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

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

September 17, 2018

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

System Requirements and installation

All the program has been written by modern Fortran90 language. LAPACK library should be properly linked in the makefile. 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 -DSPGLIB
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

LSTYPE *integer* Default: `LMDIF`
Method for parameter fitting. Currently, `TBFIT` only supports `LMDIF` method which is the Levenberg-Marquardt method[1], [2] using finite-difference for Jacobian.

PTOL & FTOL *real* Default: 0.00001
Tolerence of iteration of the fitting procedures. `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.

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.

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

[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).

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.

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
   1.00000000000000  # 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

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 absolute value of wave-function coefficient ($< \psi_{nk}|\phi|\psi_{nk} >$) will be printed out in `bandstructure_TBA.dat` 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 `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)
```

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.

SET *string*

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

CONSTRAINT TBPARAM

NN_CLASS

RIBBON

2. Details of the SET

Each SET tag should be ended up by END tag.

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

SITE_I *ineteger* : site index. `ORBT_I`th 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.

```
SET CONSTRAINT
    e_py_S   = e_px_S
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.

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 the scaling function for 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 current version does not support `MPI` parallelism for the Wannier charge center evaluation subroutine. Hence, do not use multicore for this particular calculation. Nevertheless, the compilation with `-DMPI` option is accepted.) 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 or Wilson loop calculation settings (The current version does not support `MPI` parallelism for the Wannier charge center evaluation subroutine. Hence, do not use multicore for this particular calculation. Nevertheless, the compilation with `-DMPI` option is accepted.)

```
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. (The current version does not support `MPI` parallelism for the Zak phase evaluation subroutine. Hence, do not use multicore for this particular calculation. However, the compilation with `-DMPI` option is accepted.)

```
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)
    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

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

.

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[3]:

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

[3]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)].

```
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 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 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 [`$M\ \ \theta\ \ \phi$`]`

```
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 0 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 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 $\mathtt{d}_{z2}$ orbital in `Mo` atom and $\mathtt{d}_{yz}$ orbital in `Mo`, and they are 2^{nd} neighbor pair, then the parameter should be the following form:

```
ddd_2_MoMo   -0.2
```

*case*2.) 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 *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 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  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*

*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.)