TBKOSTER

Tight-Binding Magnetic Molecular Dynamics for everyone

C. Barreteau, P. Thibaudeau*

Release 0.0.3
February 10, 2026

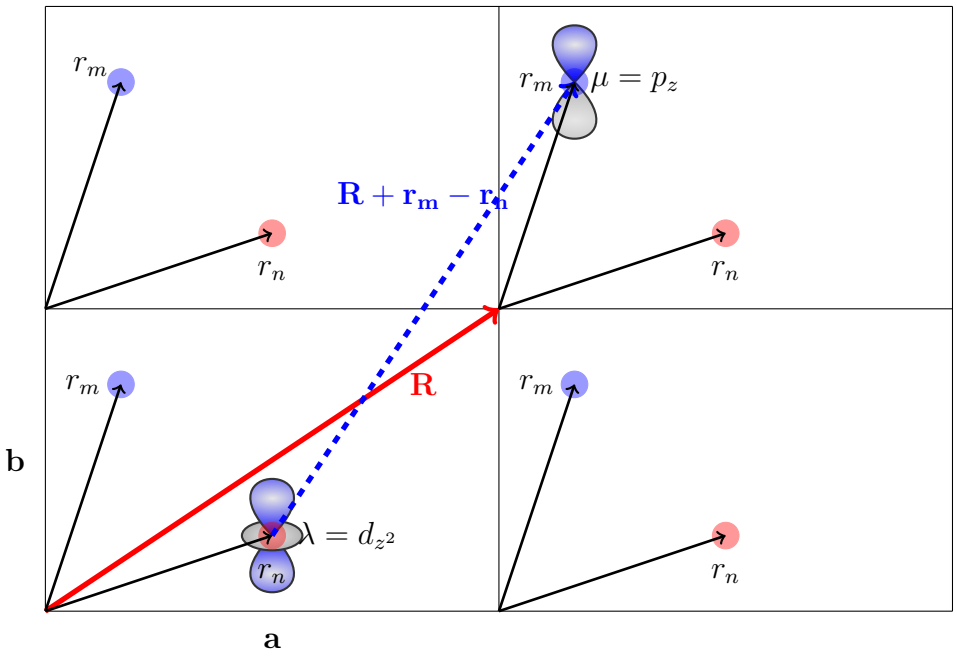
Abstract

Contents

*Supported by CEA

Preface

In this manual we describe the TBKOSTERcode.



The manual contains:

• Background theory for magnetic Tight-Binding with several atoms per unit-cell, periodic boundary conditions and overlaps.

• Installation guide

• Instructions to run the code

• TBKOSTERprocess and post-process options.

• Description of input and output files.

CHAPTER 1

Background theory

• **General TB formalism** with several atoms per unit-cell, non-collinear magnetism, overlap integrals and periodic boundary conditions is described thoroughly in file TB.pdf.

• **NRL TB** Hamiltonian is described in the review paper: *An efficient magnetic tight-binding method for transition metals and alloys* C. Barreteau, D. Spanjaard, MC Desjonqueres Comptes Rendus Physique **17** 406-429 (2016).

• **Magnetic Force Theorem (MFT)** and its application to the calculation of magneto-crystalline anisotropy can be found in: *Magnetocrystalline anisotropy energy of Fe(001) and Fe(110) slabs and nanoclusters: A detailed local analysis within a tight-binding model* D. Li, A. Smogunov, C. Barreteau, F. Ducastelle, and D. Spanjaard Phys. Rev. B **88** 214413 (2013).

• **TB spin dynamics** is described in *Spin dynamics from a constrained magnetic tight-binding model* Ramon Cardias, Cyrille Barreteau, Pascal Thibaudeau, and Chu Chun Fu Phys. Rev. B **103**, 235436 (2021).

CHAPTER 2

Installation

The first step is to download the latest release of TBKOSTER from its github repository. To proceed, you have to check that git is installed :

```
$ locate git
```

Then, clone the distant repository to your local one :

```
$ git clone https://github.com/araven/TBKOSTER.git
```

In order to build this documentation, a LaTeX distribution is mandatory, including some external packages such as pdflatex, bibtex and htlatex.

2.1 Linux

Preferred method : on Ubuntu 16.04 and later, install gfortran and cmake to compile TBKOSTER. Be sure to get the mandatory related dependencies of these packages.

```
$ sudo apt install cmake gfortran clang doxygen graphviz \
libblas-dev liblapack-dev libomp-dev texlive-latex-base \
texlive-latex-extra tex4ht
```

Be sure to update to cmake release 3.9 or higher. In order to build the code, to the root of TBKOSTER directory :

```
$ if ! test -d linux; then mkdir linux; fi
$ cd linux
$ cmake ..
$ make
```

To get access to the OpenMP implementation, then update your packages with openmp support.

If you want to use Intel compiler, MKL and Intel Lapack libraries, you have to tell this to cmake as, in sequential :

```
$ BLA_VENDOR=Intel10_64lp_seq FC=ifort cmake ..
```

Be sure that the variable MKLROOT is set accordingly.

In order to get good numerical performance, you have to produce a Release version as :

```
$ cmake -DCMAKE_BUILD_TYPE=Release ..
```

You can combine all these options. If you prefer to prepare an installation with a given installed Lapack library and gfortran try :

```
$ BLA_VENDOR=OpenBLAS FC=gfortran cmake ..
```

2.2 MacOS

Preferred method : on MacOS 10.11 and later, install the cmake, lapack and gfortran with llvm support, with the ports subsystem (http://www.macports.org)

```
$ sudo port install cmake gcc6 libgcc6 gcc_select \
llvm-3.9 llvm_select lapack libomp
```

In order to build the code, to the root of TBKOSTER directory :

```
$ if ! test -d macos; then mkdir macos; fi
$ cd macos
$ cmake ..
$ make
```

For both Linux and MacOS platform, you can invoke cmake with Release or Debug option in order to deploy these releases. Simply try

```
$ cmake -DCMAKE_BUILD_TYPE=Release/Debug ..; make
```

To get access to the OpenMP implementation, then update your ports with openmp package. You can easily change your settings with

```
$ port select --summary
$ port select --set llvm mp-llvm-3.9
$ port select --set gcc mp-gcc6
```

In order to get good numerical performance, you may use the OpenBLAS library and produce a Release version as :

```
$ port install openblas
$ BLA_VENDOR=OpenBLAS cmake -DCMAKE_BUILD_TYPE=Release ..
```

2.3 MS Windows

For Windows earlier than 10 release 1709, you have to consider the following method: download and follow the instructions to install MSYS2 software distro and building platform (http://www.msys2.org/). Open an MSYS console and first upgrade the whole system :

```
$ pacman -Syu
```

get the compilation toolchain :

```
$ pacman -S mingw-w64-x86_64-toolchain
```

get the gfortran compiler and lapack library :

```
$ pacman -S mingw-w64-x86_64-gcc-libgfortran
$ pacman -S mingw-w64-x86_64-openblas
```

get the cmake program to control the software compilation process using simple platform and compiler independent configuration files, and to generate native makefiles and workspaces that can be used in the compiler environment of your choice :

```
$ pacman -S cmake
```

In order to build the code, open a MSYS MINGW 64-bit console. To the root of TBKOSTER directory :

```
$ if ! test -d win; then mkdir win; fi
$ cd win
$ cmake -G"MinGW Makefiles" -DCMAKE_SH="CMAKE_SH-NOTFOUND" ..
$ mingw32-make
```

In order to run TBKOSTER.exe, be sure you have the TERM and TERMINFO environment variables up to date into your .bashrc file :

```
$ export TERM=xterm
$ export TERMINFO=/c/Program Files/msys/mingw64/share/terminfo
```

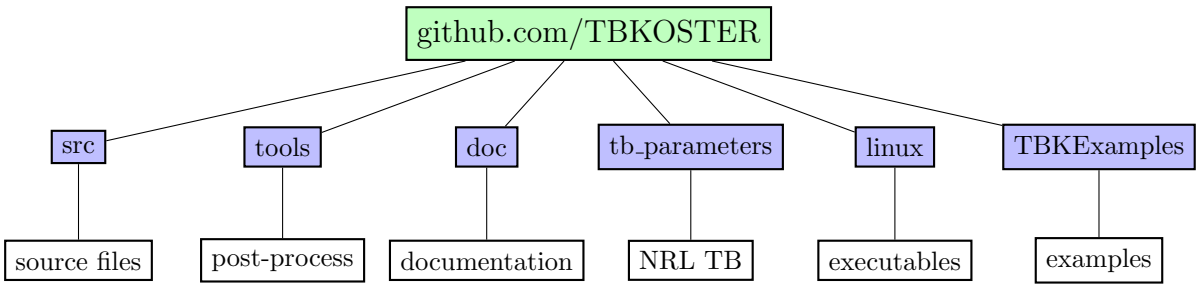
No OpenMP implementation for MS Windows has been tested.

For Windows higher than 10 release 1709, you simply have to activate the optional Windows SubSystem for Linux, download the Ubuntu Package from the Microsoft Marketplace and follow the instructions of the Linux section of this manual.

CHAPTER 3

Running the code

3.1 Directories



3.2 linux directory

The "linux" directory should contain...

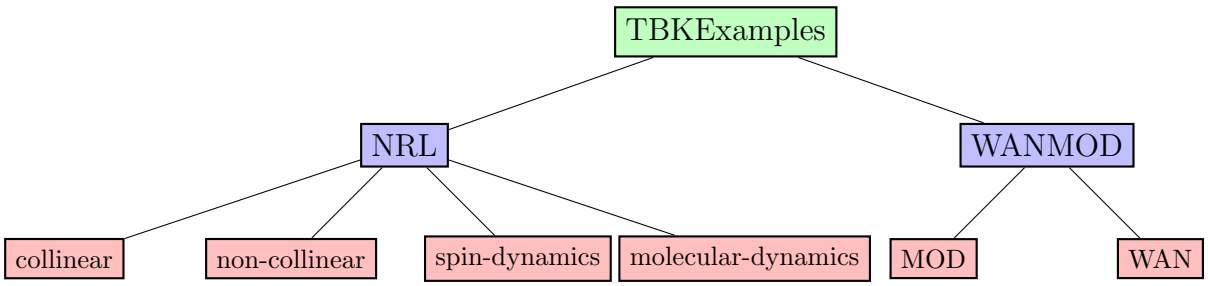
3.3 running TBKOSTER

TBKOSTERneeds an `in_master.txt` file to run. `in_master.txt` contains all the parameters of the calculation. This file will be described in details in Input File Command Reference section.

CHAPTER 4

Getting Started

To get started we recommend to first run the various examples that are provided within the TBKOSTER git repository.
In the TBKOSTER directory one can find a TBKExamples directory organized as follow;



In each sub-directory one can find a `README.md` file and a list of examplesxx directories each containing a `jobs.sh` file. This `jobs.sh` can be executed by the command `./jobs.sh`

CHAPTER 5

Input File Command Reference

5.1 input files

in_master.txt

The file `in_master.txt` is the main input file of TBKOSTER.

namelist	Variable	Description
&calculation		
	`pre_processing` (char)	type of pre-processing calculation: 'txt2xyz'→ write geometry in xyz format,
	`pre_processing_dir` (char)	directory for saving pre-processing calculation: 'txt2xyz'→ default value,

	`processing` (char)	type of processing calculation: 'scf'→ **self-consistent**, 'md'→ molecular-dynamics. 'sd'→ spin-dynamics.
	`post_processing` (char)	type of post-processing calculation (always preceded by a scf calculation): 'band'→ band structure calculation, 'dos'→ density of states calculation. 'forces'→ forces calculation. 'mft'→ magnetic force theorem.
&units		
	`energy` (char)	Energy unit: 'hau'→ **Hartree atomic units** , 'rau'→ Rydberg atomic units, 'ev'→ electronvolts.
	`length` (char)	Length unit: 'hau'→ **Hartree atomic units**, 'rau'→ Rydberg atomic units, 'nm'→ nanometer. 'ang'→ angstrom.

	`time` (char)	time unit: 'hau'→ **Hartree atomic units**, 'rau'→ Rydberg atomic units, 'fs'→ femtoseconds.
	`mass` (char)	Mass unit: 'hau'→ **Hartree atomic units**, 'rau'→ Rydberg atomic units, 'g/mol'→ g/mol.

&element		
	`ne` (int)	Number of different elements in the system.
	`symbol(i)` (char)	Symbol of the elements in the system $i = 1, \cdots, ne$
	`no(i)` (int)	number of orbitals of element $i = 1, \cdots, ne$ (optional)
	`o(i,1:no(i))` (int)	list of orbitals $i = 1, \cdots, ne$
	`q(i)` (real)	input charge of element $i = 1, \cdots, ne$
	`q_s(i)` (real)	s input charge of element $i = 1, \cdots, ne$
	`q_p(i)` (real)	p input charge of element $i = 1, \cdots, ne$
	`q_d(i)` (real)	d input charge of element $i = 1, \cdots, ne$
	`u_lcn(i)` (real)	U (in eV) value of element $i = 1, \cdots, ne$ for local charge neutrality
	`u_lcn_d(i)` (real)	Ud (in eV) value of element $i = 1, \cdots, ne$ for local d charge neutrality
	`i_stoner_d(i)` (real)	Stoner parameter of element $i = 1, \cdots, ne$ for of d orbitals for magnetic systems.
	`xi_so_p(i)` (real)	spin-orbit coupling constant of element $i = 1, \cdots, ne$ for p orbitals

	`xi_so_d(i)` (real)	spin-orbit coupling constant of element $i = 1, \cdots, ne$ for d orbitals
&element_tb		
	`tb_type` (char)	type of TB calculation 'nrl'→ **NRL TB**, 'mod'→ model TB (input=mod.dat), 'wan'→ Wannier TB (input=hr.dat).
	`filename(i)` (char)	file with the NRL TB parameters of element $i = 1, \cdots, ne$.
&lattice		
	`v_factor` (real)	Multiplication factor applied to the lattice vectors.
	`v(1:3,3)` (real)	3 translation vectors $\mathbf{a} = v(1,:)$,$\mathbf{b} = v(2,:)$,$\mathbf{c} = v(3,:)$,
&atom		
	`ns` (int)	Type of magnetic system. '1'→ **non magnetic system**, '2'→ collinear spin, '4'→ non-collinear spin.
	`na` (int)	Number of atoms in the system.
	`k_spiral(3)` (real)	spiral vector for spin-spiral calculations (defaut k_spiral=(0,0,0)).
	`ntag` (int)	Number of tags to give names to differents atoms.
	`stag(i)` (int)	Number of atoms of tag $i = 1, \cdots, ntag$.
	`tag(i)` (char)	tag of atom $i = 1, \cdots, ntag$.
	`pbc(3)` (int)	number of unit-cell in the three periodic directions to search for neighbours. if pbc(i)=0 then there is no periodicity in direction i.

`r_coord` (char)	type of coordinate for atom positions, '**direct**'→ in fractions of a, b and c coordinate, 'cartesian'→ xyz coordinates,
`x(i,3)` (real)	coordinates of atom $i = 1, \cdots, na$,
`m_coord` (char)	type of coordinate for magnetism, '**spherical**'→ m, θ and ϕ coordinate, 'cartesian'→ xyz oordinates,
`m_listing` (char)	type of magnetic assignation, '**by_atom**'→ by atom, 'by_tag'→ by tag,
`m(i,3)` (real)	magnetic coordinates of atom $i = 1, \cdots, na$ or $i = 1, \cdots, ntag$
`lambda_pen_listing` (char)	type of magnetic assignation, '**by_atom**'→ by atom, 'by_tag'→ by tag,
`lambda_pen(i)` (real)	magnetic penalization factor of atom $i = 1, \cdots, na$ or $i = 1, \cdots, ntag$
&mesh	
`type` (char)	type of k mesh '**mp**'→ monkhorst pack mesh, 'path'→ path in k-space, 'list'→ list of k-vectors.

`x_coord` (char)	type of k mesh '**direct**'→ in fractions of a, b and c coordinate, 'cartesian'→ xyz coordinates,
`gx(3)` (int)	$gx = (n_a, n_b, n_c)$, n_i integer, when type=mp.
`dx(3)` (int)	$dx = (d_a, d_b, d_c)$ $d_i = 0, 1$ shift when type=mp.
`nxs` (int)	number of symmetry point when type=path.
`gxs` (int)	number of points between two consecutive symmetry-point when type=path.
`xs(1:nxs,3)` (int)	coordinate of the symmetry points when type=path
`xs_label(1:nxs)` (int)	label (G,X,M,K etc..) of the symmetry points when type=path
`nx` (int)	number of k points when type=list.
`x(1:nx,3)` (int)	coordinate of the k points when type=list

&hamiltonian_tb

`e_e_interaction` (char)	type of electronic interaction '**stoner**'→ monkhorst pack mesh, 'ujb'→ TB+U(J,B),
`m_penalization` (char)	type of magnetic penalization '**none**'→ no penalization 'r'→ penalization of the amplitude of the spin moment $m(i)$ for each atom. 'r,θ'→ penalization of the amplitude of the spin moment $m(i)$ and on the $\theta(i)$ angle for each atom. 'r,θ,ϕ'→ penalization of the amplitude of the spin moment $m(i)$ and on the $\theta(i)$ and $\phi(i)$ angles for each atom. 'θ'→ penalization of the the $\theta(i)$ angle for each atom. 'θ,ϕ'→ penalization of the $\theta(i)$ and $\phi(i)$ angles for each atom. 'ϕ'→ penalization of the $\phi(i)$ angles for each atom.

&energy		
	`smearing` (char)	type of smearing '**mp**'→ Methfessel Paxton, 'fd'→ Fermi Dirac, 'mv'→ Marzari-Venderbilt, 'g'→ Gaussian,
	`degauss` (real)	electronic broadeding
	`fixed_fermi_level` (logical)	'**false**'→ Fermi level determined by number of electrons in the system, 'true'→ Fermi level fixed to a given value `en_f_ffl`,
	`en_f_ffl` (entier)	value of the fixed "Fermi level" if fixed_fermi_level=true
	`fixed_spin_moment` (logical)	'**false**'→ , 'true'→ Fermi spin moment calculation, the total magnetization being equal to `m_fsm`,
	`m_fsm` (entier)	value of the fixed spin moment when fixed_spin_moment=true
&mixing		
	`type` (char)	type of smearing '**broyden**'→ Broyden mixing 'linear'→ linear mixing,

	`alpha` (real)	mixing coefficient $0 < \alpha < 1$
	`n_init` (int)	first step of Broyden mixing (default 1)
	`n_hist` (int)	number of history steps of Broyden mixing (default 50)
&scf		
	`verbose` (logical)	type of smearing '**false**'→ minimum writing in out_log.txt file 'true'→ verbose writing in out_log.txt file,
	`delta_en` (real)	energy crtiterium for scf calculation
	`delta_q` (real)	charge crtiterium for scf calculation
	`ni_min` (int)	miniumum number of iteration (default 2)
	`ni_max` (int)	maximum number of iteration (default 50)

post-processing

a) band

`band/in_band.txt`

namelist	Variable	Description
&band		
	`proj` (char)	type of projection '**none**'→ no projection, 'site'→ atomic site projection, 'spin'→ spin projection (only if $ns = 4$) to plot spin-textures. 'orbit'→ orbit projection (only if $ns = 4$) to plot orbit-textures. 'spin,orbit'→ spin and orbit projection (only if $ns = 4$) to plot spin and orbit-textures.

`i_min` (int)	lowest band index to include in the calculation (default $i_min = 1$)
`i_max` (int)	largest band index to include in the calculation (default $i_max = nmax$)
`na_band` (int)	number of atomic sites on which projection should be done (default na_band=0).
`ia_band(1:na_band)`	index of the atomic sites.

`band/in_mesh.txt`

Note that for band-structure calculation, although the type of mesh is by default `type`=**mp**, band structure should be performed either with verb+type+=**path** or verb+type+=**list**.

- **path** is used for "traditional" band-structure plotting
- **list** is used for 2D plot of spin, orbit or velocity textures.

namelist	Variable	Description
	`type` (char)	type of k mesh '**mp**'→ monkhorst pack mesh, 'path'→ path in k-space, 'list'→ list of k-vector,
	`x_coord` (char)	type of k mesh '**direct**'→ in fractions of a, b and c coordinate, 'cartesian'→ xyz coordinates,

	`gx(3)` (int)	$gx = (n_a, n_b, n_c)$, n_i integer, when type=mp.
	`dx(3)` (int)	$dx = (d_a, d_b, d_c)$ $d_i = 0, 1$ shift when type=mp.
	`nxs` (int)	number of symmetry point when type=path.
	`gxs` (int)	number of points between two consecutive symmetry-point when type=path.
	`xs(1:nxs,3)` (int)	coordinate of the symmetry points when type=path
	`xs_label(1:nxs)` (int)	label (G,X,M,K etc..) of the symmetry points when type=path
	`nx` (int)	number of k points when type=list.
	`x(1:nx,3)` (int)	coordinate of the k points when type=list

b) PDOS

`dos/in_dos.txt`

namelist	Variable	Description
&dos		
	`nen` (int)	number of energy points
	`na_dos` (int)	number of atomic sites on which the pdos will be projected
	`ia(1:na_dos)` (int)	index of the sites.
	`en_min` (real)	lower bound of energy window.
	`en_max` (real)	upper bound of energy window.

`dos/in_energy.txt`

namelist	Variable	Description
&energy		
	`smearing` (char)	smearing type '**mp**'→ Mathfessel Paxton, 'mv'→ Marzari Vanderbilt, 'fd'→ derivative of Fermi-Dirac. 'g'→ Gaussian.
	`degauss` (real)	broadening

`dos/in_mesh.txt`

In `dos/in_mesh.txt` all the options of `&mesh` are available however for PDOS one usually only use a MP k-point grid. Hence we have only listed below the most common parameters.

namelist	Variable	Description
	`type` (char)	type of k mesh '**mp**'→ monkhorst pack mesh, 'path'→ path in k-space, 'list'→ list of k-vectors.
	`gx(3)` (int)	$gx = (n_a, n_b, n_c)$, n_i integer, when type=mp.
	`dx(3)` (int)	$dx = (d_a, d_b, d_c)$ $d_i = 0, 1$ shift when type=mp.

c) MFT (magnetic force theorem)

The magnetic force theorem is used in two contexts: to evaluate the magnetocrystalline anisotropy or to compare the total energy of different magnetic configurations.

`mft/in_mft.txt`

namelist	Variable	Description
&mft		
	`calc` (char)	type of calculation '**mae**'→ magnetic anisotropy calculation, 'mconfig'→ calculation of various magnetic non-collinear configurations,
	`type` (char)	type of mesh in (θ, ϕ) 'mesh'→ regular mesh over the full spherical coordinates, 'list'→ list of several (θ, ϕ) angles, 'path'→ path between various (θ, ϕ) angles,
	`na_mft` (int)	number of site on which the energy is projected (default na_mft=0)
	`ia(1:na_mfr)` (int)	index of the sites, if na_mft=0 no need to specify all the indices.
	`nxa` (int)	• **calc=mae&type=mesh**: (`nxa`,`nxa`) mesh over the spherical coordinates, • **calc=mae&type=path**: `nxa` is the number points between two angles, • **calc=mae&ype=list**: `nxa` is the number of angles in the list, • **calc=mfonfig**: `nxa` is the number of magnetic configurations.

`mconfig(na,nxa,3)` (real)	define explicitly the `nxa` magnetic configurations over the `na` atoms of system
`nangle` (int)	in case type=path; nangle is the number of angles (2) to generate the path.
`angle_xs(nxa,3)` (real)	list of angles in spherical coordinates

`mft/in_energy.txt`

namelist	Variable	Description
&energy		
	`smearing` (char)	smearing type '**mp**'→ Mathfessel Paxton, 'mv'→ Marzari Vanderbilt, 'fd'→ derivative of Fermi-Dirac. 'g'→ Gaussian.
	`degauss` (real)	broadening

`mft/in_mesh.txt`

In `mesh/in_mesh.txt` all the options of `&mesh` are available however for MFT one only use a MP k-point grid. Hence we have only listed below the most common parameters.

namelist	Variable	Description
	`type` (char)	type of k mesh '**mp**'→ monkhorst pack mesh, 'path'→ path in k-space, 'list'→ list of k-vectors.
	`gx(3)` (int)	$gx = (n_a, n_b, n_c)$, n_i integer, when type=mp.
	`dx(3)` (int)	$dx = (d_a, d_b, d_c)$ $d_i = 0, 1$ shift when type=mp.

5.2 output files

All the output files (`out_xxxx.txt`) of TBKOSTERare in namelist format (except `out_log.txt`), hence they can be re-used as input files for further calculations.

out_log.txt

`out_log.txt` is the main output file of TBKOSTER. The first lines of `out_log.txt` contains a summary of all the parameters (including the defaults) used for the calculation. Then, if `verbosity=.true.` TBKOSTERprint out in `out_log.txt` the charge and various quantities at each scf iteration.

out_charge.txt

`out_charge.txt` contains:

- the mulliken charge `q_mul(1:na,1:3,0:ns-1)`
- and net charge `rho_net(1:na,1:1:no,1:no,1:ns )`

Importantly `out_charge.txt` can be copied in `in_charge.txt` whenever one needs a restart, for instance when a calculation is not fully converged.

post-process output

a) txt2xyz

This option is used when one want to visualize the evolution of the magnetic configuration during the scf cycle. This is particularly useful in the case of non-collinear magnetism. This option necessitate the setting of `verbose=.true.` in `&scf`. `out_atom_tb_xx.txt` files are saved during the scf iterations. The `txt2xyz` option transforms the `.txt` file into `.xyz` that contain position and magnetization of atoms in the unit-cell. These files can be visualized via `avogadro` software to generate a movie showing the convergence process.

b) dos

The main output file is `dos/out_dos.txt` file. It contains in namelist format:

- `dos_tot(1:nE,1:ns)`: Total density of states
- `dos_'orb'(1:na_dos,1:nE,1:ns)`: Projected density of states on different sites, different orbitals (orb=s,px, py etc..) and different spin index.

c) band

The main output file is `band/out_band.txt` file. It contains in namelist format:

• `en_k(1:nh,1:nk,1:ns)`: Eigenvalues `nh`: size of Hamiltonian, `nk`: number of k points.

If proj=site

• `w_band_site(1:nh,1:nk,1:na_band,1:norb,1:ns)`: weight of wavefunction on different sites, different orbitals and different spin index.

If proj=spin

• `w_band_spin(1:nh,1:nk,0:na_band,1:4)`: average value of the Pauli matrices $\sigma_x, \sigma_y, \sigma_z$ (and its norm) for band index ih, k point ik, site ia_band.

If proj=orbit

• `w_band_orb(1:nh,1:nk,0:na_band,1:4)`: average value of the orbital matrices Lx, L_y, L_z (and its norm) for band index ih, k point ik, site ia_band.

d) mft

The main output file is `mft/out_mft_xxx.txt` file. It contains for each magnetic configuration `xxx` in namelist format:

• `mconfig(1:na,1:2)`: Magnetic configuration (θ, ϕ) for each atom of the unit-cell

• `mft_'orb'(1:na_mft)`: band energy projected on the selected sites for each orbitals (orb=s,px,py etc..).

• `mft_sum`: Sum of the band energies projected on the selected sites of the unit-cell

• `mft_tot`: Total band energy, it should be equal to `mft_sum` if `nsite=na`

tools output

The various tools read the output files of TBKOSTERin namelist format and reorganize them in a format directly readable by plotting software (`xmgrace`, `gnuplot` etc..)

a) pdos.x

`pdos.x` generates the following files:

• `dos_tot.dat`: Total density of states

• `pdos-s.dat`, `pdos-p.dat`, `pdos-d.dat` and `pdos-spd.dat`: PDOS

b) bands.x

`bands.x` generates different types of files depending on the options for the projection (proj) and for the type of mesh.

mesh=path

If proj=none

• `band.dat`: band structure along the k path

If proj=site

• `band.dat`: band structure along the k path

• `band_weight_site_orb.dat`: band structure along the k PATH and associated weight on the defined sites.

If proj=spin and/or proj=orb

• `band_weight_spinorb.dat`: band structure along the k PATH and associated amplitude of the spin and/or orbit.

mesh=list

Note that the "list" k-points are usually generated in cartesian coordinates via the `build_kpoints.f90` program. "list" mesh option is used to generate 2D maps

If proj=spin and/or proj=orb

`bands.x` generates two files:

• `fermi.dat`: 2D maps for Fermi surface

• `fermi.dat`: 2D maps for "spin" or "orbit" vector field.

c) mft.x

There are two type of MFT calculations: `calc`= mae or `calc`=mconfig.

calc=mae

`mft.x` reads the `out_mft_xxx.txt` files and creates several files:

If type=path

• `mae_angle.dat`: MAE along the (θ, ϕ) path where the first angle of the path is taken as the reference zero energy.

• `mae_atom.dat`: MAE energy decomposed on the different sites, and evaluated as the difference between the first and last magnetic orientation.

• `mae.xyz`: Same as `mae_atom.dat` but in (x, y, z) coordinates.

If type=mesh

• `mae_angle.dat`: MAE along the (θ, ϕ) mesh where the first angle of the path is taken as the reference zero energy.

calc=mconfig

`mft.x` reads the `out_mft_xxx.txt` files and creates one summary file:

• `mft_config.dat` which contains the list of MFT band energies for the various configurations.

Bibliography

[1] *Magnetism of iron: from the bulk to the monatomic wire* Gabriel Autès, Cyrille Barreteau, Daniel Spanjaard and Marie-Catherine Desjonquères J. Phys.: Condens. Matter **18** 6785-6813 (2006) .

[2] *Magnetocrystalline anisotropy energy of Fe(001) and Fe(110) slabs and nanoclusters: A detailed local analysis within a tight-binding model* D. Li, A. Smogunov, C. Barreteau, F. Ducastelle, and D. Spanjaard Phys. Rev. B **88** 214413 (2013).

[3] *An efficient magnetic tight-binding method for transition metals and alloys* C. Barreteau, D. Spanjaard, MC Desjonqueres Comptes Rendus Physique **17** 406-429 (2016).

[4] *Magnetocrystalline anisotropy of Fe, Co, and Ni slabs from density functional theory and tight-binding models* Ludovic Le Laurent, Cyrille Barreteau, and Troels Markussen Phys. Rev. B **100**, 174426 (2019).

[5] *Spin dynamics from a constrained magnetic tight-binding model* Ramon Cardias, Cyrille Barreteau, Pascal Thibaudeau, and Chu Chun Fu Phys. Rev. B **103**, 235436 (2021).