A guide to the Tight-Binding FITting (TBFIT) package

Hyun-Jung Kim [Infant@kias.re.kr]

August 23, 2020

This document is to provide explanation for the input file arguments of the TBFIT package.

System Requirements and installation

The program has been written by modern `Fortran2008` language. If you want to deactivate the use of some module interfaces written in Fortran2008 syntax, please remove `-DF08` option in your option tag of the makefile.

`LAPACK` library should be properly linked in the makefile. For the eigenvalue solver with sparse matrix, `Inspector-executor Sparse BLAS Routines` and `Extended Eigensolver Routines` in the `Intel Math Kernel Library (Intel MKL)` are refered. If the system size is very big, you can calculate band structure with energy window constraint. This is available with EWINDOW tag and `-DMKL_SPARSE` option. To use `-DMKL_SPARSE` option, make sure that `mkl_spblas.f90` file is located in your `$MKLPATH/include` folder.

To list up the space group information for the given geometry in the initial stages of the calculations, one can activate the use of space group library (Spglib). For this, put `-DSPGLIB` in your `OPTION` tag of the `makefile`, and provide appropriate library path in `SPGLIB` tag.

```
#---------------------------------|
# Compiler options and bin path    |
#---------------------------------|
OPTIONS= -fpp -DMPI -DF08 -DSPGLIB -DMKL_SPARSE
F90    = mpif90 $(OPTIONS)
FFLAG  = -O3 -heap-arrays -nogen-interfaces
BIN    = ~/code/bin
```

```
#----------------------------------|
# Dependencies: LAPACK, SPGLIB     |
#----------------------------------|
SPGLIB = -L/Users/Infant/code/lib/ -lsymspg
MKLPATH= $(MKLROOT)
LAPACK = -L$(MKLPATH)/lib/
         -lmkl_intel_lp64 -lmkl_sequential
         -lmkl_core -liomp5
```
makefile example

- How to install:

```
> tar -xvf TBFIT-master.zip
> cd TBFIT-master
> make tbfit
```

- How to run:
  In the `Example` directory, you can run a test cases, for example:

```
> cd TBFIT-master/Example/1H-MoS2/SOC
> tbfit
```

Note that the output log will be written in `TBFIT.out` file by default. If you want to write the log with different name, please use -log option as follows:

```
> tbfit -log FNAME.out
```

Part I.
User's Guide

1. INPUT tags of the INCAR-TB

GET_BAND *logical* Default: .TRUE. If .TRUE. TBFIT will perform tight-binding calculations for band structure evaluation.

TBFIT *logical* Default: .FALSE.

.TRUE. : Perform tight-binding parameter fitting which is defined in PFILE. After fitting is completed, whatever it is converged or not, additional tight binding calculations as defined in the INCAR-TB will be performed.

.FALSE. : Do not perform fitting procedures. In this case, regular tight binding calculations will be performed.

MITER *integer* Default: 100
Maximum number of iteration for the fitting procedures. If GA is set for LSTYPE, MITER represents the maximum number of generations.

MXFIT *integer* Default: 1
Maximum number of repeat for LMDIF procedures after MITER steps done. By checking convergence criteria defined by FDIF the variation of the fitness function, parameter constraints, i.e., upper or lower bounds defined by SET CONSTRAINT tags, TBFIT will determine to stop or not.

LSTYPE *integer* Default: LMDIF
Method for parameter fitting. Available tags are LMDIF and GA.

LMDIF method: Levenberg-Marquardt method[1, 2] using finite-difference for Jacobian.

GA method: Genetic Algorithm[3] based on PIKAIA library[4,5,6,7]. To setup control parameters for the GA, see Sec. GA.

[1] Kenneth Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares" *Quarterly of Applied Mathematics* 2, 164 (1944).
[2] Donald Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters" *SIAM Journal on Applied Mathematics* 11, 431 (1963).
[3] D. E. Goldberg, "Genetic Algorithm in Search, Optimization, & Machine Learning" *Addison−Wesley* (1989).
[4] P. Charbonneau and B. Knapp, "A user's guide to PIKAIA 1.0", (NCAR Technical Note 418+IA, 1995)
[5] P. Charbonneau, "An introduction to genetic algorithm for numerical optimization" (NCAR Technical Note 450+IA, 2002)
[6] P. Charbonneau, "Release notes for PIKAIA 1.2" (NCAR Technical Note 451+STR, 2002), http://www.hao.ucar.edu/modeling/pikaia/pikaia.php
[7] Modern Fortran Edition of the Pikaia Genetic Algorithm. https://github.com/jacobwilliams/pikaia

PTOL & FTOL *real* Default: 0.00001

Tolerence of iteration of the fitting procedures for LMDIF method. FTOL is a tolerence for the difference between target and calculated data from tight binding method. PTOL is as tolerence for the tight binding parameters. Normally, both values below 0.00001 is sufficient to reach a local minima.

FDIFF *real* Default: 0.001

Tolerence of iteration of the fitting procedures for LMDIF method. FDIFF is a tolerence for the difference between fitness function of previous and last steps. After finishing LMDIF iterations, fitting procedures might start again if CONSTRAINT and FDIFF condition does not fulfill, and will stops by MXFIT criteria.

K_UNIT *string* Default: ANGSTROM

ANGSTROM : the unit of the k-point will be written in Å^{-1} unit.

RECIPROCAL : the unit of the k-point will be written in reciprocal unit (fractional).

PFILE *string* File name for tight-binding parameters. Default: PARAM_FIT.dat For the details, see Sec.4.

Note 1: After fitting procedure is over, the WEIGHT information is written in PFILE so that one can restart fitting. To use WEIGHT information written in PFILE, one can add USE_WEIGHT tag as below:

```
PFILE  PARAM_FIT.dat  USE_WEIGHT
```

POFILE *string* Output file name for tight-binding parameters written after fitting procedures. Default: PARAM_FIT.new.dat

IS_SK or SLATER_KOSTER *logical*

.TRUE. : Slater-Koster type of hopping parameters will be assumed.

.FALSE. : User defined or direct hopping parameters will be assumed. *Warning* : This is experimental feature and on the development stages (do not set to .FALSE.).

SGPLIB *logical*

.TRUE. : Write space group information to the output log.

.FALSE. : Do not write space group information to the output log.

Note that this option is only applicable if you have put -DSPGLIB option in your makefile . See the details in System Requirements and installation section.

EFILE *string*, *integer*

File name for the *target* band structure for the fitting procedures. If the second *integer* n is followed by, TBFIT will read n-th column as a target band. Default is n=2.

EFILE DFT_BANDSTRUCTURE.dat 2

```
    # 1st eigen value
    # k-path  energy(eV)
      0.00000  -12.36137
      0.01693  -12.36162
      0.03386  -12.36118
        [...]
      0.16932  -12.33324
      0.18625  -12.32696
      0.20319  -12.32014


    # 2nd eigen value
    # k-path  energy(eV)
      0.00000  -12.36137
      0.01693  -12.36041
      0.03386  -12.35875
         [...]
      0.16932  -12.32136
      0.18625  -12.31394
      0.20319  -12.30600

        [...]
```

EFILE DFT_BANDSTRUCTURE.out example

If LORDER = .TRUE. and TBFIT = .TRUE., re-ordered target energy information should be provided in parallel with the original energy, as below:

```
# k-path          energy(eV)     energy_ordered(eV) : 1st eig
   0.00000000     -14.75638384     -14.75638384
   0.04796527     -14.74941045     -14.74941045
        [...]
   2.95861071     -14.74708715     -14.74708715
   3.01399626     -14.75638384     -14.75638384

        [...]

# k-path          energy(eV)     energy_ordered(eV) : 3rd eig
   0.00000000      -2.27711578      -2.27711578
   0.04796527      -2.27002886      -2.27002886
         [...]
   2.95861071      -2.26767593      -0.44510940
   3.01399626      -2.27711578      -0.44395430
        [...]
```

EFILE DFT_BANDSTRUCTURE_ORDERED.out example

If the second argument is $VASP$, TBFIT will read EFILE which is indicated as the first argument and recognize this file as a $EIGENVAL$ file of VASP code. (note: In this case, LORDER should be .FALSE..) For example,

```
EFILE EIGENVAL VASP
```

EFILE_EF *real*

The fermi level of EFILE. The energy will be shifted by the EFILE_EF, i.e., $\epsilon^{target}_{n,k} = \epsilon^{DFT}_{n,k} -$ EFILE_EF, where $\epsilon^{DFT}_{n,k}$ is the eigenvalues provided by the EFILE.

GFILE *string* Default: POSCAR-TB

File name for the geometry and atomic orbital informations. The format is exactly same as POSCAR of VASP program. For the details of setting atomic orbitals, see Sec.3.

```
GFILE POSCAR-TB
```

The example file format of "GFILE" is as follows,

```
MoS2 # comment
  1.00000000000000  # scaling factor
    3.1716343    0.000000    0.00000  # lattice vector a1
    1.5858171    2.746715    0.00000  # lattice vector a2
    0.0000000    0.000000   15.00000  # lattice vector a3
  Mo  S                               # atomic species
    1   2                           # number of atoms per species
Direct     # coordinate type (direct or cartesian)
 0.00000 0.00000 0.50000 dz2 dxy dx2 dyz dxz  # coord, orbital
 0.33333 0.33333 0.60645 s px py pz
 0.33333 0.33333 0.39354 s px py pz
```

POSCAR-TB example: MoS_2 with Mo-d and S-sp

Note: TBFIT reads "GFILE" and reports it on to the screen in the parsing steps. If "PRINT_GEOM .FALSE." tag is followed by, then program will not report the geometry informations, for the convenience.

```
GFILE POSCAR-TB PRINT_GEOM .FALSE.
```

KFILE *string* Default: KPOINTS_BAND

File name for the k-point setting. The format is exactly same as KPOINTS of VASP program.

```
k-points line mode example
  40  ! intersections
Line-mode
Reciprocal
  0.50000000  0.5000000 0 M
  0.33333333  0.6666666 0 K

  0.33333333  0.6666666 0 K
  0.00000000  0.0000000 0 G

  0.00000000  0.0000000 0 G
  0.66666666  0.3333333 0 K'
```

KPOINTS_BAND *line mode* example

```
k-points grid mode example
  0
GMonkhorst-Pack #'G'amma centered grid mode
  4  4  1   # grid  nk_1 nk_2 nk_3
  0  0  0   # shift
```

KPOINTS_BAND *grid mode* example

Note1: In VASP, n k-points are generated along a k-path if n division is specified in the second line of KFILE. This will gives $n \times m$ total k-points and $n-1$ divisions along each k-path, where m represents the total number of k-path specified. And at the end of each line segment of k-path, same k-points are marked twice. However, some program, such as FLEUR uses different strategy in generating k-path, that is, $n+1$ k-points along each path with only one k-points at the end of line segments. Hence, EFILE originated (or postprocessed) from FLEUR would have different format. This will gives $n \times m + 1$ total k-points and n division of each k-path. You can read this kind of EFILE file as well, if you have also specify as below, letting TBFIT to recognize the type of generated k-path.

```
KFILE KPOINTS_BAND  FLEUR
```

Note2: If your target band structure is originated from FHI-aims, the division between each k-path can be specified separately. For example, the typical way to specify band structure calculation tag in FHI-aims looks as follows:

```
output band 0.00  0.00  0.00    0.50  0.00  0.00  70   G  X
output band 0.50  0.00  0.00    0.50  0.50  0.00  20   X  M
output band 0.50  0.50  0.00    0.00  0.50  0.00. 70   M  Y
```

Here, each k-path is divided with 70, 20, and 70 divisions. In this case, you can specify each numbers (70 20 70) to your second line of KFILE.

```
k-points line mode example
  70 20 70  ! intersections
Line-mode
Reciprocal
  0.0  0.0 0.0 G
  0.5  0.0 0.0 X

  0.5  0.0 0.0 X
  0.5  0.5 0.0 M

  0.5  0.5 0.0 M
  0.0  0.5 0.0 Y
```

KPOINTS_BAND *line mode* (FHI-aims type)

KREDUCE *integer* Default: `1`

For some cases, the target EFILE contains too many k-points with larger n of divisions between each line segment. In this case, you can reduce the number of k-points to be read by skipping k-points except every nk_{reduce}.

For example, if it has been divided with 10 k-points between each symmetry points in your EFILE, you can reduce by factor of 2 or 5 by specifying

```
KREDUCE    2
or
KREDUCE    5 .
```

In this case, if KREDUCE is 2, then the total number of divisions will become 5, and this value should be same as number of k-point division which is specified in second line of your KFILE. Note that `KREDUCE` should be one of prime number of the k-point division of EFILE.

LOCCHG *logical* Default: `.FALSE.`

Setting tag for local potential. If `.TRUE.`, one should give proper local potential parameter in your PFILE and should properly setup LOCAL.POT tag in your GFILE. For the details, see the explanation of LOCAL.POT in Sec.4.

TYPMAG *string* Default: `NONMAG`

Setting tag for magnetic moment: `nonmagnetic`, `collinear`, `noncollinear` If `collinear` and `noncollinear` tag is applied, MOMENT or MOMENT.C in the GFILE should be set up appropriately. For details, see MOMENT of the Sec.3.

LSORB *logical* Default: `.FALSE.`

Setting tag for spin-orbit coupling. If `.TRUE.`, *lambda_orb_spec* should be properly defined in the PFILE. For details, see Sec.4

LORBIT *logical*,*string*(*optional*),*string*(*optional*) Default: `.TRUE.`

Setting tag for orbital decomposed output. If `.TRUE.` the local orbital contribution will be printed out in `bandstructure_TBA.dat` file. If you write `rh` or `mx` or `my` or `mz` next to the logical text with `.TRUE.`, then, corresponding magnetization values, which represents the expectation value of pauli matrices σ_i where i={0,1,2,3}, will be printed out. Here σ_0 is 2×2 identity matrix to print out local orbital contribution. For example,

```
LORBIT .TRUE. mz
```

will print out <mz>. If you write `re` or `im` next to the logical text with `.TRUE.`, then, `real` or `imaginary` part of the wavefunction coefficient will be printed out. Note that this option only applicable with `LSORB .FALSE.` in the current version.

```
LORBIT .TRUE. re
```

If you write `wf` next to the logical text with `.TRUE.`, then, the wavefunction coefficient will be printed out. The real and imaginary part for each orbital basis is written. If `LSORB .TRUE.`, the spinor-up and spinor-dn part will be written, so that four real values will construct wavefunction coefficient.

```
LORBIT .TRUE. wf
```

Note that the corresponding output file `bandstructure_TBA.dat` file will be basically written by ascii (formatted) format. If you want to write in binary (unformatted) format, specify by `bin` (complex*16; double precision) or `bin4` (complex*8; single precision) tag next[8]. For example,

```
LORBIT .TRUE. wf bin
or
LORBIT .TRUE. wf bin4
```

This tag will generate `band_structure_TBA.(up/dn).bin` file.

[8]In the current version, `bin` or `bin4` tag is only applicable when it is combined with `wf` option

PROJ_BAND *logical, integers* Default: `.FALSE.`

Setting tag for orbital/atom projected band structure output. The output will be written in separate file for each atom. The correct usage is as follows:

```
PROJ_BAND .TRUE. 1:4 7
```

then you can get `band_structure_atom.#.dat` file

```
band_structure_atom.1.dat
band_structure_atom.2.dat
band_structure_atom.3.dat
band_structure_atom.4.dat
band_structure_atom.7.dat
```

and additionally, `band_structure_atom.sum1.dat` file will be printed out as well, where projected local DOS for those atoms ({1,2,3,4,7}) are summed up in a single file.

If you write another `PROJ_BAND` tag specifying different atom sets in the separate line, for example,

```
PROJ_BAND .TRUE. 1:4 7
PROJ_BAND .TRUE. 5:8
```

then, you can get another set of files (`band_structure_atom.sum2.dat` and `band_structure_atom.{5..8}.dat`) as follows:

```
band_structure_atom.1.dat
band_structure_atom.2.dat
band_structure_atom.3.dat
band_structure_atom.4.dat
band_structure_atom.5.dat
band_structure_atom.6.dat
band_structure_atom.7.dat
band_structure_atom.8.dat
band_structure_atom.sum1.dat  (= 1+2+3+4+7)
band_structure_atom.sum2.dat  (= 5+6+7+8)
```

Note1: The atom index should be written in ascending order and should not exceed total number of atoms of your system.
Note2: Since in the `band_structure_atom.sum?.dat` file the projected local DOS for each orbital basis of the specified atoms are summed up, one should make sure that the specified atoms should have same orbital basis set. For instance, let's assume that you have atoms A={1:4, 7} with orbitals p_x, p_y, p_z, and atoms B={5:8} with orbitals s, p_x, p_y, p_z. Then it will works fine with "`PROJ_BAND .TRUE. 1:4 7`" tag, but not works fine with "`PROJ_BAND .TRUE. 1:4 5:8`".

LOAD_HOP *logical*, *string* Default: `.false.`

If `.true.`, one can load `hopping` file to read t_{ij} value. The following *string* should be the file name to be read. And the syntax of the file should be exactly same as the `hopping.dat` file, which is generated in the initial stages of the calculation. Hence, if you have pre-generated `hopping.dat` file (with `LOAD_HOP .FALSE.`), you can copy it with a different name and modify the elements of t_{ij} column, and rerun the code with following tag (for example, if you have copied `hopping.dat` → `hopping_modified.dat`):

```
LOAD_HOP   .TRUE.    hopping_modified.dat
```

Below, you can see that the original hopping element can be modified by changing values of the `t_IJ(eV)` column.

```
# Iatom Jatom     Rij       ... ORB_I ... ORB_J ...  t_IJ(eV)  ...
    1     1    0.0  0.0 0.0 ...   s   ...   s   ...   -4.0     ...
    1     1    0.0  0.0 0.0 ...   s   ...   px  ...    0.0     ...
    1     1   -1.2 -0.7 0.0 ...   s   ...   s   ...   -3.9     ...
    1     2   -1.2 -0.7 0.0 ...   s   ...   px  ...    1.9     ...
    ...
    ...
```

hopping.dat example file

```
# Iatom Jatom     Rij       ... ORB_I ... ORB_J ...  t_IJ(eV)  ...
    1     1    0.0  0.0 0.0 ...   s   ...   s   ...   -2.0     ...
    1     1    0.0  0.0 0.0 ...   s   ...   px  ...    0.0     ...
    1     1   -1.2 -0.7 0.0 ...   s   ...   s   ...   -3.9     ...
    1     2   -1.2 -0.7 0.0 ...   s   ...   px  ...    1.9     ...
    ...
    ...
```

hopping_modified.dat example file

IBAND *integer* Default: `1` (*deprecated*)

`IBAND` is the first eigenstate of the target data of EFILE. This value will be used in the WEIGHT `SET` section.

FBAND *integer* Default: `NEIG` (*deprecated*)

`NEIG` : number of orbital basis of the system. `FBAND` is the last eigenstate of the target data of EFILE. This value will be used in the WEIGHT `SET` section.

SCISSOR *integer*, *real*

If set, in the fitting procedures, target energy `EDFT(`n`,`k`)` will be shift by amound of the scissor operation. This operation works as follows: $\mathrm{E}'_{target}(n,k) = \mathrm{E}_{target}(n,k) + e_{scissor}$ if $n >= \mathrm{i}_{scissor}$. Note that this operation is only valied if TBFIT is `.TRUE.`.

```
SCISSOR 29 0.2  # i_scissor = 29 and e_scissor = 0.2 (eV)
```

NN_MAX *integer* Default: 3

Determine how many times the cell will be repeated in searching hopping pairs. If your system is sufficiently larger than the maximal value of hopping distances of your system, this can be reduced to 1, otherwise just use default value.

```
NN_MAX 3 3 3
```

or

```
NN_MAX 3
```

both settings will give 3×3×3 cell repeat.

ERANGE *integer* Default: `1 NEIG`

If provided, the energy level between these energy window will be printed out in the `bandstructure_TBA.dat` file.

```
ERANGE  4400 4700
```

Above example means that the energy level from 4400^{th} to 4700^{th} will be printed. This is particularly useful if you calculate very large systems. By setting `ERANGE` tag, you can save disk space a lot if LORBIT tag is turned on where orbital component information takes huge memory for larger systems.

EWINDOW *real*, *integer* Default: not activated

The eigenvalues within the energy window [emin:emax] will be calculated and stored. This option also useful in dealing with huge system. The usage for this tag is as follows:

```
EWINDOW  -5.0:5.0   NE_MAX 10
```

If provided, the energy level between these energy window will be printed out in the `bandstructure_TBA.dat` file.

In the above setting, the eigenvalue ($\{e\}$) within the energy window [-5.0:5.0] will be calculated and stored. The `NE_MAX` represents the maximum number of eigenvalue to be searched within the window and usually should be larger than the number of actual eigenvalues (`NE`) within the range and should not exceed the total number of eigenvalue (`NE_TOT`) of the system. The optimal values for `NE_MAX` is about 1.5×`NE`[9]. Since the `NE_MAX` is critical to the calculation speed, choosing the optimal values is essential. During the calculation, the program will find the optimal `NE_MAX` and update in every k-point loop.[10]

Note 1: If the tag is specified in your input file, the Hamiltonian matrix will be constructed with the sparse matrix format rather than dense matrix format. The libraries to dealing with the sparse matrix is referred from `Intel Math Kernel Library (MKL)`, please make sure that your library path is properly assigned. (suggest to use `MKL` version $\geq$ 11.3)

Note 2: If `NE_MAX` is not provided or exceeding `NE_TOT`, i.e., `NE_MAX`≥`NE_TOT`, `NE_MAX` will be set to `NE_TOT` by default.

PRTSEPK *logical* Default: `.FALSE.`

If `.TRUE.`, band structure file, `band_structure_TBA.dat`, will be separated for each k-point, i.e., `band_structure_TBA.kp_1.dat`, `band_structure_TBA.kp_2.dat`, ..., etc. This tag is useful if you are dealing with very large system and many k-points, where due to the the memory problem, calculation get failed to be finished.

PRTHAMK *logical* Default: `.FALSE.`

If `.TRUE.`, hamiltonian matrix H_k will be written for each k-point into separate file `Matrix.Hk_K`ik`_SP`is`.dat`, where ik and is represents k-point and spin index, respectively.

LORDER *logical* Default: `.FALSE.`

If `.TRUE.`, band structure will be re-ordered by maximizing the overlap between neighboring k-points. The overlap can be defined by the inner product of the wavefunctions $\langle u_{n,k}|u_{m,k-1}\rangle$. The re-ordered band structure will be written in `band_structure_ordered.dat` file. If `OV_CUT` tag is also provided with *real* value, the cutoff for the overlap integral to be considered with the value. The bands above `OV_CUT` will be swapped. The default value is $\frac{\sqrt{2}}{2}$.

```
LORDER .TRUE. OV_CUT 0.7
```

[9] Eric Polizzi, "Density-matrix-based algorithm for solving eigenvalue problem" *Physical Review B* 79, 115112 (2009)

[10] Though, one need to provide reasonable `NE_MAX` to save the memory, since `NE_MAX` is used to reserve memory space for the eigenvector store internally.

NOTE: (important) If the LORDER = .TRUE. with TBFIT = .TRUE., then the fitting will be done with re-ordered band structure. Therefore, re-ordered target energy information should be provided in parallel with the original energy in EFILE. Please check EFILE tag, for the instruction how to provide re-ordered energy in this case. To get re-ordered energy: for VASP, one can use VaspBandUnfolding tools for re-ordering: for QE, one can use bands.x postprocessing program. In this case, WEIGHT tag should be carefully adjusted according to the re-ordered band index.

LPHASE *logical* Default: `.TRUE.`

If `.TRUE.`, we construct Bloch basis functions $\chi_i^{\vec{k}}(\vec{r}) = \sum_{\vec{R}} e^{i\vec{k}\cdot(\vec{R}+\vec{t}_i)}\phi_i(\vec{r}-\vec{R})$.
If `.FALSE.`, the phase factor $e^{i\vec{k}\cdot\vec{t}_j}$ is not included in the definition of Bloch functions.

LTOTEN *logical* Default: `.FALSE.`

If `.TRUE.`, calculate total energy $E_{TOT} = E_{band}$, where $E_{band} = \sum_{n,k} f_{n,k} e_{n,k}$. Here, $f_{n,k}$ and $e_{n,k}$ is the Fermi-Dirac occupation function and eigen value of n-th level of k-th $\vec{k}$-point, respectively.
To calculate Fermi level, one should provide number of electrons and electronic temperature by NELECT and ELTEMP tag.

NELECT *real* Default: `.not specified.`

Total number of electrons of the tight binding models. In the collinear calculaltions, i.e., TYPMAG is *collinear*, both spin-up and spin-down components should be specified together as follows:

```
NELECT  10   10   # for spin-up and spin-dn
```

ELTEMP *real* Default: `0.0`

Electronic temperature, which will be used in Fermi-Dirac distribution function $f(T) = \frac{1}{1+\exp(e-\mu)/k_BT}$ to obtain Fermi level, where e is energy level, μ is Fermi level, and k_B is Boltzman constant.

SET *string*

Setting tags for post processing, parameter constraints, and nearest neighbor setups, etc. Available list for the `SET` tags are as follows,

`GA` : for Genetic Algorithm setting

CONSTRAINT TBPARAM

NN_CLASS

RIBBON

2. Details of the SET

Each SET tag should be ended up by END tag.

GA Setting of control parameters for the Genetic Algorithm used in parameter fitting procedures. This setting is only effective when LSTYPE is set to GA. Below you can check the default settings for GA procedures. You can modify as your purpose or comment out to use default setup as a input.

```
SET GA
  MGEN 100 # maximum number of iterations. (default:500)
  NPOP 100 # population in each generation. (default:100)
  NGENE 6  # number of genes in chromosomal encoding.
           # should be in between 2 to 9. (default:6)
  PCROSS 0.85 # crossover probability. [min:max]=[0.0:1.0]
  RMUTMIN 0.0005 # minimum mutation rate. [0.0:1.0]
  RMUTMAX 0.25   # maximum mutation rate. [0.0:1.0]
  RMUTINI 0.005  # initial mutation rate. [0.0:1.0]
  MUT_MOD 2      # mutation with 1: fixed rage
                 # mutation       with 2: fitness dependent
                 # mutation       with 3: distance dependent
                 # mutation+creep with 4: fixed rate
                 # mutation+creep with 5: fitness dependent
                 # mutation+creep with 6: distance dependent
  FDIF 1.0      # relative fitness differential [0.0:1.0]
  IREP 3   # reproduction plan 1: Full generational replacement
           #                   2: Steady-state-replace-random
           #                   3: Steady-state-replace-worst
  IELITE 0   # elitism 0: off, 1: on
             # Note that this tag applies only if IREP=1 or 2.
  VERBOSE 1  # printed output 0/1/2=None/Minimal/Verbose
  CONVTOL 0.0001 # convergence tolerance (must be > 0.0).
  CONVWIN 20 # convergence window.
             # If CONVWIN consecutive solutions are found
             # convergence will be declaired.
             # Hence, give larger convergence window to reach minima.
  IGUESSF 0.1 # fraction of the initial population to set equal
              #to the initial guess. [0.0:1.0]
  ISEED 999 # random seed value (must be > 0).
END GA
```

GA default setup example

STMPLOT Setting of integrated eigen state wavefunction $\Sigma|\psi_{nk}(r)|^2$ plot. Here, the summation runs over the eigen states within the energy window specified by STM_ERANGE or equivalently STM_WINDOW.

```
SET STMPLOT
  NGRID 40 40 80 # GRID for CHGCAR-STM output (default = 0.1 ang).
  STM_ERANGE -1.0:0.0  # energy window
  RCUT 6.0 # cut off radius(Å). Beyond this will not be calculated.
  REPEAT_CELL T T T  # repeat orbital for each lattice vector?
     # this logical tag is especially useful if you only
     #consider center region of the very large cell.
     # If set "T T F", orbital contribution which is periodically
     # repeated in a3 direction wll not be considered to calculate.
     # Try this option if  you have very large cell and you are
     # especially interested unitcell ceter.
END STMPLOT
```

STMPLOT setup example

EIGPLOT Setting of eigen state wavefunction $\psi_{nk}(r)$ or charge density $|\psi_{nk}(r)|^2$ plot.

```
SET EIGPLOT
  IEIG 3 5   # index(es) n of eigen state.
  IKPT 1 10  # index(es) k of k-point.
  NGRID 40 40 80 # GRID for CHGCAR output (default = 0.1 ang).
  RORIGIN 0.0 0.0 0.0 # shift of the origin of the cube file.
  WAVEPLOT .TRUE. # plot wavefunction (.true.) or charge density.
  RCUT 6.0 # cut off radius(Å). Beyond this will not be calculated.
END EIGPLOT
```

EIGPLOT setup example

DOS Setting of Density of states (DOS).

```
SET DOS
  GKGRID 100 100 1   # set Gamma centered Monkhorst-Pack grid
  KSHIFT 0.0 0.0 0.0 # shift of k-grid (k-offset)
  PRINT_KPTS .TRUE. IBZKPT-DOS_TB # print k-point to the file
  PRINT_EIG .TRUE. 1:2 3 # print specified energy surface
  PRINT_UNIT RECIPROCAL # k-point unit (or ANGSTROM 1/A)
  SMEARING 0.03 # gaussian smearing. Default = 0.025
  NEDOS 2000    # number of grid points in energy window (erange)
  DOS_EWINDOW -20.0:10.0 # energy window to be plotted
  DOS_NRANGE 1:NEIG # energy window to be calculated (integer)
  DOS_SPARSE .TRUE. # or .FALSE. use sparse matrix? Default=.FALSE.
  DOS_FNAME DOS_TB_projected.dat # output file name for DOS output
  PRINT_LDOS .TRUE. 1:8 12 # Print local density of states for given
                           # atoms. Here, 1 to 8-th atoms and 12-th
                           # atoms will be resolved.
  LDOS_FNAME LDOS_TB_projected  # header for LDOS file name.
```

```
                          # atom index will be appended after.
                          # For example, for atom-1,
                          # LDOS_TB_projected_atom.1.dat file will
                          # be generated.
END DOS
```
DOS setup example

Note1: NEIG variable of the DOS_NRANGE tag indicates total number of states, i.e., N_ORB×ISPINOR, where N_ORB is total number of atomic orbitals and ISPINOR = 1 (LSORB = .FALSE.) or 2 (LSORB = .TRUE.). If you want to reduce calculation loads, you can adjust DOS_NRANGE.
Note2: DOS_SPARSE tag is only available if -DMKL_SPARSE option is activated in the makefile. If set to .TRUE., DOS_NRANGE should be as following:

```
DOS_NRANGE  1:NE_MAX
or
DOS_NRANGE  NE_MAX
```

Here, NE_MAX is integer value larger than zero and less equal than total number of states NEIG. This setting will reduce the resources required for hamiltonian matrix construction and time consuming for the eigenvalue problem by the energy window constraint in the help of sparse matrix eigen solver. See EWINDOW for more informations.
Note3: PRINT_EIG is only applicable if DOS_SPARSE = .FALSE.

EFIELD Setting of E-field.

```
SET EFIELD
  EFIELD  0.0 0.0 0.1  # Efield along z direction
  EF_ORIGIN  0.0 0.0 0.345690593  # (in fractional coordinate)
 #EF_CORIGIN 0 0 0  # (in cartesian coordinate)
END EFIELD
```
EFIELD setup example

WEIGHT Setting of weight factor for the fitting procedures.

KRANGE *integer* : range of k-point where the weight factor is applied

TBABND *integer* : range of eigen states of the tight binding calculation

DFTBND *integer* : range of eigen states of the target energy bands

WEIGHT *real* : weighting factor

ORBT_I *ineteger* : orbital index. n^{th} orbital states will get a penalty

SITE_I *ineteger* : site index. ORBT_Ith orbital state at SITE_I atom will get a penalty. This prohibit certain orbital character to be stabilized from the fitting procedures.

```
SET WEIGHT
  KRANGE  :            TBABND  :    DFTBND IBAND:FBAND WEIGHT 1
  KRANGE  :            TBABND 17:20 DFTBND 17:20       WEIGHT 6
  KRANGE 20:60 100:140 TBABND 17:20 DFTBND 17:20       WEIGHT 20
  KRANGE 1     TBABND  7    ORBT_I 1  SITE_I Mo1 PENALTY 200
END WEIGHT
```

WEIGHT setup example

Note 1: After fitting procedure is over, the WEIGHT information is written in PFILE so that one can restart fitting. To use WEIGHT information written in PFILE, one can add USE_WEIGHT tag in your PFILE tag. Please find this information in PFILE tag.

CONSTRAINT TBPARAM Setting for parameter constraints for the fitting and calculation. The value of the specified two parameter will be kept same during the fitting and tight-binding calculations. If you are using GA method for the fitting procedures (LSTYPE), you are encouraged to give upper bound and lower bound for each parameters to minimize parameter search field in the randomize procedures of GA method. The default lower/upper bound for every parameter is -20.0/20.0.

```
SET CONSTRAINT
    e_py_S  = e_px_S  # e_py_S is enforced to be same as e_px_S.
    e_px_S  <=  5.0  # upper bound for e_px_S (applied in GA)
    e_px_S  >= -5.0  # lower bound for e_px_S (applied in GA)
END CONSTRAINT
```

CONSTRAINT setup example

If the second argument '=' is replaced by '==' and the third argument is not present, then this parameter will not be fitted and its initial guess as defined in PFILE will be fixed during the fitting procedures. Note that, exactly same effect can be achieved by putting 'FIXED' tag at the parameter specification line of the PFILE, and the detailed explanation can be found in Fixing parmeter of Sec.4.

Note: Some special rule can be applied in parameter. For example, assuming that s and p_x orbital are in atom A and B. Then, one can consider hopping parameter such as sps_1_AB. In this case, usually, hopping between orbital s in A and orbital p_x in B is differ from hopping between orbital p_x in A and orbital s in B since atom A and atom B is different species. [see section 4.Details of the format of PFILE] However, in many reason, sometimes it is useful to distinguish A and B although they are same species, for example, S atom of 2H-MoS2 monolayer. Here, one can specify S atom in upper layer to Mo layer as Sa and S atom in lower layer to Mo layer as Sb. In this case, to correctly define hopping between s and p orbitals in Sa and Sb, one should provide following hopping parameter separately: sps_1_SaSb, pss_1_SaSb. It is bothering to write same values twice with just changing sp to ps.

And it should be specified in the constraint section that $pss\_1\_SaSb = sps\_1\_SaSb$. It can be avoided if one provide constraint statement that atom Sb and atom Sa are actually same species as follows:

```
SET CONSTRAINT
    ....
    ....
    atom_Sb  = atom_Sb
    ....
    ....
END CONSTRAINT
```

CONSTRAINT setup example

Now, it is enough to provide sps_1_SaSb only for the s and p orbital hopping parameters between Sa and Sb atom.

NN_CLASS Setting for nearest neighbor set up.

If the distance between two atomic species (For example, `Mo` and `S`) are 1st nearest type, and its upper limit is 3.2 angstrom (e.g., below this value will be regarded as the pair), thene we can set as follows,

```
Mo-S : 3.2  R0 3.171634
```

Here, number of dash '-' occurance between two atomic species indicates the distance class n, and the above example represents 1st nearest hopping between `Mo` and `S`. The following `R0` tag defines optimal bonding distance between two neighbor pair. This value will be used in calling the scaling function to get the distance dependent hopping parameter.

```
SET NN_CLASS
    Mo-Mo  : 3.2   R0 3.171634
    S-S    : 3.28  R0 3.171634
    S--S   : 3.2   R0 3.193724
    Mo-S   : 2.5   R0 2.429624
END NN_CLASS
```

NN_CLASS setup example

RIBBON Setting for nanoribbon calculations.

At the initial stages of the calculations, TBFIT will generate GFILE`-ribbon` with the settings bellow.

NSLAB *integer* : multiplication of unitcell along each direction

VACUUM *real* : vacuum spacing along each direction.

KFILE_R *real* : KFILE for ribbon band structure. Default: KFILE

PRINT_ONLY_R *logical* : if `.TRUE.` the geometry file will be generated with `-ribbon` suffix to the GFILE and the program will imedietly stops. Default: `.FALSE.`

```
SET RIBBON
    NSLAB         1 20  1
    VACUUM        0 20  0
    KFILE_R       KPOINTS_RIBBON
    PRINT_ONLY_R .FALSE. or. TRUE.
END RIBBON
```
Ribbon calculation setup

Z2_INDEX Automatic calculations for topological index $[\nu_0\ \nu_1, \nu_2, \nu_3]$ for 3D or $\mathbb{Z}_2$ for 2D via WCC method. The output will be written at `Z2.WCC.plane_index.dat` and `Z2.GAP.plane_index.dat`. Here, `plane_index` indicates one of six B_i-B_j plane with $B_k = 0$ or π. For example, if `plane_index` = `0.0-B3.B1_B2-PLANE`, then it contains WCC information of $B1$-$B2$ plane with $k_z = \pi$.

```
SET Z2_INDEX
    Z2_ERANGE    1:28  # upto occupied
    Z2_DIMENSION 3D  # or 2D:kz (2D WCC plane perpendicular to kz)
    Z2_NKDIV  21 21 # k-grid for KPATH and k-direction for WCC
    Z2_CHERN  .TRUE. # 1st Chern number of given bands with ERANGE
END Z2_INDEX
```
Z2 index calculation using WCC method

WCC Wannier Charge Center calculation settings

```
SET WCC
    WCC_ERANGE 1:28  # upto occupied
    WCC_FNAME  WCC.OUT.dat
    WCC_FNAME_GAP WCC.GAP.dat  # largest gap will be written
    WCC_KPATH  0 0 0  1 0 0 # k_init -> k_end (ex, along b1)
    WCC_KPATH_SHIFT  0 0 0.5 # kpoint shift along b3 direction
    WCC_DIREC  2 #k-direction for WCC evolution (1:b1,2:b2,3:b3)
    WCC_NKDIV  21 21 # k-grid for KPATH and k-direction(odd number)
    WCC_CHERN  .TRUE. # 1st Chern number of given bands with ERANGE
END WCC
```
Wannier charge center (WCC) setup: kz 0.5 (shift)

ZAK_PHASE Setting for Zak phase calculations.

```
SET ZAK_PHASE
    ZAK_ERANGE 1:28  # upto occupied
    ZAK_FNAME  ZAK_PHASE.OUT.dat
    ZAK_KPATH  0 0 0  1 0 0 # k_init -> k_end (ex, along b1)
    ZAK_DIREC  2 #k-direction for Zak phase evolution (1:b1,2:b2,3:b3)
```

```
    ZAK_NKDIV  21 21 # k-grid for KPATH and k-direction
END ZAK_PHASE
```
Zak phase setup

BERRY_CURVATURE Setting for Berry curvature calculations.

```
SET BERRY_CURVATURE
    BERRYC_METHOD KUBO # .or. RESTA(not yet supported)
    BERRYC_ERANGE 17:18
    BERRYC_FNAME  BERRYCURV.17-18 # output will be BERRYC_FNAME.dat
    BERRYC_DIMENSION 2D:B3  # 2D plane perpendicular to kz)
END BERRY_CURVATURE
```
Berrycurvature setup

PARITY_CHECK Setting for Parity eigenvalue calculations for given k-points.

```
SET PARITY_CHECK
    PARITY_KP 0.0 0.0 0.0  G    # Gamma (reciprocal unit)
    PARITY_KP 0.5 0.0 0.0  M1   # M1    (reciprocal unit)
    PARITY_KP 0.0 0.5 0.0  M2   # M2    (reciprocal unit)
    PARITY_KP 0.5 0.5 0.0  M3   # M3    (reciprocal unit)
    PARITY_ORIGIN 0.0 0.0 0.0 # origin of the system (direct coord)
    PARITY_OP1 -1  0  0 # Rotation matrix (R) for inversion
    PARITY_OP2  0 -1  0 # => R*X=-X (invert coordinate)
    PARITY_OP3  0  0 -1 # => X:direct coord ; R: integer 3x3 array
    NOCC  10 # 10 occupied states
    PRINT_HAMILTONIAN  .FALSE. # print out H(k) for each k-point
END PARITY_CHECK
```
Parity check setup

Note:
• You can add (or remove) PARITY_KP tag if you want to get the parity information for another TRIM (time reversal invariant momenta: -k=k+G) point.
• To use this functionality and to get the meaningful results, your system should have inversion symmetry.
• The ROTATION tag is optional, the default is

$$R = \begin{bmatrix} PARITY\_OP1 \\ PARITY\_OP2 \\ PARITY\_OP3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

.

SYMMETRY_EIG Setting for Symmetry eigenvalue calculations for given k-points.

```
SET SYMMETRY_EIG
    SYMMETRY_KP 0.0 0.0 0.0  G    # Gamma (reciprocal unit)
    PARITY_KP 0.6666667 0.6666667 0.0  M3   # M3  (reciprocal unit)
```

```
    SYMMETRY_ORIGIN 0.0 0.0 0.0 # origin of the system (direct coord)
    SYMMETRY_OP1 -1 -1  0 # Rotation matrix (R) for 2/3pi
    SYMMETRY_OP2  0  1  0 #
    SYMMETRY_OP3  0  0  1 #
    ROT_ANGLE  120 # rotation angle counterclockwise
    NOCC  10 # 10 occupied states
    PRINT_HAMILTONIAN  .FALSE. # print out H(k) for each k-point
END PARITY_CHECK
```
Parity check setup

EFFECTIVE Setting for evaluating effective hamiltonian and its eigenvalues. This job will be performed by the Löwdin downfolding technique[11]. Note that in the current version, only the *energy*-dependent effective Hamiltonian will be constructed, where the *energy* is representing one's interested region.

```
SET EFFECTIVE HAM
    EFF_ORB C:pz # downfolding will be performed on these orbitals
    EFF_EWINDOW -2:2 # energy window of interest (e_center=-2+2/2)
END EFFECTIVE HAM
```
Effective Hamiltonian construction

REPLOT Setting for replotting where DOS/LDOS/PBAND represents density of states (DOS), local density of states (LDOS), and atom-projected band structure (PBAND) by reading `band_structure.dat` or `band_structure.up(dn).dat` file, respectively.

```
SET REPLOT
  FILE_FORMAT bin  # let code know what kind of
                   # band_structure_TBA file to be read.
                   # if bin : band_structure_TBA.(up/dn).bin
                   # if dat : band_structure_TBA.(up/dn).dat
                   # will be loaded. Default: dat
  REPLOT_BAND .TRUE. wf # replot band structure
     # with wavefunction coefficient (with tag wf). Instead of
     # wf tag, one can also request rh, mx, my, mz, where the
     # meaning of each tag can be found in LORBIT tag.
     # If no additional tag is specified, set to rh by default.
     # If no tag is specified, only energy band without orbital
     # information will be printed out. The output file will be
     # written in band_structure_TBA.replot_wf.(up/dn).dat file
     # with ascii format.
     # Note that to activate this tag, FILE_FORMAT tag should be
     # set to bin and corresponding file band_structure_TBA.replot
     # .(up/dn).bin should be exist in the folder.
```

[11] P.-O. Löwdin, J. Chem. Phys. 19, 1396 (1951)
E. Zurek, O. Jepsen, O. K. Anderson, Chem. Phys. Chem. 6, 1934 (2005)

```
  REPLOT_PROJ_BAND .TRUE. 1:8 12 # replot projected band structure
                                 # for given atoms and their sum
  REPLOT_DOS .TRUE. # Recalculate DOS using pre-calculated
                    # band_structure file, e.g.,
                    # band_structure.dat (nonmagnetic or
                    # non-collinear) or band_structure.up/dn.dat
                    # (spin polarized case).
  REPLOT_LDOS .TRUE. 1:8 12 # Print local density of states for
                            # given atoms. For example, here,
                            # 1st to 8-th atoms and 12-th
                            # atoms will be resolved.
  REPLOT_DIDV .TRUE. # Print local density of states for given
                     # discrete energy level within DOS_EWINDOW.
                     # The outputs will be stored in
                     # ./didv/DIDV.replot.dat.#i where #i=[1:NEDOS]
  SMEARING 0.03 # gaussian smearing. Default = 0.025
  NEDOS    2000 # number of grid points in energy window (erange)
  DOS_EWINDOW -20.0:10.0 # energy window to be plotted

  REPLOT_SLDOS .TRUE. # spatial LDOS within EWINDOW
  REPEAT_CELL  20 20 1 # if REPLOT_SLDOS = .TRUE.,
                       # cell periodicity for visualization
  RORIGIN  0.0 0.0 0.0 # shift of origin of atomic coordinates
                       # (fractional to unit vector a1, a2, a3,
                       # respectively).
  BOND_CUT 1.8  # bond length <= bond_cut will not be written in
                # BOND.replot.dat Default: 3.0 (ang)
END DOS
```

DOS setup example

Note 1: This tag is useful if you want to get series of DOS/LDOS calculations with respect to the SMEARING and energy window. If this SET is activated, other calculations will be ignored and the program stops immediately after the calculation.

Note 2: If REPLOT_DOS = .TRUE. or REPLOT_LDOS = .TRUE. or REPLOT_SLDOS = .TRUE., then it is encouraged to set SMEARING, NEDOS, DOS_EWINDOW, otherwise the default values will be used.

Note 3: If REPLOT_SLDOS = .TRUE., then, REPEAT_CELL, RORIGIN, BOND_CUT should be set together, otherwise the default values will be used.

Note 4: The output file names are as follows:
• if REPLOT_DOS = .TRUE. → DOS.replot.dat
• if REPLOT_LDOS = .TRUE. → LDOS.replot.ATOM_INDEX.dat, where ATOM_INDEX is the atom number. The example can be found in the "Example/Graphene/DENSITY_OF_STATE/replot" of your example folder.

• if REPLOT_DOS = .TRUE. → SLDOS.replot.dat and BOND.replot.dat.
The example can be found in the "Graphene/QSH/NANORIBBON/zigzag_ribbon/SPATIAL_LDOS_REPLOT" of your example folder.

Note 5: The multiple declaration of REPLOT_BAND, REPLOT_PROJ_BAND and REPLOT_LDOS tag result in multiple band_structure_TBA.replot_??.dat, band_structure_TBA_atom.sum#.dat and LDOS.replot.sum#.dat files, respectively, where ?? represents one of {rh, mx, my, mz, wf} tag you specified and # represents the order of declaration of the tag. The concept of multiple declaration is same as in PROJ_BAND tag.

Note 6: The purpose of REPLOT_BAND tag is clear as it only read .bin file and write .dat. That is, for converting binary file format into human readable ascii format. PROJ_BAND.

Note 7: If you add bin or bin4 tag at the end of REPLOT_BAND tag, for example,

```
REPLOT_BAND .TRUE. rh bin (or bin4)
```

then the result will be written with unformatted format with single precision. This is only available if your target file is already written in bin (or bin4 ; single precission) (see the LORBIT tag for the details). Using this tag, ascii formatted file can be saved in unformatted file, e.g., (band_structure_TBA.dat → band_structure_TBA.bin).

3. Details of the format of GFILE

Atomic orbital setup *string*

Hydrogen-like atomic orbital can be specified for the orbital basis. The possible orbital basises are[12]:

```
s px py pz dz2 dxy dx2 dxz dx2
```

```
0 0.0 0.0 s px py pz  # s, px, py, and pz orbitals at ATOM_A
0 0.0 0.5 s px py pz  # s, px, py, and pz orbitals at ATOM_B
```

setup of atomic orbital basis in GFILE

Custumized atomic orbital setup *string*

If someone does not want to use Slater-Koster type interatomic hopping parameter, customized atomic orbital can be defined instead. In this case, distance and hopping pair dependent parameterization should be properly defined in the

[12]Please note that current version does not support the f orbitals. However, we will include f in the future release of TBFIT. For the Slater-Koster tables of f orbitals, please see [K. Lendi, Phys. Rev. B 9, 2433 (1974)].

PFILE.

Warning : This is experimental feature and on the development stages (do not use).

```
0 0.0 0.0 cp1  # cp1 orbital at ATOM_1
0 0.0 0.5 cp1  # cp1 orbital at ATOM_2
```

setup of custumized atomic orbital name *cp*1

MOMENT tag *real*

Magnetic moment for Each atomic orbital can be assigned as follows,

collinear case: `0.0`

noncollinear case: `0.0 0.0 0.0` [M θ ϕ]

```
0 0.0 0.0 px py pz moment 0 0  1 # spin-up for pz
0 0.0 0.5 px py pz moment 0 0 -1 # spin-dn for pz
```

usage of *moment* tag in GFILE with *collinear* magnetism

```
0 0.0 0.0 px py pz moment 0 0 0  0 0 0  1 0 0  # spin-up for pz
0 0.0 0.5 px py pz moment 0 0 0  0 0 0 -1 0 0  # spin-dn for pz
```

usage of *moment* tag in GFILE with *noncollinear* magnetism

MOMENT.C tag *real*

Similar to MOMENT but in `noncollinear` case, the 1^{st}, 2^{nd}, and 3^{rd} value represents, m_x, m_y, and m_z, respectively. Here, x, y, and z represents the cartesian axis.

noncollinear case: `0.0 0.0 0.0` [M_x M_y M_z]

```
0 0.0 0.0 px py pz moment.c 0 0 0  0 0 0  0 0  1 # spin-up for pz
0 0.0 0.5 px py pz moment.c 0 0 0  0 0 0  0 0 -1 # spin-dn for pz
```

usage of *moment.c* tag in GFILE with *noncollinear* magnetism

4. Details of the format of PFILE

ONSITE parameters *real*

Onsite prameters for each atomic orbital should have the prefix `e_` and joint with the name of the orbital. The suffix should be the atomic species where the orbital placed.

```
e_dx2_Mo      -0.34
```

HOPPING parameters *real*

The tight binding hopping parameter used in the calculations.

*case*1.) IS_SK `.TRUE.`

In this case, `Slater-Koster` type parameter should be specified properly. The syntax is as follwos:

```
hopping-type_nn-class_AB
```

`hopping-type` will have one of following prefix: { *ss*, *sp*, *sd*, *pp*, *pd*, *dd* }, and one of following suffix: {*s*, *p*, *d* }, which implies σ-, π-, and δ-type inteaction. `nn-class` specifies the distance class. See NN_CLASS for the details. `AB` specifies the two atomic species (`A` and `B` atoms) where the orbital hopping take place. For example, for the $dd\delta$ `Slater-Koster` parameter involved with the hopping process between the d_{z2} orbital in `Mo` atom and d_{yz} orbital in `Mo`, and they are 2^{nd} neighbor pair, then the parameter should be the following form:

```
ddd_2_MoMo    -0.2
```

`IMPORTANT!` If atom `A` and `B` is different species, special care should be taken in naming parameter. For example, assume that each atom `A` and `B` has orbital *s*, p_x. Then one can consider σ-type first nearest neighbour hopping between *s* orbital of `A` and p_x orbital of `B`. In this case, such hopping parameter should be written as: *sps*_1_*AB* or *pss*_1_*BA*. On the other hand, if we consider the hopping between p_x orbital of `A` and *x* orbital of `B`, the hopping parameter for this case should be written as: *sps*_1_*BA* or *pss*_1_*AB*.

```
Example s and p orbital with different atomic species:
atom A orbital s ; atom B orbital px => sps_1_AB or pss_1_BA
atom A orbital px ; atom B orbital s => pss_1_AB or sps_1_BA
```

For the overlap integral parameters, one can add `o_` prefix, for example,

```
o_ddd_2_MoMo    -0.2
```

Note that to activate the use of overlap integral, i.e., assuming the use of non-orthogonal basis, one should put `USE_OVERLAP .TRUE.` tag into your PFILE.

To activate the use of scaling functions, i.e., distance dependent rescaling of parameters, one should set corresponding scaling parameters, for example,

```
s_ddd_2_MoMo    -0.2
```

for the `ddd_2_MoMo` parameter. Note again that to apply same rule for the overlap integral parameters, one should put `os_` prefix instead, for example,

```
os_ddd_2_MoMo    -0.2
```

*case*2.) IS_SK .FALSE.
Warning : This is experimental feature and on the development stages (do not set to .FALSE.).
In this case, the customized atomic orbital is assumed and the following scheme should be applied:

```
hopping-type_nn-class_AB
```

Here, the basic structure is same as *case*1.), however, the syntax of hopping-type is slightly different. That is: the prefix should have *cc* since this indicates *customized* hopping parameters. For the suffix, one should put user defined letter that characterize the hopping. For example,

```
cca_2_BiBi       0.01
```

represents the hopping between $2^n d$ neighbor Bi atoms with the '*a*' type of *rule* which characterizes hopping pair. If you want to setup the *rule*, you have to write the conditions to the source code: get_cc_param.f90.

```
# SET UP THE 'USER' DEFINED HOPPING 'RULE'
# NOTE: In THIS example, hopping between Bi-Bi atom along
# x-direction characterized by hopping distance at around
# 8.6 Å (cca_2_BiBi) with nn_class = 2 will be considered.
# Following 'if' routine  will find the parameter named
# cca_2_BiBi in the 'PFILE' and will asign its number as
# the 'parameter_index'.

      [...]
  elseif( (dij .gt.  8.5) .and. (dij .lt.  8.7) .and. &
          (ci_atom .eq. cj_atom) ) then
        call get_param_name(cc_custom, param_class, 'a', &
                            nn_class, ci_atom, cj_atom, &
                            flag_scale)
      [...]
```

source code example: get_cc_param.f90

LOCAL POTENTIAL: LOCAL.POT parameters *real*

If you want to apply local potential to the particular atomic site or particular orbital, then you can simply turn on LOCCHG (.TRUE.) and write local.pot tag together with the amount of local potential to be applied for each atomic orbitals in the GFILE. Next, you have to provide proper scaling parameter (U^i_{onsite}) for the local potential, since the local potential is applied on your Hamiltonian

as: $e'^{i}_{onsite} = e^{i}_{onsite} + e^{i}_{loc.pot} \times U^{i}_{onsite}$, i.e., it modifies onsite energy e^{i}_{onsite} to e'^{i}_{onsite}. Here, U^{i}_{onsite} should be defined in your PFILE so that the syntax is `local_U_orbital-type_atom-name`. `orbital-type` is one of s, p, or d type of orbital and `atom-name` is the name of atomic species you want to apply the local potential.

```
0 0.0 0.0 px py pz local.pot  1  1  1  # positive loc.pot
0 0.0 0.5 px py pz local.pot -1 -1 -1  # negative loc.pot
```

example of `local.pot` tag in GFILE

```
local_U_p_S  1.0
```

example of `local.pot` parameter in PFILE

SOC parameters *real*

*case*1.) IS_SK `.TRUE.`
Every spin-orbit coupling parameters in `Slater-Koster` method should have the prefix with `lambda_` and proper orbital information p_(as a joinder, for example p orbital) and species information `_S`(as a suffix, for example `Sulpur` atom) to precisely indicating the atomic orbital where the SOC effect will be applied.

```
lambda_p_S    0.2
```

*case*2.) IS_SK `.FALSE.`
In the case of user defined hopping parameter (orbital prefix start with c, see Sec.3 for the details) has been defined in the GFILE, *SOC* can be considered by setting up the Rashba and in-plane spin-orbit interaction. For Rashba type *SOC*, the prefix `lrashba_` should be joint with nearest neighbor class n and hopping pair as follows.

```
lrashba_c_2_BiBi    0.2
```

Above setting represents, Rashba type spin-orbit coupling between the custum type orbitals with c-prefix of the atom `Bi` and `Bi`.

Fixing parmeter
If one want some parameters not to be fitted during the fitting procedures, one can fix thoes parameters by adding `FIXED` or `F`. For example, if you want `lambda_p_S` to be kept as its initial value, then, set this parameter as follows,

```
lrashba_c_2_BiBi    0.2 FIXED
```

Example of PFILE

```
      e_dz2_Mo      -0.34636955
      e_dx2_Mo      -0.70447045
      e_dxy_Mo      -0.70447045
      e_dxz_Mo      -0.17913534
      e_dyz_Mo      -0.17913534
        e_pz_S      -2.96500556
        e_px_S      -1.47877518
        e_py_S      -1.47877518
         e_s_S     -10.51138070
    dds_1_MoMo      -1.04598377
    ddp_1_MoMo       0.44731993
    ddd_1_MoMo       0.10237760
      pps_1_SS       0.62323972
      ppp_1_SS       0.03251328
     pds_1_MoS      -2.32384045
     pdp_1_MoS       0.97229680
      sss_1_SS      -0.57287106
      sps_1_SS      -0.33278732
      sss_2_SS      -0.45573348
      sps_2_SS      -0.21906117
     sds_1_MoS       2.66111706
   lambda_d_Mo       0.08014531  Fixed
    lambda_p_S       0.07567002  Fixed
```

example of PFILE: PARAM_FIT.dat for MoS_2 (IS_SK .TRUE.)

```
          e_cp1_Bi    -0.09222821
        ccb_1_BiBi     0.01723235
        cca_2_BiBi     0.13290800
        ccy_3_BiBi    -0.0
        ccx_4_BiBi    -0.03544401
   lrashba_c_1_BiBi   -0.01119142
   lrashba_c_2_BiBi    0.04914549
   lrashba_c_3_BiBi   -0.00632175
   lrashba_c_4_BiBi   -0.00636364
```

example of PFILE: PARAM_FIT.dat for Bi/Si(110) (IS_SK .FALSE.)

Part II.
Examples

1. Graphene with s and p orbitals

1.1. Parameter fitting

1.1.1. Prepare main control file: INCAR_TB

TBFIT reads INCAR_TB in the initial stages of the calculations to set up basic control parameters. In addition, GFILE, KFILE, and PFILE is mandatory for the normal run, and it should be predefined in your INCAR_TB. For the fitting procedures, TBFIT should be `.TRUE.`. Then, one has to choose the fitting algorithm LSTYPE among `LMDIF` and `GA`. In this example, we will take `LMDIF` as a fitting algorithm. For the `GA` method, please find the example in "`Example/Graphene/BAND_FIT/Step_1.genetic_algorithm/`" of your example folder.

The users must pay some time to "SET WEIGHT" which is quite important in fitting procedures. If you want to fit more tightly than the other regions of your target band structure, you can put much higher value for that particular region. In this example, we have 16 atomic orbitals (2 atoms $\times$ 4 orbitals per atom $\{s, p_x, p_y, p_z\}$ $\times$ 2 spinors) in total. The target band structure, which is calculated by VASP, also have 16 band structure. In the INCAR_TB file below, we give 5 weights.

The first line,

```
KRANGE : TBABND : DFTBND 1:16 WEIGHT 1
```

indicates that the all the k-range (:), and all the tight binding band structure (:), and first sixteen bands (1:16) of DFT target band structure will be weighted with "1".

And the second line,

```
KRANGE : TBABND 1:8 DFTBND 1:8 WEIGHT 7
```

indicates that the all the k-range (:), and first eight tight binding bands (1:8) will be targeted to the first eight (1:8) of DFT band structure with weight of "7". Hence, the valence bands will be much more tightly fitted.

And the third line,

```
KRANGE 10:51 TBABND 5:10 DFTBND 5:10 WEIGHT 27
```

indicates that the particular k-range (10:51, around M-K rigion), and 5-th to 10-th tight binding bands (5:10) will be targeted to those of DFT band structure with much larger weight of "27". The bands near the Fermi level and near the Dirac point will be fitted tightly. For the other lines, the same rules can be applied.

Next, the basic control tags should be informed. For example, MITER, PTOL, FTOL, PFILE, POFILE, EFILE ..., etc.

And then, one should specify whether your system is one of followings: *nonmagnetic*, *noncollinear*, *collinear*, which can be defined by TYPMAG, together with LSORB which is for the spin-orbit coupling effect. If you have put some LOCAL.POT tag in your PFILE to consider some local potential, you should turn on LOCCHG to `.TRUE.`, and add some relevant parameters (see LOCAL.POT for the details) into the PFILE to manage the strength of the local potential. The tag IS_SK is to make sure that the hopping integral evaluation will follow the Slater and Koster's rule[13]. Note that in the current version, IS_SK = `.FALSE.` is experimental and under developing for the ease of use, so recommend to leave it to `.TRUE.`.

Now you have to make some rule for the nearest neighbor (nn) hoppings by setting up "SET NN_CLASS" which defines interatomic hopping. Once you set nn pair, the necessary hopping parameter for those hopping will be automatically determined and TBFIT will trying to find those parameters in your PFILE. The detailed syntax for the naming of hopping parameters can be found in Section-4: PFILE-detail.

Finally, with "SET CONSTRAINT TBPARAM" tag, one can add some constraint rule between parameters or restrict the magnitude of the parameters by applying upper/lower bound to them. For example, here, we have assumed that every p orbitals shall have same onsite energy. So by setting "`e_px_C = e_pz_C`", during the calculations, `e_px_C` value will be kept same as `e_pz_C`.

INCAR_TB : control tags

```
TBFIT    .TRUE.
LSTYPE   LMDIF
MITER    300
PTOL     0.000000001
FTOL     0.000000001

GFILE    POSCAR-TB
KFILE    KPOINTS_BAND
PFILE    PARAM_FIT.dat
EFILE    DFT_BAND.dat
POFILE   PARAM_FIT.new.dat

LOCCHG   .TRUE.
TYPMAG   noncollinear
LSORB   .TRUE.
LORBIT  .FALSE.

IS_SK    .TRUE.
```

[13] J. C. Slater and G. F. Koster,''Simplified LCAO Method for the Periodic Potential Problem'', Phys. Rev. 94, 1498 (1954)

```
SET WEIGHT
    KRANGE  :          TBABND :       DFTBND 1:16  WEIGHT 1
    KRANGE  :          TBABND 1:8     DFTBND 1:8   WEIGHT 7
    KRANGE  10:51      TBABND 5:10    DFTBND 5:10  WEIGHT 27
    KRANGE  20:40      TBABND 1:4     DFTBND 1:4   WEIGHT 50
    KRANGE  1:8 53:80  TBABND 3:6     DFTBND 3:6   WEIGHT 10
END WEIGHT

SET NN_CLASS
    C-C   : 1.8   R0 1.4145
END NN_CLASS

SET CONSTRAINT TBPARAM
    e_px_C    = e_pz_C
    e_py_C    = e_pz_C
     e_s_C   >= -10
    e_px_C   >= -10
    e_py_C   >= -10
    e_pz_C   >= -10
  sss_1_CC   >= -10
  sps_1_CC   >= -10
  pps_1_CC   >= -10
  ppp_1_CC   >= -10
lambda_p_C   >= -10
     e_s_C   <= 10
    e_px_C   <= 10
    e_py_C   <= 10
    e_pz_C   <= 10
  sss_1_CC   <= 10
  sps_1_CC   <= 10
  pps_1_CC   <= 10
  ppp_1_CC   <= 10
lambda_p_C   <= 10
END CONSTRAINT TBPARAM
```
example of INCAR-TB for Graphene

1.1.2. Prepare parameter file: PARAM_FIT

PARAM_FIT.dat : PFILE

```
PRINT_INDEX = .TRUE.  # control tag for indexing parameters
                      # (not affected on your actual calculations)
USE_OVERLAP = .TRUE. # control tag for the use of overlap integral
                     # parameters that specified with "o_" prefix.
```

```
# index       param_name         parameter
 1                 e_s_C       -5.50250547
 2                e_px_C        0.27589107
 3                e_py_C        0.27589107
 4                e_pz_C        0.27589107
 5              sss_1_CC        4.14407741
 6              sps_1_CC        5.06877443
 7              pps_1_CC       -4.33297737
 8              ppp_1_CC        2.54569565
 9            o_sss_1_CC        0.02586060
10            o_sps_1_CC       -0.04261519
11            o_pps_1_CC        0.05063120
12            o_ppp_1_CC        0.01711425
13          lambda_p_C         0.00000000
```
example of PFILE for Graphene

POSCAR-TB : GFILE

```
# Graphene with honeycomb lattice
  1.00000000000000
  2.45                   0.0000000000000000    0.0
  1.2250000000000000    2.1217622392718746    0.0
  0.0000000000000000    0.0000000000000000   15.0
  C
  2
  Direct coordinate
  0.16666666667  0.16666666667   0.0   s  px  py  pz
 -0.16666666667 -0.16666666667   0.0   s  px  py  pz
```
example of GFILE for Graphene

Here, we have set s and p (p_x, p_y, and p_z) orbital basis for each carbon atom.

KPOINTS_BAND : KFILE

```
# k-points along high symmetry lines
  20  # number of division between each line segment.
  Line-mode
  Reciprocal
  0 0 0                                              G
  0.50000000  0.500000000 0                          M

  0.50000000  0.500000000 0                          M
  0.33333333333333333 0.66666666666666667 0 K

  0.33333333333333333 0.66666666666666667 0 K
  0.00000000  0.000000000 0                          G
```

```
0.00000000  0.000000000 0                    G
0.00000000  0.000000000 0.5                  A
```
example of KFILE for Graphene

Note that the file syntax is exactly same as KPOINTS file of VASP program. See here for the details.

DFT_BAND.dat : EFILE

For the details of the EFILE, please go to EFILE section and also please find the example file in "Example/Graphene/BAND_FIT/Step_2.lmdif_method/" of your example folder.