Welcome to the FLEUR-project

This is the homepage of FLEUR, a feature-full, freely available FLAPW (full-potential linearized augmented planewave) code, based on density-functional theory.

The FLAPW-Method is an all-electron method which within density functional theory is universally applicable to all atoms of the periodic table and to systems with compact as well as open structures. It is widely considered to be the most precise electronic structure method in solid state physics.

FLEUR is one of the flagship codes of the MaX-Centre of Excellence. Within MaX we aim at creating a new FLEUR version fit for the challenges of high-throughput and exascale computing.

(http://www.max-centre.eu)

Fleur is part of the juDFT family (http://www.judft.de) of codes developed in Jülich.

Downloading FLEUR (downloads/)

To obtain FLEUR have a look at out download page (downloads/)

FLEUR development team

FLEUR is mainly developed at the Forschungszentrum Jülich at the Institute of Advanced Simulation and the Peter Grünberg Institut.

Impressum (about/)

Downloads of the FLEUR code

Within the MaX project (http://www.max-centre.eu) we created a series of FLEUR-releases which can be downloaded here:

- FLEUR MaX Release 3.1 of Version 0.30 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR3.1.tgz) Current as of 05/09/2019
- FLEUR MaX Release 3 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR3.tgz) Current as of 30/06/2018
- FLEUR MaX Release 2.1 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR2.1.tgz) Current as of 30/11/2017
- FLEUR MaX Release 2 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR2.tgz) Current as of 31/08/2017
- FLEUR MaX Release 1.3 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR1.3.tgz) Current as of 27/06/2017
- FLEUR MaX Release 1.2 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR1.2.tgz) Current as of 04/05/2017
- FLEUR MaX Release 1.1 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR1.1.tgz) Current as of 06/12/2016
- FLEUR MaX Release 1 of Version 0.27 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaXR1.tgz) Current as of 31/08/2016

Accessing the GITLAB

The source code of FLEUR can also be found at the Fleur GitLab (https://iffgit.fz-juelich.de/fleur/fleur). This includes all the versions mentioned above as well as the most recent snapshots and development branches.

More ....

Quantum Mobile -- A virtual machine with all MaX-codes and AiiDA installed can be found on Github (https://github.com/marvel-nccr/quantum-mobile/releases).

There is also a page with a few precompiled binaries (../binaries/).

After downloading the source we strongly recommend to have a look at the Documentation (../Docu-Main/).

Documentation for FLEUR (https://www.flapw.de) build using MkDocs (https://www.mkdocs.org)

Description of the FLEUR codes

The FLEUR code family is a program package for calculating ground-state as well as excited-state properties of solids. It is based on the full-potential linearized augmented-plane-wave (FLAPW) method [1-4]. The strength of the FLEUR code [5,6] lies in applications to bulk, semi-infinite, two- and one-dimensional solids [7], solids of all chemical elements of the periodic table, solids with complex open structures, low symmetry, with complex non-collinear magnetism [8] in combination with spin-orbit interaction [9,10], external electric fields, and the treatment of spin-dependent transport properties [11,12]. It is an all-electron method and thus treats core and valence electrons and can deal with hyperfine properties. The inclusion of local orbitals allows a systematic extension of the LAPW basis that enables a precise treatment of semicore states [13], unoccupied states [14,15], and an elimination of the linearization error in general [16]. A large variety of local and semi-local (GGA) exchange and correlation functionals are implemented, including the LDA+U approach. In recent years the code has been developed further to make contact to electronically complex materials. Hybrid functionals [17,18] and the optimized-effective-potential (OEP) method [15,19] have been implemented. Wannier functions [20] can be constructed to make contact to realistic model Hamiltonians. Excitations can be treated on the basis of the GW approximation [21,22] and ladder diagrams are included to compute spin-wave excitations [23]. The Hubbard U can be calculated in the constrained random phase approximation (cRPA) [24].

Literature:

1. O.K. Andersen, "Linear methods in band theory", Phys. Rev. B 12, 3060 (1975) (http://dx.doi.org/10.1103/PhysRevB.12.3060)
2. D. D. Koelling and G. O. Arbman, Use of energy derivative of the radial solution in an augmented plane wave method: application to copper, J. Phys. F: Metal Phys. 5, 2041 (1975) (http://dx.doi.org/10.1088/0305-4608/5/11/016)
3. 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) (http://dx.doi.org/10.1103/PhysRevB.24.864)
4. 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) (http://dx.doi.org/10.1103/PhysRevB.26.4571)
5. S. Blügel and G. Bihlmayer, " Full-Potential Linearized Augmented Planewave Method (http://www2.fz-juelich.de/nic-series/volume31/bluegel.pdf)", 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)
6. http://www.flapw.de
7. 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) (http://dx.doi.org/10.1103/PhysRevB.72.045402)

8. Ph. Kurz, F. Foerster, L.Nordström, G. Bihlmayer, and S. Blügel, "Ab initio treatment of non-collinear magnets with the full-potential linearized augmente planewave method", Phys. Rev. B 69, 024415 (2004) (http://dx.doi.org/10.1103/PhysRevB.69.024415)
9. M. Heide, G. Bihlmayer, and S. Blügel, "Describing Dzyaloshinskii-Moriya spirals from first principles", Physica B 404, 2678 (2009) (http://dx.doi.org/10.1016/j.physb.2009.06.070)
10. B. Zimmermann, M. Heide, G. Bihlmayer, and S. Blügel, "First-principles analysis of a homochiral cycloidal magnetic structure in a monolayer Cr on W(110)", Phys. Rev. B 90, 115427 (2014) (http://dx.doi.org/10.1103/PhysRevB.90.115427)
11. 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) (http://dx.doi.org/10.1103/PhysRevB.65.165103)
12. 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) (http://dx.doi.org/10.1103/PhysRevB.66.075113)
13. D. Singh, "Ground-state properties of lanthanum: Treatment of extended-core states", Phys. Rev. B 43, 6388 (1991) (http://dx.doi.org/10.1103/PhysRevB.43.6388)
14. 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) (http://dx.doi.org/10.1103/PhysRevB.74.045104)
15. 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) (http://dx.doi.org/10.1103/PhysRevB.83.045105)
16. 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) (http://dx.doi.org/10.1016/j.cpc.2013.07.002)
17. 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) (http://dx.doi.org/10.1103/PhysRevB.81.195117)
18. 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) (http://dx.doi.org/10.1103/PhysRevB.84.125142)
19. 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) (http://dx.doi.org/10.1103/PhysRevB.85.245124)
20. 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) (http://dx.doi.org/10.1103/PhysRevB.78.035120)
21. 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) (http://dx.doi.org/10.1103/PhysRevB.81.125102)
22. 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) (http://dx.doi.org/10.1016/j.cpc.2008.10.009)
23. 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) (http://dx.doi.org/10.1103/PhysRevB.81.054434)

24. 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) (http://dx.doi.org/10.1103/PhysRevB.83.121101)

Features of FLEUR

The FLEUR code allows you to investigate structural, electronic and magnetic properties of periodic systems, in bulk (3D), film (2D) and wire (1D) geometry. Furthermore, it provides the necessary input for the calculation of non-periodic systems (semi-infinite crystals or transport geometries) within the G-Fleur code, or for the calculation of excited state properties.

FLEUR is based on density functional theory (DFT (http://www2.fz-juelich.de/nic-series/volume31/jones.pdf)) and is an implementation of the full-potential linearized augmented planewave (FLAPW (http://www2.fz-juelich.de/nic-series/volume31/bluegel.pdf)) method, which

- is a highly-precise all electron method
- has a basis set equally suited for open and close-packed systems
- is suitable for elements from the whole periodic table

Among other things, FLEUR allows to calculate

- structural and magnetic ground state properties
- electronic properties like bandstructures, densities of states etc.
- charge densities, field gradients, or hyperfine fields

Although FLEUR calculations can be performed for all kinds of materials, it is especially suited for:

- magnetic systems (collinear or non-collinear)
- open systems (surfaces, wires, nanostructures)
- transition metals, lanthanides, actinides

Contacting the FLEUR developers

The FLEUR codes are developed in the group of Stefan Blügel (http://www.fz-juelich.de/pgi/pgi-1/EN) at the Forschungszentrum Juelich (http://www.fz-juelich.de).

User-Support

Please keep in mind that FLEUR is free program package that comes without liability and without support!

However, we strongly encourage the users to participate in the FLEUR-mailing list to discuss their problems and/or experiences. Also this should be more effective than writing emails to individual persons as other users can profit from and participate in discussions on the mailing-list.

To subscribe to the list simply send a mail to fleur-join@fz-juelich.de (mailto://fleur-join@fz-juelich.de).

Please describe your problem as accurate as possible. In particular you might include:

- version of FLEUR
- compiler version, operating system
- inp.xml -file
- error messages
- out-file (please only relevant part, i.e. last couple of lines)

Reporting Bugs

We appreciate if you use the Gitlab Issue system (https://iffgit.fz-juelich.de/fleur/fleur/issues) to report any issues you find while using FLEUR.

Welcome to FLEUR

This is the documentation of the MaX release of FLEUR (https://www.flapw.de/pm/index.php?n=FLEUR.Downloads).

(http://www.max-centre.eu)

- Installation of FLEUR (../Install/) including some hints for configuration.
- Running FLEUR (../Running/) describes how to actually execute a FLEUR calculation.
- Using the input-generator (../inpgen/) to generate the full input out of a simple file.
- Basic calculations (../Basic/) like self-consistence, DOS and bandstructure calculations.
- XML based input-file (../xml-inp/): documentation of the input of FLEUR, its features and hints how to use them.
- More advanced features (../Advanced/) like non-collinear magnetism, SOC, LDA+U.
- The AiiDA interface to FLEUR (http://aiida-fleur.readthedocs.io/en/develop/) can be used to generate, run and store complex workflows.

If you are a more expert user or developer, you might be interested in:

- The Fleur gitlab repository. (https://iffgit.fz-juelich.de/fleur/fleur/)
- Information for developers (../developers/) with the doxygen documentation of the source.
- The doxygen documentation of the source code (https://fleur.iffgit.fz-juelich.de/fleur/html|) You will also find some hints for developing FLEUR there.
- The coverage analysis (https://fleur.iffgit.fz-juelich.de/fleur/coverage_html) of the source code showing which part of the code are covered by the standard tests.
- Discussion of reasons why v27 gives differences (../v26differences/) to v26.
- A Guide/Manual (../developers/) for developers of FLEUR.

Part of the documentation of the Version v0.26 of FLEUR was also made available here (../v26/v26/).

You might also download a pdf-version of the documentation (https://www.flapw.de/site/manual.pdf).

2. Configuration and 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" 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 MailingList (../support/) and 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. Quick guide

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

- 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.
- 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`
- 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.

2.2. 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 (http://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 (http://www.netlib.org) 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 (http://www.openbas.net).
- *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 (http://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 a MPI parallel calculation. This is needed if you can not store the data in memory or want to preprocess the eigenvectors. Please be aware that you need the Fortran-90 interface of the HDF5!
- *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 [[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.

2.3. 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 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 specifies a debugging build.
- CONFIG 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 the output 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.

2.4. 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.

2.4.1. 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.

2.4.2. Adding flags for the compiler

This should be done using the `-flag` option to `configure.sh`. So for example you might want to say `configure.sh -flag "-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 the `-r8`). But additional switches can/should be used.

2.4.2.1. Adding include directories

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

2.4.2.2. Adding linker flags

To add flags to the linker you can do

- 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 ';'.

2.4.3. 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:

1. Versions <13.0 will most probably not work correctly
2. Version 19.0 has issues with the debugging version of FLEUR.

GFortran:

GFortran is knwon to work with versions newer thant 6.3.

PGI:

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

2.4.4. CI-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 -n2 "` to run with two MPI tasks. * You can use the environment variable `juDFT` to give 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 Testing/work. You can check this directory to see why a specific test fails.

Documentation for FLEUR (https://www.flapw.de) build using MkDocs (https://www.mkdocs.org)

3. Running Fleur

Here we deal with the question of how to run FLEUR "by hand". If you are interested in running FLEUR in a scripting environment you might want to check the AiiDA plug-in (http://aiida-fleur.readthedocs.io/en/develop/).

At first you might notice that there are several executables created in the build process. You might find:

- **inpgen**: The input generator (../inpgen/) 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.

In most cases you will first run the input generator (../inpgen/) to create an inp.xml (../xml-inp/).

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

3.1. Command line options

The run-time behaviour of FLEUR can be modified using command line switches. You should understand that 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 (../xml-inp/) file.

!!! note Here we document the most relevant command line options. 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 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.
- `-toXML`: Convert an old **inp**-file into the new inp.xml (../xml-inp/) file.

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. ""Only available in 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 EVP)
- `-diag scalapack`: Use SCALAPACK for parallel EVP.
- `-diag elpa`: Use ELPA for parallel EVP.
- Further options might be available, check `fleur -h` for a list.

3.2. 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.

So you 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.

3.3. Hybrid MPI/OpenMP Parallelization

The efficient usage of FLEUR on modern (super)computers is ensured by 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 OpenMP (Open Multi-Processing) interface or multi-threaded libraries.

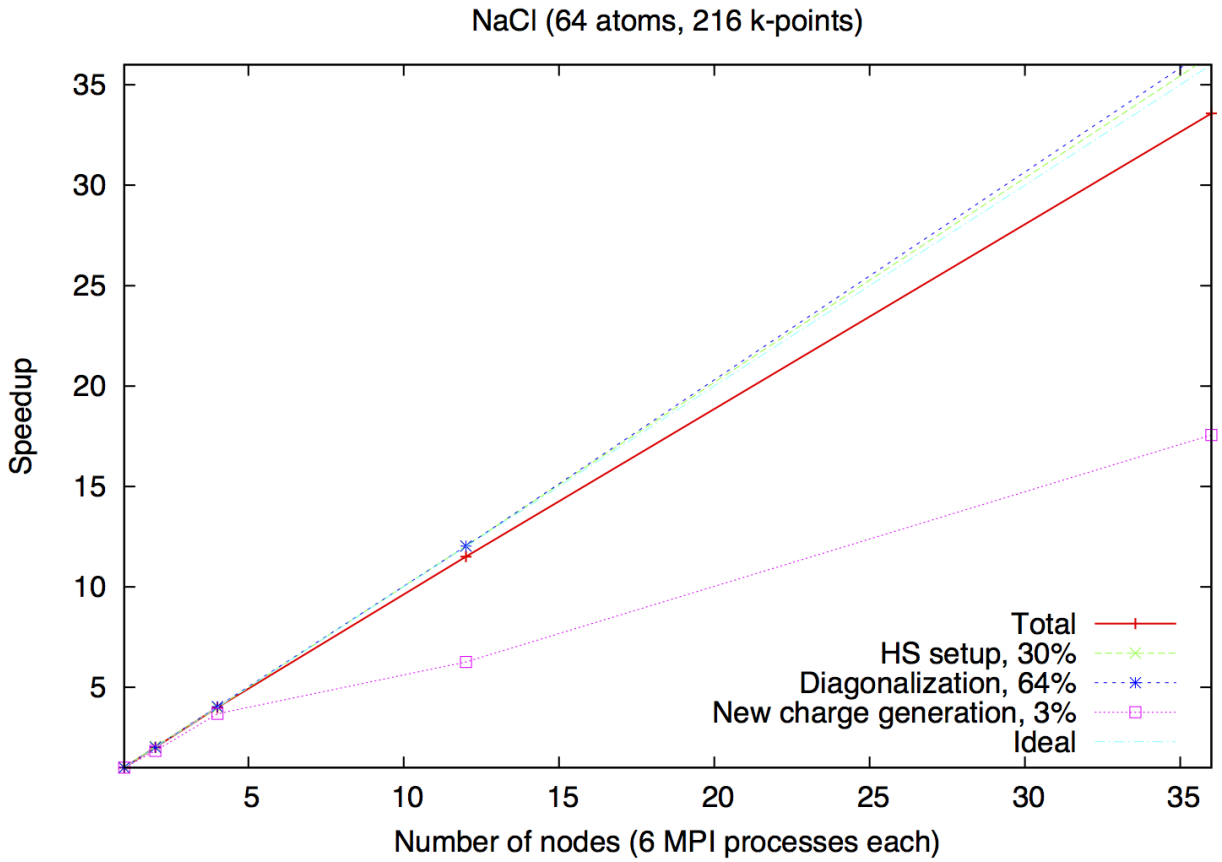
3.3.1. MPI parallelization

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

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

3.3.2. k-point Parallelisation

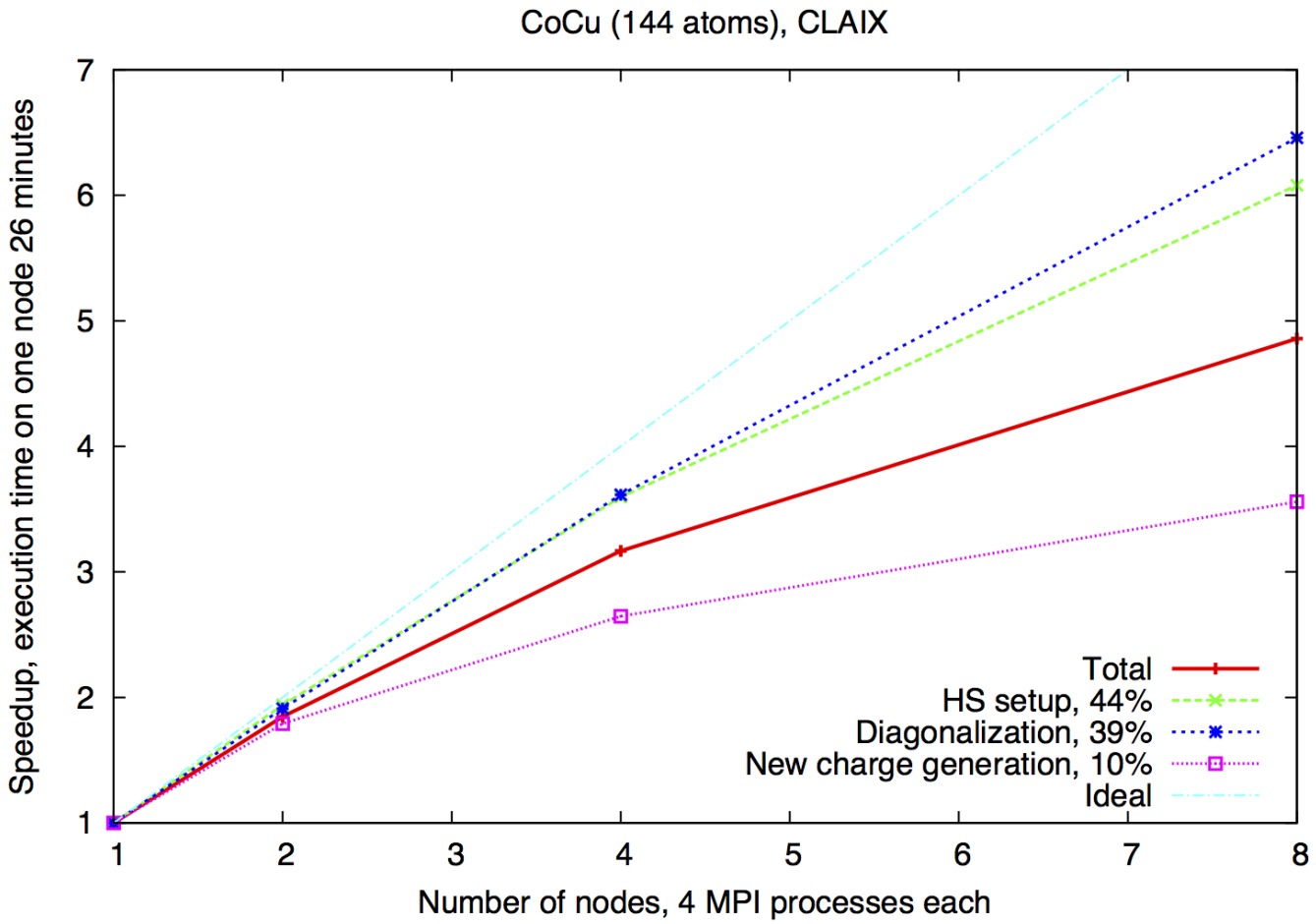
This type of parallelization is always chosen, if the number of k-points (K) is a multiple of the number of MPI processes (P). If K/P is not integer, a mixed parallelization will be attempted and M MPI processes will work on a single k-point, so that K.M/P is integer. This type of parallelization can be very efficient, because all three most time-consuming parts of the code (Hamiltonian matrix setup, diagonalization and generation of the new charge density) are independent for different k-points and there is no need to communicate during the calculation. That is why this type of parallelization is fine, 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. sufficient memory per MPI process to solve a single eigenvalue-problem for a k-point is required. The scaling is good, as long as many k-points are calculated and the potential generation does not get a bottleneck. The saturation of the memory bandwidth might cause the deviation of the speedup from the ideal.



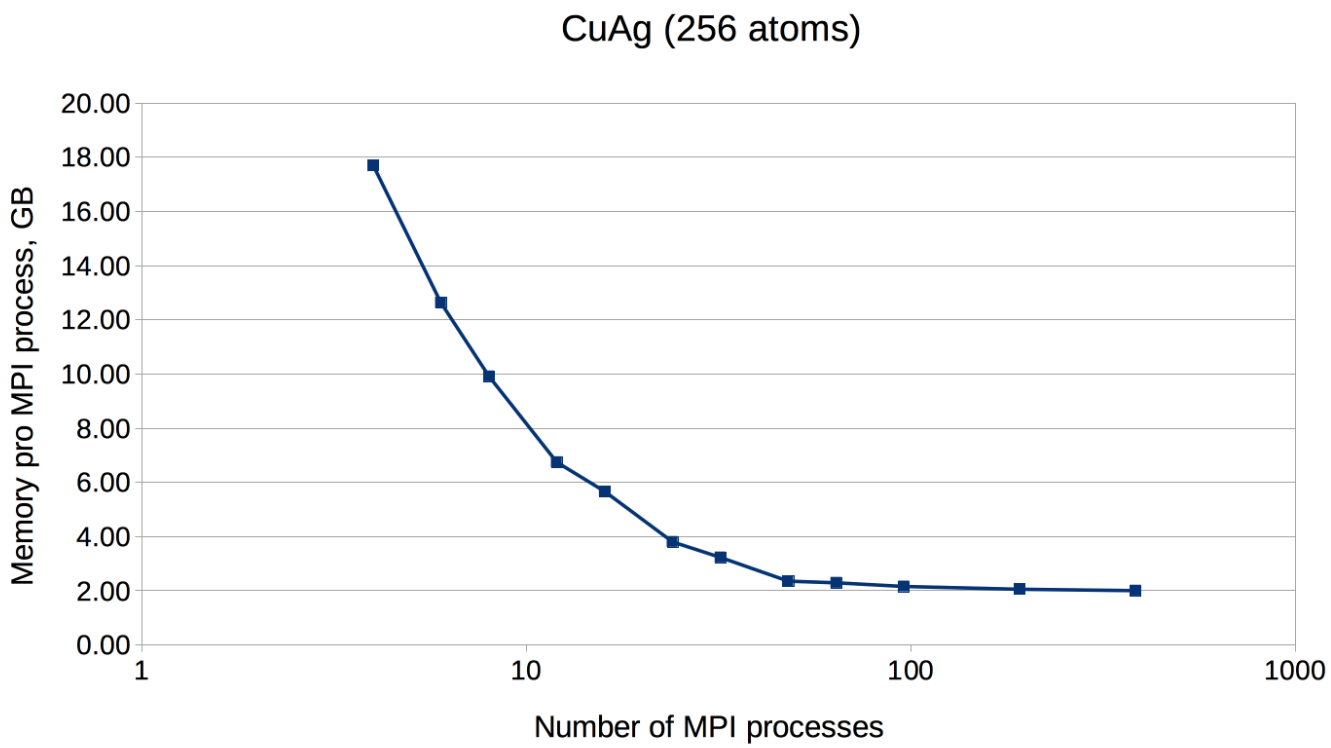
Typical speedup of the k-point parallelization for a small system (NaCl, 64 atoms, 216 k-points) on a computer cluster (Intel E5-2650V4, 2.2 GHz). Execution time of one iteration is 3 hours 53 minutes.

3.3.3. 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 eigen problem on parallel architectures. The FLEUR code contains interfaces to ScaLAPACK, ELPA and Elemental. It is possible to use HDF library if needed.



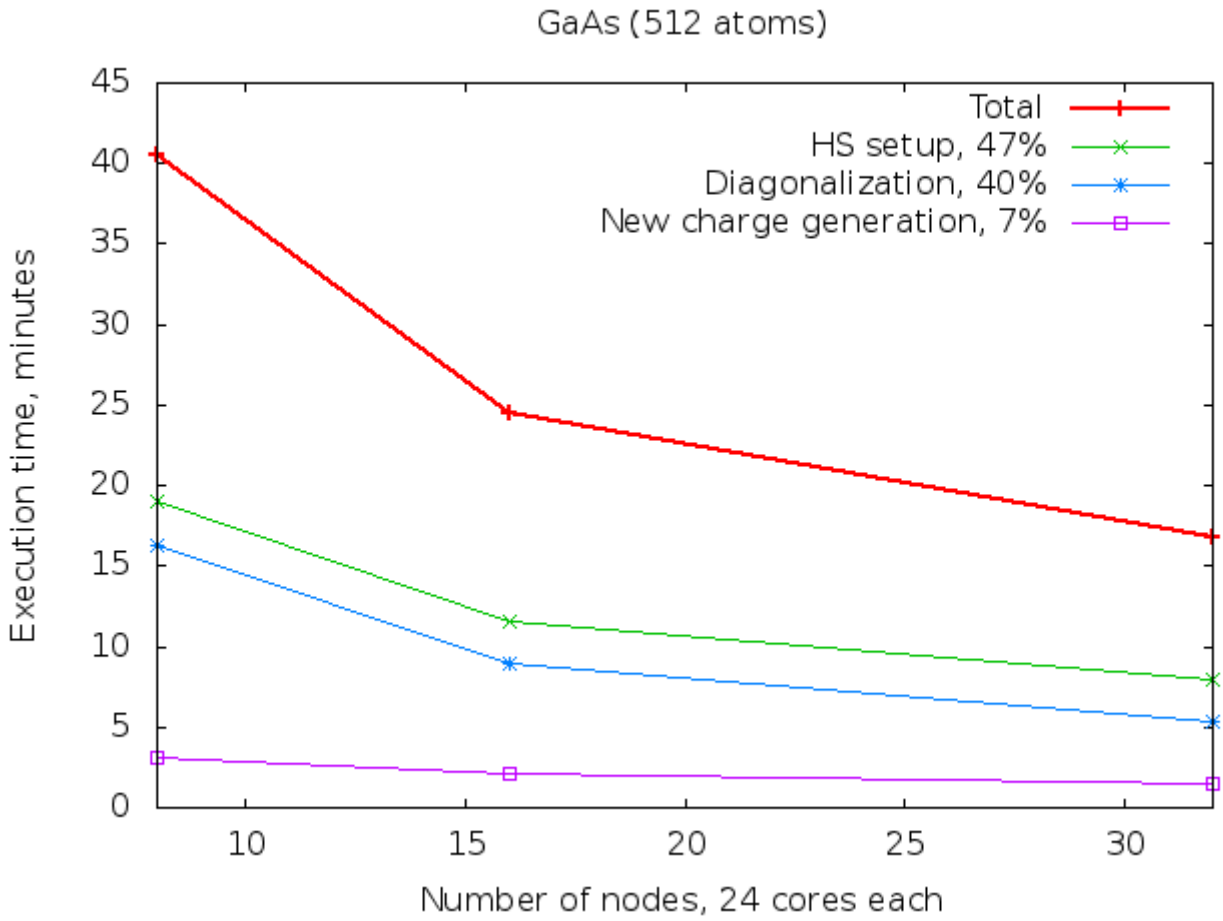
Example of eigenvector parallelization of a calculation with 144 atoms on the CLAIX (Intel E5-2650V4, 2.2 GHz).



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)

3.3.4. 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 (that means MPI parallelization) also inside the node, but in this case the memory available for every MPI process will be considerably smaller. Imagine you use a node with 24 cores and 120 GB memory. If you start one MPI process it will get all 120 GB, two will only get 60 GB and so on, if you start 24 MPI processes, only 5 GB memory will be available for each of them. The intra-node parallelism can be utilized more efficiently when using shared-memory programming paradigm, for example OpenMP interface. In the FLEUR code the hybrid MPI/OpenMP parallelization is realised either by directly implementing OpenMP pragmas or by usage of multi-threaded libraries. If you want to profit from this type of parallelization, you would need ELPA and multithreaded MKL library.



Timing measurements of the GaAs system (512 atoms) on the CLAIX supercomputer (Intel E5-2650V4, 2.2 GHz, 24 cores per node, 128 GB per node).

3.3.5. 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 does one need, how many MPI processes per node, how many threads per MPI process. First of all, if your system contains several k-point, choose the number of MPI processes accordingly. If the number of k-points (K) is a multiple of the number of MPI processes (P) than every MPI procces will work on a given k-point alone. If K/P is not integer, a mixed parallelization will be attempted and M MPI processes will work on a single k-point, so that K.M/P is integer. That means for example, that if you have 48 k-points in your system, it is not a good idea to start 47 MPI processes.

The next question is: how many nodes do I need? That depends strongly on the size of the unit cell you simulating and the memory size of the node you are simulating on. In the table below you can find some numbers from our experience on a commodity Intel cluster with 120 GB and 24 cores per node - if your unit cell (and hardware you use) is similar to what is shown there, it can be a good start point. The two numbers in the "# nodes"-column show the range from the "minimum needed" to the "still reasonable". Note that our test systems have only one k-point. If your simulation crashed with the run-out-of-memory-message, try to double your requested resources (after having checked that ulimit -s is set unlimited, of course ;)). The recommended number of MPI processes per node can be found in the next column. As for the number of OpenMP threads,

on the Intel architecture it is usually a good idea to fill the node node with the threads (i.e. if the node consist of 24 cores and you start 4 MPI processes, you spawn each to 6 threads), but not to use the hyper-threading.

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

And last but not least - if you use the node exclusively, bind your processes and check your environment. If the processes are allowed to vagabond through the node (which is usually default), the performance can be severely damaged.

4. The FLEUR input generator

The basic input of FLEUR is the inp.xml file (../xml-inp/). 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 up by hand. Hence, an input-file generator is provided.

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

- the symmetry information
- the atom types and the equivalent atoms
- muffin-tin radii, l-cutoffs and radial grid parameters for the atom types
- plane-wave cutoffs (kmax,gmax,gmaxxc).
- (in film calculations) the vacuum distance and d-tilda
- 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 (../xml-inp/) 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.

4.1. Running inpgen

To call the input generator you tyically do

```
inpgen <simple_file
```

!!! warning Please note that the program expects its 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
`-explicit`	Some input data that is typically not directly provided in the inp.xml file is now generated. This includes a list of k points and a list of symmetry operations.

4.2. Basic input

Your input should contain (in this order):

- (a) A title
- (b) Optionally: input switches (whether your atomic positions are in internal or scaled Cartesian units)
- (c) Lattice information (either a Bravais-matrix or a lattice type and the required constants (see below); in a.u.)
- (d) Atom information (an identifier (maybe the nuclear number) and the positions (in internal or scaled Cartesian coordinates)).
- (e) 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.
- (f) Optionally: Preset parameters (Atoms/General)

4.2.1. Title

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

4.2.2. Input switches

The namelist input should start with an &input and end with a / . 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')
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

4.2.3. An example (including the title):

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

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

4.2.4. Lattice information

There are two possibilities to input 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(|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	cub	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)

full name	No	short	other_possible_names	Description	Variants
simple-tetragonal	6	tet	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	trg	hR2,r2,R2	hexagonal-R2
base-centered-orthorhombic2	17	orA	oA	orthorhombic-A (centering on A)
base-centered-orthorhombic3	18	orB	oB	orthorhombic-B (centering on B)
centered-monoclinic2	19	moA	mA	monoclinic-A (centering on A)
centered-monoclinic3	20	moB	mB	monoclinic-B (centering on B)

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

4.2.5. 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 logical `cartesian` (see above). For supercells, sometimes more natural to input positions in scaled Cartesian.

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).

4.2.6. Ending an input file

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

`&end /`

4.2.7. Special cases

4.2.7.1. Film calculations

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

lattice	**description**
square	primitive tetragonal
primitive rectangular	primitive orthorhombic

lattice	description
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.

4.2.7.2. 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 q is only consistent with symmetry elements that operate on a plane perpendicular to 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 q or the SQA, e.g.:

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

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

```
&soc 1.5708 0.0 /
```

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

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

4.2.8. Preset parameters

4.2.8.1. 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. All parameters available are

parameter	description
id=[atomic identification number]	identifies the atom you wish to modify.
z=[charge number]	specifies the charge number of the atom.

parameter	description
rmt=[muffin-tin radius]	specifies a muffin-tin radius for the atom to modify.
dx=[log increment]	specifies the logarithmic increment of the radial mesh for the atom to modify.
jri=[# mesh points]	specifies the number of mesh points of the radial mesh for the atom to modify.
lmax=[spherical angular momentum]	specifies the maximal spherical angular momentum of the atom to modify.
lnonsph=[nonspherical angular momentum]	specifies the maximal angular momentum up to which non-spherical parts are included to quantities of the atom to modify.
ncst=[number of core state]	specifies the number of states you wish to include in the core of the atom to modify.
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 of the atom to modify.
lo=[list of local orbitals]	specifies, which states shall be treated as local orbitals. This is a string.
element=[name of the element]	identifies the atom to modify by its element name. This is a string. You must specify this.

4.2.8.2. General

You also might 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 input file for the input file generator of 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. XML based input file

!!! Note "Advice for first time readers" This section is more of reference character. You might want to skip it first and come back later.

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' (../inpgen/) but FLEUR workflows require adjustments to this file.

6.1. Example inp.xml file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<fleurInput fleurInputVersion="0.27">
   <comment>
      alpha Si
   </comment>
   <calculationSetup>
      <cutoffs Kmax="3.70" Gmax="11.00" GmaxXC="9.20" numbands="0"/>
      <scfLoop itmax="9" maxIterBroyd="99" imix="Anderson" alpha=".05"
preconditioning_param="0.0" spinf="2.00"/>
      <coreElectrons ctail="T" frcor="F" kcrel="0" coretail_lmax="0" />
      <magnetism jspins="1" l_noco="F" l_J="F" swsp="F" lflip="F"/>
      <soc theta=".00" phi=".00" l_soc="F" spav="F" off="F" soc66="T"/>
      <expertModes gw="0" pot8="F" eig66="F" lpr="0" isec1="99" secvar="F"/>
      <geometryOptimization l_f="F" qfix="0" />
      <bzIntegration valenceElectrons="20.00" mode="hist" fermiSmearingEnergy=".0010">
         <kPointCount count="16" gamma="F"/>
      </bzIntegration>
      <energyParameterLimits ellow="-1.80" elup="1.00"/>
   </calculationSetup>
   <cell>
      <symmetryFile filename="sym.out"/>
      <bulkLattice scale="1.00" latnam="any">
         <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>
   </cell>
   <xcFunctional name="pbe" relativisticCorrections="F"/>
   <atomSpecies>
      <species name="Si-1" element="Si" atomicNumber="14" coreStates="2" magMom=".00"
flipSpin="T">
         <mtSphere radius="2.18" gridPoints="721" logIncrement=".016"/>
         <atomicCutoffs lmax="8" lnonsphr="6"/>
         <energyParameters s="3" p="3" d="3" f="4"/>
         <lo type="SCLO" l="1" n="2" eDeriv="0"/>
      </species>
   </atomSpecies>
   <atomGroups>
      <atomGroup species="Si-1">
         <relPos>1.000/8.000 1.000/8.000 1.000/8.000</relPos>
         <relPos>-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" vacdos="F" slice="F">
      <checks vchk="F" cdinf="F" disp="F"/>
      <densityOfStates ndir="0" minEnergy="-.50" maxEnergy=".50" sigma=".0150"/>
      <vacuumDOS layers="0" integ="F" star="F" nstars="0" locx1=".00" locy1=".00"
locx2=".00" locy2=".00" nstm="0" tworkf=".00"/>
```

```
      <unfoldingBand unfoldBand="F" supercellX="1" supercellY="1" supercellZ="1"/>
      <plotting iplot="F" score="F" plplot="F"/>
      <chargeDensitySlicing numkpt="0" minEigenval=".00" maxEigenval=".00" nnne="0"
pallst="F"/>
      <specialOutput form66="F" eonly="F" bmt="F"/>
   </output>
</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. Comment and Constants section

```
   <comment>
      alpha Si
   </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.

6.3. Calculation Setup section

```
   <calculationSetup>
      <cutoffs Kmax="3.70" Gmax="11.00" GmaxXC="9.20" numbands="0"/>
      <scfLoop itmax="9" maxIterBroyd="99" imix="Anderson" alpha=".05"
preconditioning_param="0.0" spinf="2.00"/>
      <coreElectrons ctail="T" frcor="F" kcrel="0"/>
      <magnetism jspins="1" l_noco="F" l_J="F" swsp="F" lflip="F"/>
      <soc theta=".00" phi=".00" l_soc="F" spav="F" off="F" soc66="T"/>
      <expertModes gw="0" pot8="F" eig66="F" lpr="0" isec1="99" secvar="F"/>
      <geometryOptimization l_f="F" qfix="0"/>
      <bzIntegration valenceElectrons="20.00" mode="hist" fermiSmearingEnergy=".0010">
         <kPointCount count="16" gamma="F"/>
      </bzIntegration>
      <energyParameterLimits ellow="-1.80" elup="1.00"/>
   </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
	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
	imix	The mixing scheme. This can be one of "straight", "Broyden1", "Broyden2", and "Anderson"
	alpha	The mixing factor
	preconditioning_param	The preconditioning parameter for bulk metals. Typical value: 0.8. Choose higher mixing parameter, f.e. 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
	l_J	Set this to true to calculate Heisenberg J_ij parameters.
	swsp	If true generate spin-polarized from unpolarized density.

Tag	Attribute	Description
	lflip	If true flip spin directions for each atom with set flipSpin flag.
soc		Parameters needed for calculations with spin-orbit coupling
	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.
	off	Only soc contributions from certain muffin tins are considered.
	soc66	This is only relevant for spin-orbit calculations with set eig66 flag.
nocoParams		See the section on the non-collinear magnetism setup for details.
expertModes		Parameters for the control of certain advanced Fleur calculation modes
	gw	The different output modes for GW approximation calculations are set here.
	pot8	Set this to true to use the potential from the files pottot and potcoul.
	eig66	If true and the eig file exists read the eigenvalues and eigenvectors from it. If true and the eig file does not exist create it and stop.
	lpr	If lpr is greater than 0, also list eigenvectors in the out file
	isec1	After iteration isec the iterative diagonalization is used.
	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.
	qfix	This parameter determines the calls to the qfix routine that ensures charge neutrality. qfix=0 restricts the number of calls to the routine. This should be fine in general. qfix=1 calls the routine more often. This is the old FLEUR behaviour. For further options on relaxation..

Tag	Attribute	Description
ldaU		Optional parameters for the density matrix mixing in LDA+U calculations. The LDA+U setup. for details.
bzIntegration		Parameters required for the Brillouin zone integration
	valenceElectrons	The number of electrons to be represented within the valence electron framework.
	mode	The Brillouin zone integration mode. It can be one of 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.
kPointCount		See the page on the k point set setup for details.
energyParameterLimits		Boundaries for energy parameters
	ellow	
	elup	

6.4. Unit cell section

```
<cell>
   <symmetryFile filename="sym.out"/>
   <bulkLattice scale="1.0000000000" latnam="any">
      <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>
</cell>
```

The unit cell section covers the declaration of the symmetry operations available in the unit cell and the definition of the lattice vectors.

Definition of the Bravais lattice

In the XML input file lattices for bulk or film unit cells can be defined in the unit 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" latnam="any">
   <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. If the Bravais matrix is directly provided, the bulkLattice attribute latnam has to be set to "any".

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 can be given by up to two vacuumEnergyParameters elements for the two vacua. 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" latnam="squ" dVac="5.79" dTilda="9.68">
   <a1>5.458864500000</a1>
   <vacuumEnergyParameters vacuum="1" spinUp="-.25" spinDown="-.25"/>
</filmLattice>
```

!!! warning "Alternative specification of lattice" While we currently still support some of the options outlined here, these are deprecated. Please use them in the input of the input-generator 'inpgen' (../inpgen/) instead of 'inp.xml'

```
<bulkLattice scale="0.97" latnam="squ">
   <a1>4.815397</a1>
   <c>6.81</c>
</bulkLattice>
```

Another way to define the lattice is to set the latnam attribute to the name of the Bravais lattice. In this case instead of the Bravais matrix the corresponding lattice parameters have to be provided. They are set in the `<a1>`, `<a2>`, and `<c>` xml elements. Note that not all lattices require all three lattice constants. In such a case only the required parameters have to be set. The lattices definable by this approach are

latnam	`<a1>`	`<a2>`	`<c>`	**description**
squ	x		x	
c-b	x		x	
fcc	x		*	face-centered cubic
hex	x		x	
hx3	x		x	
bcc	x		*	body-centered cubic
c-r	x	x	x	
p-r	x	x	x	
obl	**	**	x	

- * an arbitrary value is required but ignored
- ** the two rows of a 2D Bravais matrix without an enclosing bravaisMatrix element have to be provided

The declaration of film lattices is illustrated with the following examples:

```
      <filmLattice scale="1.00" latnam="any" dVac="47.66" dTilda="51.37">
         <bravaisMatrix>
            <row-1>6.0157233797 .0000000000 .0000000000</row-1>
            <row-2>.0000000000 6.0157233797 .0000000000</row-2>
            <row-3>.0000000000 .0000000000 48.1100636580</row-3>
         </bravaisMatrix>
         <vacuumEnergyParameters vacuum="1" spinUp="-.25" spinDown="-.25"/>
      </filmLattice>
```

Of course, film lattices can also be defined by naming the type of 2D lattice and providing the lattice constants. In comparison to the bulk lattice the lattice constant does not have to be provided as it is given by dVac.

6.5. Declaration of the unit cell symmetry in the XML input

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

!!! warning "Adjusting the symmetry" For certain calculations you might 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 it might be desired to provide the symmetry operations in a separate file. You can use the x-include option for this purpose.

!!! warning "Deprecated options to specify the symmetry" The options below are still in frequent use but should be considered as deprecated. We will remove them in future.

Symmetries in an external sym.out file

```
   <symmetryFile filename="sym.out"/>
```

By providing the symmetryFile element the symmetry operations are read in from an external file. Typically this is the sym.out file written out by the input file generator. It is, of course, possible to change the filename with the associated attribute. **Explicit specification of space-groups**

```
      <symmetry spgrp="p4m" invs="T" zrfs="T"/>
```

With the XML element symmetry it is possible to define the symmetries by providing one of the 2D space groups in the attribute spgrp and additionally providing information about the availability of inversion symmetry in invs and z reflection symmetry in zrfs. The applicable 2D space groups are (where the angles denote the number of centers for corresponding rotations):

name	lattice	180°	120°	90°	60°	reflection axes	glide reflections
p1	oblique	-	-	-	-	-	-
p2	oblique	4	-	-	-	-	-
pmy							
pgy							
cmy							
pmm	rectangular	4	-	-	-	4 (in 2 perp. directions)	-
pmg	rectangular	2	-	-	-	2 (parallel)	2 (parallel, perp. to refl. axes)
pgg	rectangular	2	-	-	-	-	4 (in 2 perp. directions)
cmm	rhombic	3	-	-	-	2 (in 2 perp. directions)	4 (in 2 perp. directions)
p4	square	2	-	2	-	-	-
p4m	square	2	-	2	-	6 (2 horizontal, 2 vertical, 2 diagonal)	4 (in 2 perp. directions, not on refl. axes)
p4g	square	2	-	2	-	4 (2 per diagonal)	6 (in 4 directions, not on refl. axes)
p3	hexagonal	-	3	-	-	-	-
p3m1	hexagonal	-	3	-	-	5 (in 3 directions)	8 (in 3 directions, in middle between refl. axes)
p31m	hexagonal	-	3	-	-	3 (in 3 directions)	4 (in 3 directions, in middle between refl. axes)
p6	hexagonal	3	2	-	1	-	-
p6m	hexagonal	3	2	-	1	8 (in 6 directions)	12 (in 6 directions, in middle between refl. axes)
pm	rectangular	-	-	-	-	2 (parallel)	-
pg	rectangular	-	-	-	-	-	2 (parallel)
cm	rhombic	-	-	-	-	2 (parallel)	2 (parallel to, in middle between refl. axes)

6.6. Setup of the k point set

To define the k point set used for the calculation three alternatives are provided. Additionally the XML input provides a way of defining the k point path for band structure calculations. The different means of setting the k point set are selected by providing one of three possible XML elements in the bzIntegration part of the calculation setup section. The following examples illustrate each of these elements:

```
        <kPointCount count="100" gamma="F"/>
```

By using the kPointCount element one only defines a number of k points that Fleur should use. Fleur then generates a default k point mesh that features approximately the desired number of k points. Some advanced orbital dependent exchange correlation functionals require k point sets that include the gamma point. For such calculations the gamma attribute can be used to ensure that the gamma point is included in the k point set, even if this is not provided by the default mesh.

```
        <kPointMesh nx="12" ny="12" nz="12" gamma="F"/>
```

If more control over the k point mesh is needed the kPointMesh element can be used. Here the number of k points in each dimension (nx, ny, nz) can be set directly. In analogy to the kPointCount element kPointMesh also provides the above discussed gamma attribute.

```
         <kPointList posScale="48.00000000" weightScale="288.00000000" count="24">
            <kPoint weight="    8.000000">   20.000000    20.000000    21.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000    12.000000    21.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000     4.000000    21.000000</
kPoint>
            <kPoint weight="    8.000000">   12.000000    12.000000    21.000000</
kPoint>
            <kPoint weight="   16.000000">   12.000000     4.000000    21.000000</
kPoint>
            <kPoint weight="    8.000000">    4.000000     4.000000    21.000000</
kPoint>
            <kPoint weight="    8.000000">   20.000000    20.000000    15.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000    12.000000    15.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000     4.000000    15.000000</
kPoint>
            <kPoint weight="    8.000000">   12.000000    12.000000    15.000000</
kPoint>
            <kPoint weight="   16.000000">   12.000000     4.000000    15.000000</
kPoint>
            <kPoint weight="    8.000000">    4.000000     4.000000    15.000000</
kPoint>
            <kPoint weight="    8.000000">   20.000000    20.000000     9.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000    12.000000     9.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000     4.000000     9.000000</
kPoint>
            <kPoint weight="    8.000000">   12.000000    12.000000     9.000000</
kPoint>
            <kPoint weight="   16.000000">   12.000000     4.000000     9.000000</
kPoint>
            <kPoint weight="    8.000000">    4.000000     4.000000     9.000000</
kPoint>
            <kPoint weight="    8.000000">   20.000000    20.000000     3.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000    12.000000     3.000000</
kPoint>
            <kPoint weight="   16.000000">   20.000000     4.000000     3.000000</
kPoint>
            <kPoint weight="    8.000000">   12.000000    12.000000     3.000000</
kPoint>
            <kPoint weight="   16.000000">   12.000000     4.000000     3.000000</
kPoint>
            <kPoint weight="    8.000000">    4.000000     4.000000     3.000000</
kPoint>
         </kPointList>
```

If the input file generator is used with the -explicit command line switch it generates a kPointList that directly shows each k point and the number of k points.

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.

The weightScale and count attributes of kPointList only have informative character. The here provided values are not used by Fleur but only calculated by the input file generator. weightScale is the sum of the weights of all provided k points. count is the number of the provided k points.

6.6.1. The k point path for band structure calculations

For band structure calculations only the kPointCount element should be used. Fleur then generates a k point path along several default high-symmetry points that consists exactly of the number of k points provided in the count attribute of kPointCount.

If a special k point path deviating from the default one should be used this can be achieved by defining several special k points within the kPointCount element:

```
        <kPointCount count="100" gamma="F">
           <specialPoint name="g">0.0 0.0 0.0</specialPoint>
           <specialPoint name="X">0.5 0.0 0.0</specialPoint>
        </kPointCount>
```

The two or more specialPoints defined in this way replace the high-symmetry k points in the generation of the k point path. Again for each specialPoint the three coordinates in the Brillouin zone have to be provided. Additionally a name should be set with the associated attribute.

6.7. Exchange correlation functional section

```
   <xcFunctional name="pbe" relativisticCorrections="F"/>
```

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 (http://dx.doi.org/10.1103/PhysRevB.23.5048)
vwn	The functional by Vosko, Wilk, and Nusair (http://www.nrcresearchpress.com/doi/abs/10.1139/p80-159)
bh	The functional by Barth and Hedin (http://dx.doi.org/10.1088/0022-3719/5/13/012)

GGA functionals	
pw91	The functional by Perdew and Wang (http://dx.doi.org/10.1103/PhysRevB.45.13244)
pbe	The functional by Perdew, Burke, and Ernzerhof (http://dx.doi.org/10.1103/PhysRevLett.77.3865)
rpbe	The revPBE functional by Zhang and Yang (http://dx.doi.org/10.1103/PhysRevLett.80.890)
Rpbe	The RPBE functional by Hammer, Hansen, and Nørskov (http://dx.doi.org/10.1103/PhysRevB.59.7413)
wc	The functional by Wu and Cohen (http://dx.doi.org/10.1103/PhysRevB.73.235116)

Tag	Attribute	Description
	relativisticCorrections	A boolean switch used to activate optional relativistic corrections according to MacDonnald-Vosko (http://dx.doi.org/10.1088/0022-3719/12/15/007).

6.8. Atom species

The atom species 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 atom groups section each atom group is associated to one of the species.

```
   <atomSpecies>
      <species name="Si-1" element="Si" atomicNumber="14" coreStates="2" magMom=".00"
flipSpin="T">
         <mtSphere radius="2.18" gridPoints="721" logIncrement=".016"/>
         <atomicCutoffs lmax="8" lnonsphr="6"/>
         <energyParameters s="3" p="3" d="3" f="4"/>
         <lo type="SCLO" l="1" n="2" eDeriv="0"/>
      </species>
   </atomSpecies>
```

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.
	coreStates	The number of states whose electrons are to be treated as core electrons.
	magMom	The magnetic moment of each atom in the unit cell belonging to this species.
	flipSpin	A boolean switch to flip the spin direction for the associated atoms.

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 (http://www.sciencedirect.com/science/article/pii/S003810989900577 3?via%3Dihub) 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 (http://dx.doi.org/10.1103/PhysRevB.64.195134).
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 specie. 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.
nocoParams		See the description of the non-collinear magnetism 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.
lo		For a more detailed description have a look at [[xmlLDAUSetup This element is used to introduce local orbitals to each of the associated atoms. It can be present multiple times. See the page on the [[xmlLOSetup
	type	The type of the LO. This can be SCLO for semicore LOs or HELO for LOs at higher energies

Tag	Attribute	Description
	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.

6.8.1. 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 atom species 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 semicore 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 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. (http://dx.doi.org/10.1103/PhysRevB.43.6388)
- In the context of GW calculations local orbitals employing higher energy derivatives have been introduced by Friedrich et al. (http://dx.doi.org/10.1103/PhysRevB.74.045104)
- For the representation of unoccupied states local orbitals on the basis of the HELO -(l+1) criterion have been used by Betzinger et al. (http://dx.doi.org/10.1103/PhysRevB.83.045105)
- 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. (http://dx.doi.org/10.1016/j.cpc.2013.07.002)

6.8.2. Setup of the electron configurations

Within each species in the atom species 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. Furthermore, occupations for the different electron states are 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="1.00000000"
spinDown="1.00000000"/>
         </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.

6.9. Atom groups section

```
   <atomGroups>
      <atomGroup species="Si-1">
         <relPos>1.000/8.000 1.000/8.000 1.000/8.000</relPos>
         <relPos>-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.

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.

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:

6.9.1. Relative positions

```
      <atomGroup species="W-1">
         <relPos>.0000000000 .5000000000 .0600000000</relPos>
         <relPos>.5000000000 .0000000000 -.0600000000</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 XML element 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.

6.9.2. Film positions

```
      <atomGroup species="W-2">
         <filmPos>1.0000000000/2.0000000000 1.0000000000/2.0000000000 -12.0314467594</
filmPos>
         <filmPos>1.0000000000/2.0000000000 1.0000000000/2.0000000000 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).

6.10. Output section

```
   <output dos="F" band="F" vacdos="F" slice="F">

      <checks vchk="F" cdinf="F" disp="F"/>
      <densityOfStates ndir="0" minEnergy="-.50" maxEnergy=".50" sigma=".015"/>
      <vacuumDOS layers="0" integ="F" star="F" nstars="0" locx1=".00" locy1=".00"
locx2=".00" locy2=".00" nstm="0" tworkf=".00"/>
      <unfoldingBand unfoldBand="F" supercellX="1" supercellY="1" supercellZ="1"/>
      <plotting iplot="F" score="F" plplot="F"/>
      <chargeDensitySlicing numkpt="0" minEigenval=".00" maxEigenval=".00" nnne="0"
pallst="F"/>
      <specialOutput form66="F" eonly="F" bmt="F"/>
   </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.
	vacdos	A boolean switch that determines whether a vacuum DOS has to be calculated.
	slice	A boolean switch controlling whether a slice has to be calculated. The associated parameters are set in the chargeDensitySlicing element.
	coreSpec	A boolean switch controlling whether a core spectrum has to be calculated. The associated parameters are set in the coreSpectrum 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.
densityOfStates		Parameters for DOS calculations are set in this element.
	ndir	Select the DOS calculation mode. For details have a look at the respective page for the conventional input file.
	minEnergy	The lower energy boundary of the window for the DOS plot in Htr.
	maxEnergy	The upper energy boundary of the window for the DOS plot in Htr.
	sigma	The Gaussian smearing factor used in the plot whenever the tetrahedron method is not used.

Tag	Attribute	Description
vacuumDOS		
	layers	The number of layers in which the vacuum DOS is integrated. The value can be between 1 and 99.
	integ	If integ is true the vacuum DOS is also integrated in the direction normal to the film.
	star	if star is true, star coefficients are calculated at values of izlay for 0 (=q) to nstars-1.
	nstars	The number of star functions to be used (0th star is given by value of q=charge integrated in 2D)
	locx1, locy1,	The four attributes loc[xy]1, loc[xy]2 are used to calculate a local DOS at a certain vertical z position (or integrated in z).
	locx2, locy2	The local DOS is restricted to an area in the 2D unit cell which is defined by the corner points given by loc[xy]1 and loc[xy]2. The two corners have to be provided in internal coordinates.
	nstm	For the consideration of STM: 0: s-Tip, 1: p_z-Tip, 2: d_z^2-Tip (following Chen's derivative rule)
	tworkf	For the consideration of STM: Workfunction of Tip (in Hartree units) is needed for d_z^2-Orbital.
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. The region is defined by the plot_inp file.
	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.
specialOutput		
	form66	Use this switch to write out a formatted eigenvector file.

Tag	Attribute	Description
	eonly	This switch can be used to prevent prevent the output of eigenvectors into associated file.
	bmt	This switch is used to generate a charge density file 'cdnbmt' with suppressed magnetization in the interstitial and vacuum regions.
coreSpectrum		See the section on the core spectrum setup for details.

6.11. Setup of non-collinear magnetism in the XML input file

To configure a Fleur calculation incorporating non-collinear magnetism, some parameters have to be set in the calculation setup section and further parameters have to be set for each atomGroup in the atom groups section. In each of these sections nocoParams elements have to be added. Templates with default parameters are generated by using the input generator with the -explicit command line option.

An example for the nocoParams element in the calculation setup section is:

```
      <nocoParams l_ss="F" l_mperp="F" l_constr="F" l_disp="F" sso_opt="FFF" mix_b=".
00000000" thetaJ=".00000000" nsh="0">
         <qss>.0000000000 .0000000000 .0000000000</qss>
      </nocoParams>
```

The following attributes have to be set here:

Attribute	Description
l_ss	This boolean switch is used to activate spin-spiral calculations.
l_mperp	Here the output of the magnetization perpendicular to the chosen axis can be activated.
l_constr	Switch this on to constrain the magnetic moments.
l_disp	This is the dispersion switch. If l_disp is set to true, the Force theorem is used to calculate the sum of eigenvalues for each predefined qss.
sso_opt	This attribute sets three logical switches used in the context of [[SsoDetails
mix_b	This is a mixing factor. If l_constr is set to true then the constraint field is mixed. In this case mix_b="0.5" should work fine. In the case of an atom with l_relax being set to true the input/output-directions of the moments are mixed. Here you can choose mix_b > 1 (e.g. 4).
thetaJ	Used for the [[HeisenbergInteractionParametersJij
nsh	Used for the calculation of Heisenberg J_ij interaction coefficients. This is the number of shells of neighbors to be considered.

The enclosed XML element is used to define the spin spiral vector in reciprocal lattice vectors.

An example for the nocoParams element in each atomGroup element of the atom groups section is:

```
        <nocoParams l_relax="F" l_magn="F" M=".0000000000" alpha=".0000000000"
beta=".0000000000" b_cons_x=".0000000000" b_cons_y=".0000000000"/>
```

The following attributes have to be set here:

Attribute	Description
l_relax	This has to be set to true to relax the directions of the magnetic moments at the associated atoms.
l_magn	Used for the calculation of Heisenberg J_ij interaction coefficients. Set this to true iff the atoms in the group are magnetic.
M	Used for the calculation of Heisenberg J_ij interaction coefficients. The value of the magnetic moment(including sign) has to be set here.
alpha	The 1st angle that determines the magnetic structure. It is equal to "phi" in spherical coordinates.
beta	The 2nd angle that determines the magnetic structure. It is equal to "theta" (measured from the z axis) in spherical coordinates.
b_cons_x,b_cons_y	These are the constraint fields in x and y direction. They are determined self-consistently if l_constr is set to true.

6.12. Setup of LDA+U calculations

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.

6.12.1. 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 calculation setup 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 XML element in the calculation setup section is not present all parameters that can be set in it have their default values.

6.12.2. Further reading

- The LDA+U method has been developed by Anisimov et al. (http://dx.doi.org/10.1088/0953-8984/9/4/002)
- The implementation of LDA+U in Fleur is similar to the one proposed by Shick et al. (https://doi.org/10.1103/PhysRevB.60.10763)
- A comparison between the around mean field limit and the atomic limit is available in an article by Petukhov et al. (https://doi.org/10.1103/PhysRevB.67.153106)

6.13. Using the force-theorem in FLEUR calculations

Fleur has the option to calculate some properties using the force-theorem. In such calculations the last self-consistency iteration (or the only iteration if itmax=1) is replaced by a loop in which the eigenvalue sum for different configurations at a fixed potential are calculated.

This is controlled by a special section in the inp.xml file to be inserted after the output-section:

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

Here several different modes are possible. Exactly one should be present:

6.13.1. Magnetic Anisotropy Energy MAE

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

This is a simple mode to estimate the magnetocyrstalline anisotropy energy. A loop over different spin-quantization directions is performed. A list of angles should be given. Please note that the number of "theta" and "phi" angles must of course be the same.

6.13.2. Spin-Spiral Dispersion

```
 <spinSpiralDispersion>
     <q> 0.0 0.0 0.0 </q>
     <q> 0.1 0.0  0.0 </q>
</spinSpiralDispersion>
```

This is a simple mode to calculate a spin-spiral dispersion for several q-values. Please note that the first q-Vector given will overwrite the q-vector specified in the '''qss'''-tag given in the '''nocoParams'''.

6.13.3. Dzyaloshinskii Moriya Interaction

```
 <DMI theta="0.0 0.1*Pi" phi="0.0 0.2*Pi" >
    <qVectors>
        <q> 0.0 0.0 0.0 </q>
        <q> 0.1 0.0  0.0 </q>
    </qVectors>
</DMI>
```

This mode is actually slightly different from the modes above as it actually does not only calculate the eigenvalue sum for different setups, but also employs first order perturbation theory to estimate the effect of spin-orbit coupling on a spin-spiral calculation. Hence you can specify here a list of q-vectors as well as different angles for the magnetisation.

6.14. Structure relaxations with FLEUR

!!! warning "WORK in progress" ATTENTION: this section describes work in progress. Your version might not support it yet.

General notes:

1. Structural relaxations should be performed using the HDF5-version of the code to enable better handling of charge densities after a relaxation step
2. You should be aware that good forces in LAPW are only obtained if you perform very accurate calculations. In particular you should:
    - Use high cutoffs for your LAPW-basis set (high `Kmax` and high `lmax`).
    - Use the core-tail correction `ctail=T` possibly checking the influence of `coretail_lmax`.
    - Use LOs for semi-core states as tails or core states can give contributions to forces otherwise not covered.

6.14.1. Switching on the calculation of forces

To calculate forces on an Atom use the ''force'' tag in the ''atomgroup'' tag.

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

For each atom you specify if forces are calculated and which of the directions should be used. Please be aware that setting this tag alone only enables the calculation of the contributions to the force simple to obtain. The full force including the Pulay terms are calculated by setting `l_f="T"`.

6.14.1.1. Relaxing the structure

To calculate all forces (including Pulay terms) and to perform a structure relaxation you have to specify the ''geometryOptimization'' tag in the input:

```
<geometryOptimization l_f="F" epsdisp="0.001" epsforce="0.001" forcemix="2"
forcealpha="1.0" qfix="0" force_converged="0.00001" />
```

All attributes except the `l_f` are optional with defaults as specified above.

Attribute	Description
l_f	This switches on the calculation of the full force including the more expensive to obtain Pulay terms
epsdisp	This is the minimal displacement at which the relaxation is considered converged
espforce	The minimal force at which the relaxation is considered converged
forcemix	The scheme to use for calculating displacements **forcemix=0** Simple relaxation by moving the atoms in the direction of the force (forcealpha gives the corresponding scaling factor) **forcemix=1** A CG scheme for relaxations **forcemix>1** A BFGS scheme to determine new displacements.
forcealpha	This is the scaling factor used to shift the atoms in the ''forcemix=0'' case or in the case of no history.
qfix	The qfix parameter determines how to deal with non-charge neutral densities. These occur if you want to reuse the charge after new displacements have been calculated. In addition, this parameter determines the number of times the code checks of charge neutrality in a self-consistency run. The logic is as follow: **even number** leads to less frequent checks. This should be OK in general. **odd number** leads to frequent checks as in older FLEUR versions **numbers 0/1** will just fix the charge after a displacement **numbers 2/3** will fix the charge by adjusting the interstitial charge only. This is a better approximation if your charge was generated by a displaced configuration. **numbers 4/5** will try to use an additional decomposition of the charge to get a better estimate for displaced positions (experimental)
force_converged	the value given here determines a criterion if an actual relaxation step is performed. Only if the maximal change of force between two SCF iterations is below this value a structural relaxation step is done.

6.14.2. Output of new coordinates

If a relaxation step is performed the code stops with the message of either `Structual relaxation: new displacements generated` or `Structual relaxation: Done` if the forces or displacements are smaller than `epsforce` or `epsdisp`, respectively.

In addition a file `relax.xml` is generated. This file contains the current displacements and the history of the positions and forces from previous relaxation steps.

6.14.3. Shifting positions

During structural relaxation the atomic positions will update. These updates should be preformed by specifying additional tags in the inp.xml file. FLEUR creates such a tag-structure in the `relax.xml` file which looks similar to the one given here:

```
<relaxation>
   <displacements>
    <displace>   0.0000000000    0.0000000000   -0.0122134660 </displace>
    <displace>   0.0000000000    0.0000000000    0.0031814755 </displace>
    <displace>   0.0000000000    0.0000000000   -0.0044667051 </displace>
    <displace>   0.0000000000    0.0000000000    0.0179966687 </displace>
  </displacements>
   <relaxation-history>
    <step energy="   -22005.9436915333">
      <posforce>   0.0000000000    0.0000000000    0.5000000000    0.0000000000
0.0000000000   -0.0111067330 </posforce>
      <posforce>   0.5000000000    0.5000000000    0.0350000000    0.0000000000
0.0000000000    0.0030907378 </posforce>
      <posforce>   0.5000000000    0.0000000000    0.1130000000    0.0000000000
0.0000000000   -0.0042333525 </posforce>
      <posforce>   0.5000000000    0.5000000000   -0.3960000000    0.0000000000
0.0000000000    0.0189983343 </posforce>
    </step>
   </relaxation-history>
 </relaxation>
```

You should include this file into `inp.xml` by incorporating a line like

```
<xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="relax.xml"> <xi:fallback/
> </xi:include>
```

just before the closing `</fleurInput>` tag. This is done automatically by the input-generator now. The empty fallback enables you to use this line even if no `relax.xml` is present.

!!! warning "Spheres might overlap during relaxation" The code will check that no overlapping MT-spheres arise from your displacements and it will decrease them if necessary. Hence you have to make sure that no such warning for overlapping MT-spheres still remain if your structure is considered converged or you might be stuck with a configuration limited by the size of your MT-spheres.

6.15. External fields

6.15.1. Magnetic fields

!!! note * these are simple Zeeman terms added 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.

There are two options to apply B-Fields:

1. You can specify an overall B-field in the fields-tag of the calculation setup:

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

1. If you want to have a field only applied within the MT-sphere of a single atom (old mfee-file) you should use the tag in the species definition:

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

6.15.2. Electic field settings

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 subtracting 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" ...
```

1. More complex settings are possible using the `<fields>` tag:

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

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_1/2` for charges on the upper and one for the lower plate (default 0.0). Setting these to different values enables to place an asymmetric field. * the `autocomp` switch makes sure that overall charge neutrality is automatically calculated. * the `dirichlet` switch enables use of Dirichlet boundary conditions. * the `eV` 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.

Since version 0.26b inhomogeneous fields can 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 places using the in which stings 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 {$A' = A L_z$}, where {L_z} is the number of points in ''z'' direction. Contrary to **circ** and **rect**, charges are mask out without being redistributed to non-masked positions. It is {$\int_0^{2\pi} A|\sin(x)| {\rm d}x = 4A'$}, **n** is the order and **delta** ({$ \delta$}) 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 **3*k1d** 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 temporary 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
```

6.16. Setup of core spectrum calculations for EELS in the XML input file

In the output section it can be specified that a core spectrum has to be calculated. For this the attribute coreSpec has to be set to "T" and in the output section the optional xml element coreSpectrum has to be defined analogously 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 (http://monalisa.phys.washington.edu/PAPERS/dissertations/thesis_jorissen.pdf)), 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
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.

6.17. Using the x-include feature

The xml-parser of FLEUR allows you to use the XInclude functionality of XML. By specifying a line like

```
<xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="relax.xml"> <xi:fallback/
> </xi:include>
```

you can include additional files into 'inp.xml' (in this case the 'relax.xml' file. An additional fallback can be specified if this file is not present (here an empty string).

Glossary

Here we will describe a few terms often used in the context of FLEUR calculations

atomic units

Almost all input and output in the FLEUR code is given in atomic units, with the exception of the U and J parameters for the LDA+U method in the input-file and the bandstructure and the DOS output-files where the energy unit is eV.

energy units: 1 Hartree (htr) = 2 Rydberg (Ry) = 27.21 electron volt (eV)
length units: 1 bohr (a.u.) = 0.529177 Ångström = 0.0529177 nm
electron mass, charge and Planks constant h / 2 π (*ħ*) are unity
speed of light = e'^2^'/*ħ* 1/ α ; fine-structure constant α: 1/α = 137.036

band gap

The band-gap printed in the output ([[out]] file) of the FLEUR code is the energy separation between the highest occupied Kohn-Sham eigenvalue and the lowest unoccupied one. Generally this value differs from the physical band-gap, or the optical band-gap, due to the fact that Kohn-Sham eigenvalues are in a strict sense Lagrange multipliers and not quasiparticle energies (see e.g. Perdew & Levy, PRL 51, 1884 (1983) (http://dx.doi.org/10.1103/PhysRevLett.51.1884)).

core levels

States, which are localized near the nucleus and show no or negligible dispersion can be treated in an atomic-like fashion. These core levels are excluded from the valence electrons and not described by the FLAPW basisfunctions. Nevertheless, their charge is determined at every iteration by solving a Dirac equation for the actual potential. Either a radially symmetric Dirac equation is solved (one for spin-up, one for spin-down) or, if @@kcrel=1@@ in the input file, even a magnetic version (cylindrical symmetry) is solved.

distance (charge density)

In an iteration of the self consistency cycle, from a given input charge density, ρ'^in^', a output density, ρ'^out^', is calculated. As a measure, how different these two densities are, the distance of charge densities (short: distance, d) is calculated. It is defined as the integral over the unit cell: {$ d = \int || \rho^{in} - \rho^{out} || d \vec r $}\ and gives an estimate, whether self-consistency is approached or not. Typically, values of 0.001 milli-electron per unit volume (a.u.'^3^') are small enough to ensure that most properties have converged. You can find this

value in the out-file, e.g. by @@grep dist out@@. In spin-polarized calculations, distances for the charge- and spin-density are provided, for non-Collinear magnetism calculations even three components exists. Likewise, in an LDA+U calculation a distance of the density matrices is given.

energy parameters

To construct the FLAPW basisfunctions such, that only the relevant (valence) electrons are included (and not, e.g. 1s, 2s, 2p for a 3d-metal) we need to specify the energy range of interest. Depending slightly on the shape of the potential and the muffin-tin radius, each energy corresponds to a certain principal quantum number "n" for a given "l". E.g. if for a 3d transition metal all energy parameters are set to the Fermi-level, the basis functions should describe the valence electrons 4s, 4p, and 3d. Also for the vacuum region we define energy parameters, if more than one principal quantum number per "l" is needed, local orbitals can be specified.

Fermi level

In a calculation, this is the energy of the highest occupied eigenvalue (or, sometimes it can also be the lowest unoccupied eigenvalue, depending on the "thermal broadening", i.e. numerical issues). In a bulk calculation, this energy is given relative to the average value of the interstitial potential; in a film or wire calculation, it is relative to the vacuum zero.

interstitial region

Every part of the unit cell that does not belong to the
muffin-tin spheres and not to the vacuum region. Here, the basis (charge density, potential) is described as 3D planewaves.

lattice harmonics

Symmetrized spherical harmonics. According to the point group of the atom, only certain linear combinations of spherical harmonics are possible. A list of these combinations can be found at the initial section of the out-file.

local orbitals

To describe states outside the valence energy window, it is recommended to use local orbitals. This can be useful for lower-lying semicore-states, as well as unoccupied states (note, however, that this just enlarges the basis-set and does not cure DFT problems with unoccupied states).

magnetic moment

The magnetic (spin) moment can be defined as difference between "spin-up" and "spin-down" charge, either in the entire unit cell, or in the muffin-tin spheres. Both quantities can be found in the out-file, the latter one explicitly marked by " --> mm", the former has to be calculated from the charge analysis (at the end of this file). \ The orbital moments are found next to the spin-

moments, when SOC is included in the calculation. They are only well defined in the muffin-tin spheres as {$ m_{orb} = \mu_B \sum_i < \phi_i | r \times v | \phi_i > $}.\ The in a collinear calculation, the spin-direction without SOC is arbitrary, but assumed to be in z-direction. With SOC, it is in the direction of the specified spin-quantization axis. The orbital moment is projected on this axis. In a non-collinear calculation, the spin-directions are given explicitely in the input-file.

muffin-tin sphere

Spherical region around an atom. The muffin-tin radius is an important input parameter. The basis inside the muffin-tin sphere is described in spherical harmonics times a radial function. This radial function is given numerically on a logarithmic grid. The charge density and potential here are also described by a radial function times a the lattice harmonics.

FLEUR tutorials

Online Tutorials (../online-tutorials/)

Tutorial to DFT lecture 2018 (../DFT-lecture-tutorial-2018/)

Tutorial to Fleur workshop 2019 (picking flowers) (../fleur-workshop-tutorial-2019/)

Developing FLEUR

The development effort for FLEUR is mainly hosted at the Institute Quantum Theory of Materials @Forschungszentrum Juelich Germany (https://www.fz-juelich.de/pgi/pgi-1/EN).

GITLAB

The development process is performed using gitlab. You can access the main gitlab page here (http://iffgit.fz-juelich.de/fleur/fleur).

If you checkout the code please be aware that there are several branches.

- The release branch contains the code of the last release published on the FLEUR webpage. You can not push to this branch directly.
- You probably want to use the development branch to insert your changes.
- If your changes are large, it might be a good idea to create your own branch first.

The changes you push to the gitlab will be tested by our CI directly: (https://iffgit.fz-juelich.de/fleur/fleur/pipelines).

Doxygen

We use doxygen to create the documentation of the source. This can be found here (https://fleur.iffgit.fz-juelich.de/fleur/doxygen).

Coverage

The automatic tests of FLEUR cover only part of the source. Here you find the analysis (https://fleur.iffgit.fz-juelich.de/fleur/coverage_html).

Further information

Some more information for developers are collected here (../developers/).

Contributors guide

Everyone is very welcome to contribute to the enhancement of FLEUR. Please use the [gitlab service] (https://iffgit.fz-juelich.de/fleur/fleur) to obtain the latest development version of FLEUR.

Coding rules for FLEUR:

In no particular order we try to collect items to consider when writing code for FLEUR

- Instead of 'stop' use calls to judft_error, judft_warn, judft_end
- Do not read and write any files. Files are neither replacements for common-blocks nor a storage for status variables. Only exceptions:
- you create nice IO subroutines in the subdirectory io
- you write to the typical FLEUR output files

Useful info for developers

Using fleur with the HDF5 library and debugging it with valgrind

HDF5 has to be built with the same compiler that is also used to compile fleur. If adapted the following commands can be used to compile a HDF5 library for fleur:

- 'FC=/usr/local/intel/impi/4.0.3.008/intel64/bin/mpiifort CC=/usr/local/intel/impi/4.0.3.008/intel64/bin/mpiicc CXX=/usr/local/intel/impi/4.0.3.008/intel64/bin/mpiicc ./configure --enable-fortran --enable-fortran2003 --enable-parallel --enable-using-memchecker --enable-clear-file-buffers'
- 'make'
- 'make install'
- 'make check' (optional)

Note:

- The paths have to be adjusted such that that compiler is used which is also used to compile fleur.
- The parallel switch is not needed for every calculation: Only for parallel calculations in which HDF5 is also used for the eigenvector IO.
- The last two command line switches in the configure command turn on initializations of irrelevant array parts in HDF5. If valgrind is not needed it is probably the better choice to leave them away. If left away valgrind will complain about missing initializations in the HDF5 library.
- valgrind gives partially strange behavior if used together with the intel compiler. It would be better to use it together with gfortran.

- At the moment HDF5 is needed in version 1.8.. *Usage of version 1.10.* yields some problems.

Furthermore to configure and start fleur with HDF5 the following has to be done:

- In your .bashrc the HDF5 library has to be added to the LD_LIBRARY_PATH. This implies a line like 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/hdf5/current/hdf5/lib'
- configure fleur with some line like 'CMAKE_Fortran_FLAGS="-I~/hdf5/current/hdf5/include" FLEUR_LIBRARIES="-L~/hdf5/current/hdf5/lib;-lhdf5_fortran;-lhdf5" ./fleur/configure.sh IFF'