

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

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

January 18, 2019

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 referred. If the system size is very big, you can calculate band structure with energy window constraint. This is available with ?? 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 < /dev/null | tee log.out

```

Note that the output log will be written in `log.out` file.

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 ?? is set for **LSTYPE**, MITER represents the maximum number of generations.

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 Algorithm³ based on PIKAIA library^{4,5,6,7}. To setup control parameters for the GA, see Sec. ??.

PTOL & FTOL *real* Default: 0.00001

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

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>

K_UNIT *string* Default: `ANGSTROM`

`ANGSTROM` : the unit of the k -point will be written in \AA^{-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.

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.

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

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.

```

MoS2 # comment
1.0000000000000000 # 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: MoS₂ with Mo-*d* and S-*sp*

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*

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_POTENTIAL](#) tag in your [GFILE](#). For the details, see the explanation of [LOCAL_POTENTIAL](#) 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* 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 m_x or m_y or m_z next to the logical text with .TRUE., then, corresponding magnetization values will be printed out instead of local orbital contribution. For example,

```
LORBIT .TRUE. 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
```

LDOS *logical* Default: .FALSE.

Setting tag for orbital/atom decomposed output. The output will be written in separate file for each atom. The format is same as `band_structure_TBA.dat` but specified with atom index, for example,

```
band_structure_atom.1.dat
band_structure_atom.2.dat
...
```

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	1	0.0	0.0	0.0	...	s	...	s	...	-4.0	...
1	1	1	0.0	0.0	0.0	...	s	...	px	...	0.0	...
1	1	1	-1.2	-0.7	0.0	...	s	...	s	...	-3.9	...
1	2	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	1	0.0	0.0	0.0	...	s	...	s	...	-2.0	...
1	1	1	0.0	0.0	0.0	...	s	...	px	...	0.0	...
1	1	1	-1.2	-0.7	0.0	...	s	...	s	...	-3.9	...
1	2	2	-1.2	-0.7	0.0	...	s	...	px	...	1.9	...
...												
...												

_____ hopping_modified.dat example file _____

IBAND *integer* Default: 1

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

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 $\text{EDFT}(n, k)$ will be shift by amount of the scissor operation. This operation works as follows: $E'_{\text{target}}(n, k) = E_{\text{target}}(n, k) + e_{\text{scissor}}$ if $n \geq i_{\text{scissor}}$. Note that this operation is only valied if **TBFIT** is **.TRUE..**

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

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 4400th to 4700th 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
```

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 $1.5 \times \text{NE}$ ⁸. 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.⁹

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**

⁸Eric Polizzi, “Density-matrix-based algorithm for solving eigenvalue problem” *Physical Review B* 79, 115112 (2009)

⁹Though, 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.

Library (MKL), please make sure that your library path is properly assigned.
(suggest to use MKL version ≥ 11.3)

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

SET *string*

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

?? : for Genetic Algorithm setting

CONSTRAINT TBPARAM

NN_CLASS

RIBBON

BERRY_CURVATURE

ZAK_PHASE

WCC

Z2_INDEX

PARITY_CHECK

EFIELD

WEIGHT

DOS

EIGPLOT

STMPLOT

??

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
  CONVTOLE 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 will not be considered to calculate.
    # Try this option if you have very large cell and you are
    # especially interested unitcell center.
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/Å)
  SMEARING 0.03 # gaussian smearing. Default = 0.025
  NEDOS 2000 # number of grid points in energy window (erange)
  DOS_ERANGE -20.0:10.0 # energy window to be plotted
  DOS_NERANGE 1:NEIG # energy window to be calculated (integer)
  DOS_FNAME DOS_TB_projected.dat # output file name for DOS output
END DOS
_____ DOS setup example _____

```

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

SITE_I *integer* : 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

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 ?? 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 ?? method. The default lower/upper bound for every parameter is -20.0/20.0. Note that imposing the upper/lower bound for the parameter is not supported for ?? method in the current version.

```

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 ??)
e_px_S >= -5.0 # lower bound for e_px_S (applied in ??)
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 parameter](#) of Sec.4.

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), then we can set as follows,

Mo-S : 3.2 R0 3.171634

Here, number of dash '-' occurrence 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
  BERRYCURV_METHOD KUBO # .or. RESTA(not yet supported)
  BERRYCURV_ERANGE 17:18
  BERRYCURV_FNAME BERRYCURV.17-18 # output will be BERRYCURV_FNAME.dat
  BERRYCURV_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)
ORIGIN_SHIFT 0.0 0.0 0.0 # origin of the system (direct coord)
ROTATION1 -1 0 0 # Rotation matrix (R) for inversion
ROTATION2 0 -1 0 # => R*X=-X (invert coordinate)
ROTATION3 0 0 -1 # => X:direct coord ; R: integer 3x3 array
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

$$R = \begin{bmatrix} ROTATION1 \\ ROTATION2 \\ ROTATION3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

EFFECTIVE Setting for evaluating effective hamiltonian and its eigenvalues. This job will be performed by the Löwdin downfolding technique¹⁰. 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 _____

```

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 bases are¹¹:

s px py pz dz2 dxy dx2 dxz dx2

¹⁰ P.-O. Löwdin, *J. Chem. Phys.* **19**, 1396 (1951)

E. Zurek, O. Jepsen, O. K. Anderson, *Chem. Phys. Chem.* **6**, 1934 (2005)

¹¹Please 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)].

```
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](#) _____

Customized 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 [PFILE](#).

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

_____ setup of customized atomic orbital name *cp1* _____

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 1st, 2nd, and 3rd 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 0 0 0 0 0 0 0 1 # spin-up for pz
0 0.0 0.5 px py pz moment 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 parameters 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.

case1.) `IS_SK .TRUE.`

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

```
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 interaction. `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 δ* 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 2nd neighbor pair, then the parameter should be the following form:

```
ddd_2_MoMo    -0.2
```

case2.) `IS_SK .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 *case1.*), 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 2nd 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 assign 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_{onsite}^i) for the local potential, since the local potential is applied on your Hamiltonian as: $e_{onsite}^i = e_{onsite}^i + e_{loc.pot}^i \times U_{onsite}^i$, i.e., it modifies onsite energy e_{onsite}^i to e_{onsite}^i . Here, U_{onsite}^i 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 1 # positive loc.pot
0 0.0 0.5 px py pz local.pot -1 -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*

case1.) **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 joiner, for example *p* orbital) and species information **_S**(as a suffix, for example **S**ulphur atom) to precisely indicating the atomic orbital where the SOC effect will be applied.

```
lambda_p_S 0.2
```

case2.) **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 custom type orbitals with *c*-prefix of the atom Bi and Bi.

Fixing parameter

If one want some parameters not to be fitted during the fitting procedures, one can fix those 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 MoS₂ (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 ?? 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 ?. 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/GA_METHOD/" 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 ?? 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 region), 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**, ??, ??, **PFILE**, ??, **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_POTENTIAL` tag in your `PFILE` to consider some local potential, you should turn on `LOCCHG` to `.TRUE.`, and add some relevant parameters (see `LOCAL_POTENTIAL` 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 rule¹². 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 try 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     ??
MITER      300
??         0.000000001
??         0.000000001

GFILE      ??
KFILE      ??
PFILE      ??
EFILE      ??
??  PARAM_FIT.new.dat

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

IS_SK      .TRUE.

```

¹²J. 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](#)

```

  e_s_C      0.80302000
  e_px_C     0.24526000
  e_py_C     0.24526000
  e_pz_C     0.24526000

```

sss_1_CC	-6.83068000
sps_1_CC	6.20074000
pps_1_CC	4.02474000
ppp_1_CC	-2.55074000
lambda_p_C	0.00000000

example of [PFILE](#) for Graphene

POSCAR-TB : [GFILE](#)

```
# Graphene with honeycomb lattice
1.0000000000000000
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

KPOINTS_BAND : [KFILE](#)

```
# Graphene with honeycomb lattice
1.0000000000000000
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 [KFILE](#) for Graphene

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/LMDIF_METHOD/" of your example folder.