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 4 of Version 0.30 (http://www.flapw.de/pm/uploads/FLEUR/fleurMaX4.tgz) Current as of 28/11/2019
- 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.md).

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.md) including some hints for configuration.
- Running FLEUR (Running.md) 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.md): documentation of the input of FLEUR, its features and hints how to use them.
- More advanced features (Advanced.md) 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).

The new documentation (documentation.md) is under development.

4. The FLEUR input generator

The basic input of FLEUR is the inp.xml file (xml-inp.md). 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.md) 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)
simple-tetragonal	6	tet	tP, st	tetragonal-P	
body-centered-tetragonal	7	bct	tI, bct	tetragonal-I	

full name	No	short	other_possible_names	Description	Variants
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
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.
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.

parameter	description
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.

parameter	description
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 /
```

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/)

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

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'