*
- FLAPW (consideration of the full potential): *E. Wimmer, A. J. Freeman, H. Krakauer, and M. Weinert, Full-potential self-consistent linearized-augmented-plane-wave method for calculating the electronic structure of molecules and surfaces: O2 molecule, Phys. Rev. B 24, 864 (1981)*
- Total energy calculations in FLAPW: *M. Weinert, E. Wimmer, and A. J. Freeman, Total-energy all-electron density functional method for bulk solids and surfaces, Phys. Rev. B 26, 4571 (1982)*
- Comprehensive description of FLAPW: *D. J. Singh and L. Nordström, Planewaves, Pseudopotentials, and the LAPW Method, Springer (2005)*
- FLAPW review: *S. Blügel and G. Bihlmayer, Full-Potential Linearized Augmented Planewave Method, in Computational Nanoscience: Do It Yourself! edited by J. Grotendorst, S. Blügel, and D. Marx, NIC Series Vol. 31, p. 85 (John von Neumann Institute for Computing, Jülich, 2006)*

- Extending the LAPW basis with local orbitals: *D. Singh, Ground-state properties of lanthanum: Treatment of extended-core states, Phys. Rev. B 43, 6388 (1991)*
- Eliminating the linearization error for unoccupied states: *C. Friedrich, A. Schindlmayr, S. Blügel, and T. Kotani, Elimination of the linearization error in GW calculations based on the linearized augmented-plane-wave method, Phys. Rev. B 74, 045104 (2006)*
- Linearization error for valence states: *G. Michalicek, M. Betzinger, C. Friedrich, and S. Blügel, Elimination of the linearization error and improved basis-set convergence within the FLAPW method, Comp. Phys. Commun. 184, 2670 (2013)*

- FLAPW for one-dimensional systems: *Y. Mokrousov, G. Bihlmayer, and S. Blügel, A full-potential linearized augmented planewave method for one-dimensional systems: gold nanowire and iron monowires in a gold tube, Phys. Rev. B. 72, 045402 (2005)*
- Non-collinear magnetism in FLAPW: *Ph. Kurz, F. Foerster, L. Nordström, G. Bihlmayer, and S. Blügel, Ab initio treatment of non-collinear magnets with the full-potential linearized augmented planewave method, Phys. Rev. B 69, 024415 (2004)*

- Spin spirals in FLAPW: *M. Heide, G. Bihlmayer, and S. Blügel, Describing Dzyaloshinskii-Moriya spirals from first principles, Physica B 404, 2678 (2009)*

- Hybrid functionals in FLAPW: *M. Betzinger, C. Friedrich, and S. Blügel, Hybrid functionals within the all-electron FLAPW method: implementation and applications of PBE0, Phys. Rev. B 81, 195117 (2010)*
- HSE hybrid functional in FLAPW: *M. Schlipf, M. Betzinger, C. Friedrich, M. Ležaić, and S. Blügel, HSE hybrid functional within the FLAPW method and its application to GdN, Phys. Rev. B 84, 125142 (2011)*
- Exact exchange in FLAPW: *M. Betzinger, C. Friedrich, S. Blügel, and A. Görling, Local exact exchange potentials within the all-electron FLAPW method and a comparison with pseudopotential results, Phys. Rev. B 83, 045105 (2011)*
- Response functions: optimized effective potential in FLAPW: *M. Betzinger, C. Friedrich, A. Görling, and S. Blügel, Precise response functions in all-electron methods: Application to the optimized-effective-potential approach, Phys. Rev. B 85, 245124 (2012)*
- Response functions: COHSEX, RPA correlation energy in FLAPW: *M. Betzinger, C. Friedrich, A. Görling, and S. Blügel, Precise all-electron dynamical response functions: Application to COHSEX and the RPA correlation energy, Phys. Rev. B 92, 245101*

- Wannier functions within FLAPW: *F. Freimuth, Y. Mokrousov, D. Wortmann, S. Heinze, and S. Blügel, Maximally Localized Wannier Functions within the FLAPW formalism, Phys. Rev. B. 78, 035120 (2008)*

- Green function formulation of the transfer matrix: *D. Wortmann, H. Ishida, and S. Blügel, Ab initio Green-function formulation of the transfer matrix: Application to complex bandstructures, Phys. Rev. B 65, 165103 (2002)*
- Green function embedding for transport through an interface: *D. Wortmann, H. Ishida, and S. Blügel, Embedded Green-function approach to the ballistic electron transport through an interface, Phys. Rev. B 66, 075113 (2002)*

- GW approximation on top of FLAPW: *C. Friedrich, S. Blügel, and A. Schindlmayr, Efficient implementation of the GW approximation within the all-electron FLAPW method, Phys. Rev. B 81, 125102 (2010)*

- Coulomb matrix in FLAPW: *C. Friedrich, S. Blügel, and A. Schindlmayr, Efficient calculation of the Coulomb matrix and its expansion around k=0 within the FLAPW method, Comp. Phys. Comm. 180, 347 (2009)*
- Spin excitations in GW: *E. Şaşıoğlu, A. Schindlmayr, Ch. Friedrich, F. Freimuth, and S. Blügel, Wannier-function approach to spin excitations in solids, Phys. Rev. B 81, 054434 (2010)*
- Calculating Hubbard U with cRPA: *E. Şaşıoğlu, C. Friedrich, and S. Blügel, Effective Coulomb interaction in transition metals from constrained random-phase approximation, Phys. Rev. B 83, 121101(R) (2011)*

- Parallelization and performance optimization: *U. Alekseeva, G. Michalicek, D. Wortmann, and S. Blügel, Hybrid Parallelization and Performance Optimization of the FLEUR Code: New Possibilities for All-Electron Density Functional Theory. In: Aldinucci M., Padovani L., Torquati M. (eds) Euro-Par 2018: Parallel Processing. Euro-Par 2018. Lecture Notes in Computer Science, vol 11014. Springer, Cham.*
- Parallelization of Hybrid functionals calculations with Fleur: *M. Redies, G. Michalicek, J. Bouaziz, C., M. S. Müller, S. Blügel,and D. Wortmann, Fast All-Electron Hybrid Functionals and Their Application to Rare-Earth Iron Garnets, Front. Mater., 21 March 2022*

- The Kerker preconditioner for charge density mixing in FLAPW: *M. Winkelmann, E. Di Napoli, D. Wortmann, and S. Blügel, Kerker mixing scheme for self-consistent muffin-tin based all-electron electronic structure calculations, Phys. Rev. B 102, 195138 (2020)*
- Solving the modified Helmholtz equation in FLAPW: *M. Winkelmann, E. Di Napoli, D. Wortmann, and S. Blügel, Solution to the Modified Helmholtz Equation for Arbitrary Periodic Charge Densities, Front. Phys. 8, 618142 (2021)*