Libtetrabz Documentation

Release 2.0.0

kawamura

Jun 17, 2023

CONTENTS

CHAPTER ONE

INTRODUCTION

This document explains a tetrahedron method library `libtetrabz`. `libtetrabz` is a library to calculate the total energy, the charge density, partial density of states, response functions, etc. in a solid by using the optimized tetrahedron method *[1]*. Subroutines in this library receive the orbital (Kohn-Sham) energies as an input and calculate weights $w_{nn'k}$ for integration such as

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} F(\varepsilon_{nk}, \varepsilon_{n'k+q}) X_{nn'k} = \sum_{nn'} \sum_{k}^{N_k} w_{nn'k} X_{nn'k} \tag{1.1}$$

`libtetrabz` supports following Brillouin-zone integrations

$$\sum_{n} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) X_{nk} \tag{1.2}$$

$$\sum_{n} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \delta(\omega - \varepsilon_{nk}) X_{nk}(\omega) \tag{1.3}$$

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \delta(\varepsilon_{\rm F} - \varepsilon_{nk}) \delta(\varepsilon_{\rm F} - \varepsilon'_{n'k}) X_{nn'k} \tag{1.4}$$

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon_{nk} - \varepsilon'_{n'k}) X_{nn'k} \tag{1.5}$$

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \frac{\theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon'_{n'k} - \varepsilon_{\rm F})}{\varepsilon'_{n'k} - \varepsilon_{nk}} X_{nn'k} \tag{1.6}$$

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon'_{n'k} - \varepsilon_{\rm F}) \delta(\varepsilon'_{n'k} - \varepsilon_{nk} - \omega) X_{nn'k}(\omega) \tag{1.7}$$

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \frac{\theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon'_{n'k} - \varepsilon_{\rm F})}{\varepsilon'_{n'k} - \varepsilon_{nk} + i\omega} X_{nn'k}(\omega) \tag{1.8}$$

CHAPTER

TWO

INSTALLATION

2.1 Important files and directories

- **doc/**

  [Directory for manuals]

  - doc/index.html : Index page
- src/ : Directory for the sources of the library
- example/ : Directory for the sample program
- test/ : Directory for tests
- configure : Configuration script for the build

2.2 Prerequisite

libtetrabz requires the following

- fortran and C compiler
- MPI library (If you use MPI/Hybrid version)

2.3 Installation guide

1. Download .tar.gz file from following web page.

   https://github.com/mitsuaki1987/libtetrabz/releases/

2. Uncompress .tar.gz file and enter the generated directory.

   ```
   $ tar xzvf xzvf libtetrabz_1.0.1.tar.gz
   $ cd libtetrabz
   ```

3. Configure the build environment.

   ```
   $ ./configure --prefix=install_dir
   ```

   Then, this script checks the compiler and the libraries required for the installation, and creates Makefiles. install_dir indicates the full path of the directory where the library is installed (you should replace it according to your case). If none is specified, /use/local/ is chosen for storing libraries by make install

(Therefore, if one is not the admin, `install_dir` must be specified to the different directory). `configure` has many options, and they are used according to the environment etc. For more details, please see *Options for configure*.

4. After `configure` finishes successfully and Makefiles are generated, please type

```
$ make
```

to build libraries. Then please type

```
$ make install
```

to store libraries and the sample program to `install_dir/lib` and `install_dir/bin`, respectively. Although one can use libraries and the sample program without `make install`, they are a little different to the installed one.

5. Add the libtetrabz's library directory (`install_dir/lib`) to the search path of the dynamically linked program (environment variable `LD_LIBRARY_PATH`).

```
$ export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:install_dir/lib
```

6. Sample programs using `libtetrabz` are also compiled in `example/` .

`example/dos.x` : Compute DOS of a tight-binding model in the cubic lattice. The source code is `dos.f90`

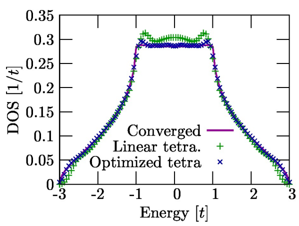

Fig. 1: Density of states of the tight-binding model on the cubic lattice calculated by using `dos.x`. The solid line indicates the result converged about the number of k. " $+$ " and " $\times$ " indicate results by using the linear tetrahedron method and the optimized tetrahedron method, respectively with $8 \times 8 \times 8k$ grid.

`example/lindhard.x` : Compute the Lindhard function. The source code is `lindhard.f90`

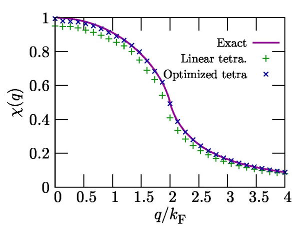

Fig. 2: (solid line) The analytical result of the Lindhard function. " $+$ " and " $\times$ " indicate results by using the linear tetrahedron method and the optimized tetrahedron method, respectively with $8 \times 8 \times 8k$ grid.

2.4 Options for configure

`configure` has many options and environment variables. They can be specified at once. E.g.

```
$ ./configure --prefix=/home/libtetrabz/ --with-mpi=yes FC=mpif90
```

All options and variables have default values. We show a part of them as follows:

`---prefix`

Default: `---prefix=/usr/local/`. Specify the directory where the library etc. are installed.

`--with-mpi`

Default: `--with-mpi=no` (without MPI). Whether use MPI (`--with-mpi=yes`), or not.

`--with-openmp`

Default: `--with-openmp=yes` (with OpenMP). Whether use OpenMP or not (`--with-openmp=no`).

`--enable-shared`

Default: `--enable-shared`. Whether generate shared library.

`--enable-static`

Default: `--enable-static`. Whether generate static library.

`FC`, `C`

Default: The fortran/C compiler chosen automatically from those in the system. When `--with-mpi` is specified, the corresponding MPI compiler (such as `mpif90`) is searched. If `FC` printed the end of the standard-output of `configure` is not what you want, please set it manually as `./configure FC=gfortran`.

`--help`

Display all options including above, and stop without configuration.

CHAPTER

THREE

LINKING LIBTETRABZ

e. g. / For intel fortran

```
$ ifort program.f90 -L install_dir/lib -I install_dir/include -ltetrabz -
↪fopenmp
$ mpiifort program.f90 -L install_dir/lib -I install_dir/include -ltetrabz -
↪fopenmp
```

e. g. / For intel C

```
$ icc -lifcore program.f90 -L install_dir/lib -I install_dir/include -ltetrabz␣
↪-fopenmp
$ mpiicc -lifcore program.f90 -L install_dir/lib -I install_dir/include -
↪ltetrabz_mpi -fopenmp
```

CHAPTER

FOUR

SUBROUTINES

You can call a subroutine in this library as follows:

```
USE libtetrabz, ONLY : libtetrabz_occ

CALL libtetrabz_occ(ltetra,bvec,nb,nge,eig,ngw,wght)
```

Every subroutine has a name starts from `libtetrabz_`.

For the C program, it can be used as follows:

```
#include "libtetrabz.h"

libtetrabz_occ(&ltetra,bvec,&nb,nge,eig,ngw,wght)
```

Variables should be passed as pointers. Arrays should be declared as one dimensional arrays. Also, the communicator argument for the routine should be transformed from the C/C++'s one to the fortran's one as follows.

```
comm_f = MPI_Comm_c2f(comm_c);
```

4.1 Total energy, charge density, occupations

$$\sum_n \int_{\mathrm{BZ}} \frac{d^3k}{V_{\mathrm{BZ}}} \theta(\varepsilon_{\mathrm{F}} - \varepsilon_{nk}) X_{nk} \tag{4.1}$$

```
CALL libtetrabz_occ(ltetra,bvec,nb,nge,eig,ngw,wght,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. $1 \cdots$ the linear tetrahedron method. $2 \cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$).

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.2 Fermi energy and occupations

$$\sum_n \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) X_{nk} \tag{4.2}$$

```
CALL libtetrabz_fermieng(ltetra,bvec,nb,nge,eig,ngw,wght,ef,nelec,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. 1 $\cdots$ the linear tetrahedron method. 2 $\cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\mathrm{F}} = 0$).

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
REAL(8),INTENT(OUT) :: ef
```

The Fermi energy.

```
REAL(8),INTENT(IN) :: nelec
```

The number of (valence) electrons per spin.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.3 Partial density of states

$$\sum_n \int_{\mathrm{BZ}} \frac{d^3 k}{V_{\mathrm{BZ}}} \delta(\omega - \varepsilon_{nk}) X_{nk}(\omega) \tag{4.3}$$

```
CALL libtetrabz_dos(ltetra,bvec,nb,nge,eig,ngw,wght,ne,e0,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. 1 $\cdots$ the linear tetrahedron method. 2 $\cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$).

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(ne,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN) :: ne
```

The number of energy where DOS is calculated.

```
REAL(8),INTENT(IN) :: e0(ne)
```

Energies where DOS is calculated.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.4 Integrated density of states

$$\sum_{n} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\omega - \varepsilon_{nk}) X_{nk}(\omega) \tag{4.4}$$

```
CALL libtetrabz_intdos(ltetra,bvec,nb,nge,eig,ngw,wght,ne,e0,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. 1 $\cdots$ the linear tetrahedron method. 2 $\cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$).

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(ne,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN) :: ne
```

The number of energy where DOS is calculated.

```
REAL(8),INTENT(IN) :: e0(ne)
```

Energies where DOS is calculated.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.5 Nesting function and Fr&oumlhlich parameter

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \delta(\varepsilon_{\rm F} - \varepsilon_{nk}) \delta(\varepsilon_{\rm F} - \varepsilon'_{n'k}) X_{nn'k} \tag{4.5}$$

```
CALL libtetrabz_dbldelta(ltetra,bvec,nb,nge,eig1,eig2,ngw,wght,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. 1 $\cdots$ the linear tetrahedron method. 2 $\cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig1(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$). Do the same with `eig2`.

```
REAL(8),INTENT(IN) :: eig2(nb,nge(1),nge(2),nge(3))
```

Another orbital energy. E.g. ε_{k+q} on a shifted grid.

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(nb,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.6 A part of DFPT calculation

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon_{nk} - \varepsilon'_{n'k}) X_{nn'k} \tag{4.6}$$

```
CALL libtetrabz_dblstep(ltetra,bvec,nb,nge,eig1,eig2,ngw,wght,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. 1 $\cdots$ the linear tetrahedron method. 2 $\cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig1(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$). Do the same with `eig2`.

```
REAL(8),INTENT(IN) :: eig2(nb,nge(1),nge(2),nge(3))
```

Another orbital energy. E.g. ε_{k+q} on a shifted grid.

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(nb,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.7 Static polarization function

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \frac{\theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon'_{n'k} - \varepsilon_{\rm F})}{\varepsilon'_{n'k} - \varepsilon_{nk}} X_{nn'k} \tag{4.7}$$

```
CALL libtetrabz_polstat(ltetra,bvec,nb,nge,eig1,eig2,ngw,wght,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. $1 \cdots$ the linear tetrahedron method. $2 \cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig1(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$). Do the same with `eig2`.

```
REAL(8),INTENT(IN) :: eig2(nb,nge(1),nge(2),nge(3))
```

Another orbital energy. E.g. ε_{k+q} on a shifted grid.

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(nb,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.8 Phonon linewidth

$$\sum_{nn'} \int_{\rm BZ} \frac{d^3 k}{V_{\rm BZ}} \theta(\varepsilon_{\rm F} - \varepsilon_{nk}) \theta(\varepsilon'_{n'k} - \varepsilon_{\rm F}) \delta(\varepsilon'_{n'k} - \varepsilon_{nk} - \omega) X_{nn'k}(\omega) \tag{4.8}$$

```
CALL libtetrabz_fermigr(ltetra,bvec,nb,nge,eig1,eig2,ngw,wght,ne,e0,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. $1 \cdots$ the linear tetrahedron method. $2 \cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig1(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\rm F} = 0$). Do the same with `eig2`.

```
REAL(8),INTENT(IN) :: eig2(nb,nge(1),nge(2),nge(3))
```

Another orbital energy. E.g. ε_{k+q} on a shifted grid.

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
REAL(8),INTENT(OUT) :: wght(ne,nb,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN) :: ne
```

The number of branches of the phonon.

```
REAL(8),INTENT(IN) :: e0(ne)
```

Phonon frequencies.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

4.9 Polarization function (complex frequency)

$$\sum_{nn'} \int_{\mathrm{BZ}} \frac{d^3 k}{V_{\mathrm{BZ}}} \frac{\theta(\varepsilon_{\mathrm{F}} - \varepsilon_{nk})\theta(\varepsilon'_{n'k} - \varepsilon_{\mathrm{F}})}{\varepsilon'_{n'k} - \varepsilon_{nk} + i\omega} X_{nn'k}(\omega) \tag{4.9}$$

```
CALL libtetrabz_polcmplx(ltetra,bvec,nb,nge,eig1,eig2,ngw,wght,ne,e0,comm)
```

Parameters

```
INTEGER,INTENT(IN) :: ltetra
```

Specify the type of the tetrahedron method. $1 \cdots$ the linear tetrahedron method. $2 \cdots$ the optimized tetrahedron method *[1]*.

```
REAL(8),INTENT(IN) :: bvec(3,3)
```

Reciprocal lattice vectors (arbitrary unit). Because they are used to choose the direction of tetrahedra, only their ratio is used.

```
INTEGER,INTENT(IN) :: nb
```

The number of bands.

```
INTEGER,INTENT(IN) :: nge(3)
```

Specify the k-grid for input orbital energy.

```
REAL(8),INTENT(IN) :: eig1(nb,nge(1),nge(2),nge(3))
```

The orbital energy measured from the Fermi energy ($\varepsilon_{\mathrm{F}} = 0$). Do the same with `eig2`.

```
REAL(8),INTENT(IN) :: eig2(nb,nge(1),nge(2),nge(3))
```

Another orbital energy. E.g. ε_{k+q} on a shifted grid.

```
INTEGER,INTENT(IN) :: ngw(3)
```

Specify the k-grid for output integration weights. You can make `ngw` $\neq$ `nge` (See *Appendix*).

```
COMPLEX(8),INTENT(OUT) :: wght(ne,nb,nb,ngw(1),ngw(2),ngw(3))
```

The integration weights.

```
INTEGER,INTENT(IN) :: ne
```

The number of imaginary frequencies where polarization functions are calculated.

```
COMPLEX(8),INTENT(IN) :: e0(ne)
```

Complex frequencies where polarization functions are calculated.

```
INTEGER,INTENT(IN),OPTIONAL :: comm
```

Optional argument. Communicators for MPI such as `MPI_COMM_WORLD`. Only for MPI / Hybrid parallelization. For C compiler without MPI, just pass `NULL` to omit this argment.

CHAPTER

FIVE

PIECE OF SAMPLE CODE

This sample shows the calculation of the charge density.

$$\rho(r) = 2 \sum_{nk} \theta(\varepsilon_{\mathrm{F}} - \varepsilon_{nk}) |\varphi_{nk}(r)|^2 \tag{5.1}$$

```fortran
SUBROUTINE calc_rho(nr,nb,ng,nelec,bvec,eig,ef,phi,rho)
  !
  USE libtetrabz, ONLY : libtetrabz_fermieng
  IMPLICIT NONE
  !
  INTEGER,INTENT(IN) :: nr ! number of r
  INTEGER,INTENT(IN) :: nb ! number of bands
  INTEGER,INTENT(IN) :: ng(3)
  ! k-point mesh
  REAL(8),INTENT(IN) :: nelec ! number of electrons per spin
  REAL(8),INTENT(IN) :: bvec(3,3) ! reciplocal lattice vector
  REAL(8),INTENT(IN) :: eig(nb,ng(1),ng(2),ng(3)) ! Kohn-Sham eigenvalues
  REAL(8),INTENT(OUT) :: ef ! Fermi energy
  COMPLEX(8),INTENT(IN) :: phi(nr,nb,ng(1),ng(2),ng(3)) ! Kohn-Sham orbitals
  REAL(8),INTENT(OUT) :: rho(nr) ! Charge density
  !
  INTEGER :: ib, i1, i2, i3, ltetra
  REAL(8) :: wght(nb,ng(1),ng(2),ng(3))
  !
  ltetra = 2
  !
  CALL libtetrabz_fermieng(ltetra,bvec,nb,ng,eig,ng,wght,ef,nelec)
  !
  rho(1:nr) = 0d0
  DO i1 = 1, ng(3)
     DO i2 = 1, ng(2)
        DO i1 = 1, ng(1)
           DO ib = 1, nb
              rho(1:nr) = rho(1:nr) + 2d0 * wght(ib,i1,i2,i3) &
              &     * DBLE(CONJG(phi(1:nr,ib,i1,i2,i3)) * phi(1:nr,ib,i1,i2,i3))
           END DO
        END DO
     END DO
  END DO
  !
END SUBROUTINE calc_rho
```

CHAPTER

SIX

RE-DISTRIBUTION OF THIS PROGRAM

6.1 Contain libtetrabz in your program

libtetrabz is distributed with the *MIT License*. To summarize this, you can freely modify, copy and paste libtetrabz to any program such as a private program (in the research group, co-workers, etc.), open-source, free, and commercial software. Also, you can freely choose the license to distribute your program.

6.2 Build libtetrabz without Autoconf

In this package, libtetrabz is built with Autotools (Autoconf, Automake, Libtool). If you do not want to use Autotools for your distributed program with libtetrabz's source, you can use the following simple Makefile (please care about TAB).

```
F90 = gfortran
FFLAGS = -fopenmp -O2 -g

OBJS = \
libtetrabz.o \
libtetrabz_dbldelta_mod.o \
libtetrabz_dblstep_mod.o \
libtetrabz_dos_mod.o \
libtetrabz_fermigr_mod.o \
libtetrabz_occ_mod.o \
libtetrabz_polcmplx_mod.o \
libtetrabz_polstat_mod.o \
libtetrabz_common.o \

.SUFFIXES :
.SUFFIXES : .o .F90

libtetrabz.a:$(OBJS)
	ar cr $@ $(OBJS)

.F90.o:
	$(F90) $(FFLAGS) -c $<

clean:
	rm -f *.a *.o *.mod
```

(continues on next page)

(continued from previous page)

```
libtetrabz.o:libtetrabz_polcmplx_mod.o
libtetrabz.o:libtetrabz_fermigr_mod.o
libtetrabz.o:libtetrabz_polstat_mod.o
libtetrabz.o:libtetrabz_dbldelta_mod.o
libtetrabz.o:libtetrabz_dblstep_mod.o
libtetrabz.o:libtetrabz_dos_mod.o
libtetrabz.o:libtetrabz_occ_mod.o
libtetrabz_dbldelta_mod.o:libtetrabz_common.o
libtetrabz_dblstep_mod.o:libtetrabz_common.o
libtetrabz_dos_mod.o:libtetrabz_common.o
libtetrabz_fermigr_mod.o:libtetrabz_common.o
libtetrabz_occ_mod.o:libtetrabz_common.o
libtetrabz_polcmplx_mod.o:libtetrabz_common.o
libtetrabz_polstat_mod.o:libtetrabz_common.o
```

6.3 MIT License

CHAPTER SEVEN

CONTACTS

Please post bug reports and questions to the forum

> http://sourceforge.jp/projects/fermisurfer/forums/

When you want to join us, please contact me as follows.

The Institute of Solid State Physics

Mitsuaki Kawamura

`mkawamura__at__issp.u-tokyo.ac.jp`

CHAPTER

EIGHT

APPENDIX

8.1 Inverse interpolation

We consider an integration as follows:

$$\langle X \rangle = \sum_k X_k w(\varepsilon_k) \tag{8.1}$$

If this integration has conditions that

- $w(\varepsilon_k)$ is sensitive to ε_k (e. g. the stepfunction, the delta function, etc.) and requires ε_k on a dense k grid, and
- the numerical cost to obtain X_k is much larger than the cost for ε_k (e. g. the polarization function),

it is efficient to interpolate X_k into a denser k grid and evaluate that integration in a dense k grid. This method is performed as follows:

1. Calculate ε_k on a dense k grid.
2. Calculate X_k on a coarse k grid and obtain that on a dense k grid by using the linear interpolation, the polynomial interpolation, the spline interpolation, etc.

$$X_k^{\rm dense} = \sum_{k'}^{\rm coarse} F_{kk'} X_{k'}^{\rm coarse} \tag{8.2}$$

1. Evaluate that integration in the dense k grid.

$$\langle X \rangle = \sum_{k}^{\rm dense} X_k^{\rm dense} w_k^{\rm dense} \tag{8.3}$$

The inverse interpolation method (Appendix of *[2]*) arrows as to obtain the same result to above without interpolating X_k into a dense k grid. In this method, we map the integration weight on a dense k grid into that on a coarse k grid (inverse interpolation). Therefore, if we require

$$\sum_{k}^{\rm dense} X_k^{\rm dense} w_k^{\rm dense} = \sum_{k}^{\rm coarse} X_k^{\rm coarse} w_k^{\rm coarse} \tag{8.4}$$

we obtain

$$w_k^{\rm coarse} = \sum_{k}^{\rm dense} F_{k'k} w_{k'}^{\rm dense} \tag{8.5}$$

The numerical procedure for this method is as follows:

1. Calculate the integration weight on a dense k grid $w_k^{\rm dense}$ from ε_k on a dense k grid.
2. Obtain the integration weight on a coarse k grid $w_k^{\rm coarse}$ by using the inverse interpolation method.

3. Evaluate that integration in a coarse k grid where X_k was calculated.

All routines in `libtetrabz` can perform the inverse interpolation method; if we make k grids for the orbital energy (`nge`) and the integration weight (`ngw`) different, we obtain $w_k^{\rm coarse}$ calculated by using the inverse interpolation method.

8.2 Double delta integration

For the integration

$$\sum_{nn'k} \delta(\varepsilon_{\rm F} - \varepsilon_{nk}) \delta(\varepsilon_{\rm F} - \varepsilon'_{n'k}) X_{nn'k} \tag{8.6}$$

first, we cut out one or two triangles where $\varepsilon_{nk} = \varepsilon_{\rm F}$ from a tetrahedron and evaluate $\varepsilon_{n'k+q}$ at the corners of each triangles as

$$\varepsilon'^{k+q}_{i} = \sum_{j=1}^{4} F_{ij}(\varepsilon_1^k, \cdots, \varepsilon_4^k, \varepsilon_{\rm F}) \epsilon_j^{k+q}. \tag{8.7}$$

Then we calculate $\delta(\varepsilon_{n'k+q} - \varepsilon {\rm F})$ in each triangles and obtain weights of corners. This weights of corners are mapped into those of corners of the original tetrahedron as

$$W_i = \sum_{j=1}^{3} \frac{S}{\nabla_k \varepsilon_k} F_{ji}(\varepsilon_1^k, \cdots, \varepsilon_4^k, \varepsilon_{\rm F}) W'_j. \tag{8.8}$$

F_{ij} and $\frac{S}{\nabla_k \varepsilon_k}$ are calculated as follows ($a_{ij} \equiv (\varepsilon_i - \varepsilon_j)/(\varepsilon_{\rm F} - \varepsilon_j)$):

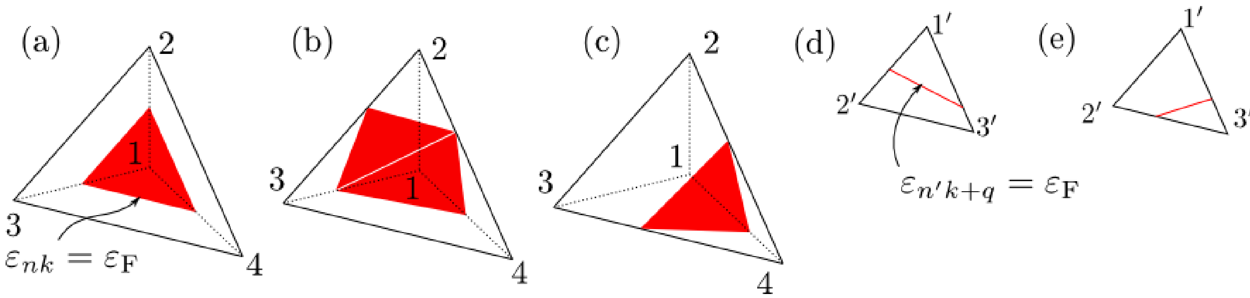

Fig. 1: How to divide a tetrahedron in the case of $\epsilon_1 \leq \varepsilon_{\rm F} \leq \varepsilon_2$ (a), $\varepsilon_2 \leq \varepsilon_{\rm F} \leq \varepsilon_3$ (b), and $\varepsilon_3 \leq \varepsilon_{\rm F} \leq \varepsilon_4$ (c).

- When $\varepsilon_1 \leq \varepsilon_{\rm F} \leq \varepsilon_2 \leq \varepsilon_3 \leq \varepsilon_4$ [Fig. 1 (a)],

$$F = \begin{pmatrix} a_{12} & a_{21} & 0 & 0 \\ a_{13} & 0 & a_{31} & 0 \\ a_{14} & 0 & 0 & a_{41} \end{pmatrix}, \qquad \frac{S}{\nabla_k \varepsilon_k} = \frac{3 a_{21} a_{31} a_{41}}{\varepsilon_{\rm F} - \varepsilon_1} \tag{8.9}$$

- When $\varepsilon_1 \leq \varepsilon_2 \leq \varepsilon_{\rm F} \leq \varepsilon_3 \leq \varepsilon_4$ [Fig. 1 (b)],

$$F = \begin{pmatrix} a_{13} & 0 & a_{31} & 0 \\ a_{14} & 0 & 0 & a_{41} \\ 0 & a_{24} & 0 & a_{42} \end{pmatrix}, \qquad \frac{S}{\nabla_k \varepsilon_k} = \frac{3 a_{31} a_{41} a_{24}}{\varepsilon_{\rm F} - \varepsilon_1} \tag{8.10}$$

$$F = \begin{pmatrix} a_{13} & 0 & a_{31} & 0 \\ 0 & a_{23} & a_{32} & 0 \\ 0 & a_{24} & 0 & a_{42} \end{pmatrix}, \qquad \frac{S}{\nabla_k \varepsilon_k} = \frac{3 a_{23} a_{31} a_{42}}{\varepsilon_{\rm F} - \varepsilon_1} \tag{8.11}$$

- When $\varepsilon_1 \leq \varepsilon_2 \leq \varepsilon_3 \leq \varepsilon_{\rm F} \leq \varepsilon_4$ [Fig. 1 (c)],

$$F = \begin{pmatrix} a_{14} & 0 & 0 & a_{41} \\ a_{13} & a_{24} & 0 & a_{42} \\ a_{12} & 0 & a_{34} & a_{43} \end{pmatrix}, \qquad \frac{S}{\nabla_k \varepsilon_k} = \frac{3 a_{14} a_{24} a_{34}}{\varepsilon_1 - \varepsilon_{\rm F}} \tag{8.12}$$

Weights on each corners of the triangle are computed as follows [$(a'_{ij} \equiv (\varepsilon'_i - \varepsilon'_j)/(\varepsilon_{\rm F} - \varepsilon'_j))$]:

- When $\varepsilon'_1 \leq \varepsilon_{\rm F} \leq \varepsilon'_2 \leq \varepsilon'_3$ [Fig. 1 (d)],

$$W'_1 = L(a'_{12} + a'_{13}), \qquad W'_2 = L a'_{21}, \qquad W'_3 = L a'_{31}, \qquad L \equiv \frac{a'_{21} a'_{31}}{\varepsilon_{\rm F} - \varepsilon'_1} \tag{8.13}$$

- When $\varepsilon'_1 \leq \varepsilon'_2 \leq \varepsilon_{\rm F} \leq \varepsilon'_3$ [Fig. 1 (e)],

$$W'_1 = L a'_{13}, \qquad W'_2 = L a'_{23}, \qquad W'_3 = L(a'_{31} + a'_{32}), \qquad L \equiv \frac{a'_{13} a'_{23}}{\varepsilon'_3 - \varepsilon_{\rm F}} \tag{8.14}$$

CHAPTER

NINE

REFERENCE

[1] M. Kawamura, Y. Gohda, and S. Tsuneyuki, Phys. Rev. B 89, 094515 (2014).

[2] M. Kawamura, R. Akashi, and S. Tsuneyuki, Phys. Rev. B 95, 054506 (2017).