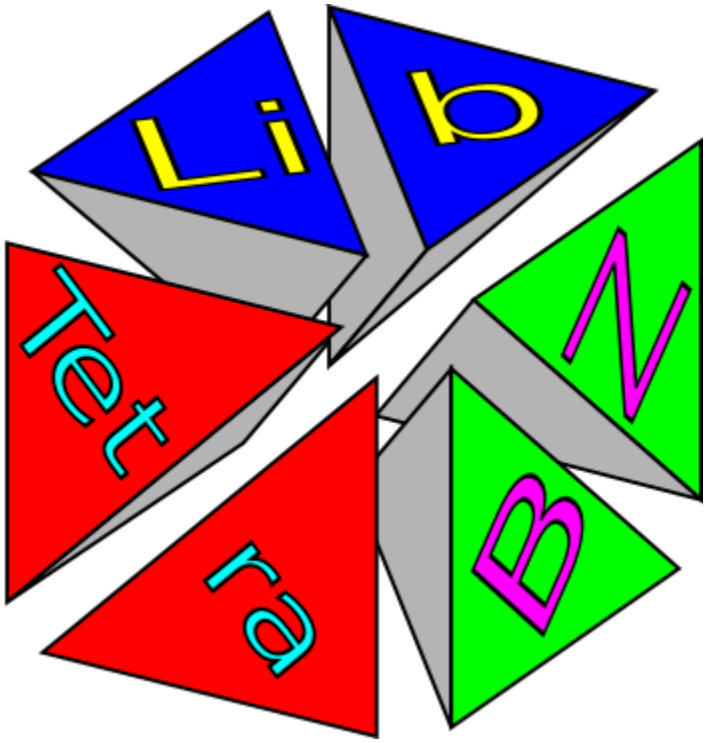

Libtetrabz Documentation

リリース *2.0.0*

kawamura

2023 年 05 月 29 日

目 次

第1章　はじめに

この文書ではテトラヘドロン法ライブラリ libtetrabz についての解説を行っている. libtetrabz は線形テトラヘドロン法もしくは最適化線形テトラヘドロン法 *[1]* を用いて全エネルギーや電荷密度, 部分状態密度, 分極関数等を計算するためのライブラリ群である. このライブラリには, 軌道エネルギーをインプットとして,

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

のような積分における, 積分重み $w_{nn'k}$ を出力するサブルーチンを, 各種計算について取り揃えている. 具体的には以下の計算に対応している.

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

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

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

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

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

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

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

第2章　インストール方法

2.1 主なファイルとディレクトリ

- **doc/**
  [マニュアルのディレクトリ]
  - doc/index.html : 目録ページ
- src/ : ライブラリのソースコードのディレクトリ
- example/ : ライブラリ使用例のディレクトリ
- test/ : ライブラリのビルドのテスト用ディレクトリ
- configure : ビルド環境設定用スクリプト

2.2 要件

以下のものが必要となる.

- fortran および C コンパイラ
- MPI ライブラリ (MPI/ハイブリッド並列版を利用する場合)

2.3 インストール手順

1. 以下の場所から .tar.gz ファイルをダウンロードする.

   http://osdn.jp/projects/libtetrabz/releases/

2. ダウンロードした .tar.gz ファイルを展開し, 出来たディレクトリに入る.

```
$ tar xzvf libtetrabz-version.tar.gz
$ cd libtetrabz-version
```

3. ビルド環境を設定する.

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

これにより, ビルドに必要なコンパイラやライブラリ等の環境のチェックが行われ, Makefile 等が作成される. ただし install_dir はインストール先のディレクトリの絶対パスとする (以後各自のディレクトリ名で読み替えること). なにも指定しないと /use/local/ が設定され, 後述の make install で/usr/local/lib 内にライブラリが置かれる (したがって, 管理者権限がない場合には install_dir を別の場所に指定しなければならない). configure にはこの他にも様々なオプションがあり, 必要に応じて用途や環境に合わせてそれらを使用する. 詳しくは *configure* **のオプション** を参照.

4. configure の実行が正常に行われ, Makefile が生成された後は

```
$ make
```

とタイプしてライブラリ等のビルドを行う. これが成功したのちに

```
$ make install
```

とすると, ライブラリが install_dir/lib に置かれる. make install をしなくても, ビルドをしたディレクトリ内にあるライブラリやミニアプリを使うことは可能であるが, 使い勝手がやや異なる.

5. 共有リンクを行ったプログラムの実行時にライブラリを探しにいけるよう, 環境変数 LD_LIBRARY_PATH に libtetrabz をインストールしたディレクトリを追加する.

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

6. また, example/ 以下のライブラリ使用例のプログラムもコンパイルされる.

   example/dos.x : 立法格子シングルバンドタイトバインディングモデルの DOS を 計算する. ソースコードは dos.f90

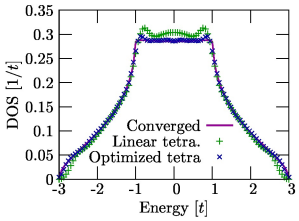

図 1: dos.x を使って計算した, 立法格子タイトバインディング模型の状態密度. 実線は十分多くの k 点を利用した時の結果. " + ", " × "はそれぞれ $8 \times 8 \times 8k$ グリッドでの線形テトラヘドロン法および最適化テトラヘドロン法の結果.

   example/lindhard.x : リントハルト関数を計算する. ソースコードは lindhard.f90

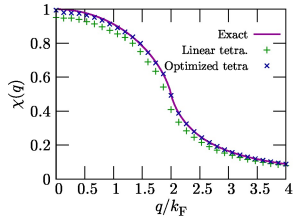

図 2: lindhard.x を使って計算した Lindhard 関数. 実線は解析的な結果. " + ", " × "はそれぞれ $8 \times 8 \times 8k$ グリッドでの線形テトラヘドロン法および最適化テトラヘドロン法の結果.

2.4 configure のオプション

configure には多数のオプションと変数があり, それらを組み合わせて指定する. 指定しない場合にはデフォルト値が使われる.

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

おもなものを次に挙げる.

`---prefix`

デフォルト: `---prefix=/usr/local/`. ライブラリ等のインストールを行うディレクトリツリーを指定する.

`--with-mpi`

デフォルト: `--with-mpi=no` (MPI を用いない). MPI を用いるか (`--with-mpi=yes`), 否かを指定する.

`--with-openmp`

デフォルト: `--with-openmp=yes` (OpenMP を用いる). OpenMP を用いるか否か (`--with-openmp=no`) を指定する.

`--enable-shared`

デフォルト: `--enable-shared`. 共有ライブラリを作成するか否か

`--enable-static`

デフォルト: `--enable-static`. 静的ライブラリを作成するか否か.

FC, CC

デフォルト: システムにインストールされている fortran/C コンパイラをスキャンして, 自動的に設定する. `--with-mpi` を指定した時にはそれに応じたコマンド (`mpif90` 等) を自動で探し出して設定する. `configure` の最後に出力される `FC` が望んだものでは無かった場合には `./configure FC=gfortran` のように手で指定する.

`--help`

このオプションを指定した時には, ビルドの環境設定は行われず, 上記を含めたすべてのオプションを表示する.

第3章　ライブラリのリンク方法

例/ intel fortran の場合

```
$ 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
```

例/ intel C の場合

```
$ 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
```

第4章　各サブルーチンの説明

以下のサブルーチンを任意のプログラム内で

```
USE libtetrabz, ONLY : libtetrabz_occ

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

のように呼び出して使用できる. サブルーチン名はすべて libtetrabz_ からはじまる.

C 言語で書かれたプログラムから呼び出す場合には次のようにする.

```
#include "libtetrabz.h"

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

変数はすべてポインタとして渡す. 配列はすべて 1 次元配列として定義し一番左の添字が内側のループとなるようにする. また MPI/ハイブリッド並列のときにライブラリに渡すコミュニケーター変数を, 次のように C/C++のものから fortran のものに変換する。

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

4.1 全エネルギー, 電荷密度等 (占有率の計算)

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$).

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

`ngw(3)` : (入力, 整数配列) 積分重みの k メッシュ. `nge` と違っていても構わない (**補遺** 参照).

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

`wght(nb,ngw(1),ngw(2),ngw(3))` : (出力, 実数配列) 積分重み

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

オプショナル引数. MPI のコミュニケーター (`MPI_COMM_WORLD` など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには `NULL` を入れる.

4.2 Fermi エネルギー (占有率も同時に計算する)

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー.

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

軌道エネルギーのメッシュ数.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

Fermi エネルギー

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

スピンあたりの (荷) 電子数

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

オプショナル引数. MPI のコミュニケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.3 (部分) 状態密度

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーの k メッシュ数.

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

軌道エネルギー.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

状態密度を計算するエネルギー点数

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

状態密度を計算するエネルギー

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.4 積分状態密度

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーの k メッシュ数.

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

軌道エネルギー.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

状態密度を計算するエネルギー点数

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

状態密度を計算するエネルギー

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.5 ネスティング関数, Fröhlich パラメーター

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーの k メッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$). eig2 も同様.

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

軌道エネルギー. 移行運動量の分だけグリッドをずらしたものなど.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.6 DFPT 計算の一部

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$). eig2 も同様.

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

軌道エネルギー. 移行運動量の分だけグリッドをずらしたものなど.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.7 独立分極関数 (静的)

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$). eig2 も同様.

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

軌道エネルギー. 移行運動量の分だけグリッドをずらしたものなど.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

4.8 フォノン線幅等

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$). `eig2` も同様.

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

軌道エネルギー. 移行運動量の分だけグリッドをずらしたものなど.

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

積分重みの k メッシュ. `nge` と違っていても構わない (**補遺** 参照).

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

積分重み

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

フォノンモード数

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

フォノン振動数

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

オプショナル引数. MPI のコミニュケーター (`MPI_COMM_WORLD` など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには `NULL` を入れる.

4.9 分極関数 (複素振動数)

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

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

パラメーター

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

テトラヘドロン法の種類を決める. 1 · · · 線形テトラヘドロン法, 2 · · · 最適化線形テトラヘドロン法 *[1]*

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

逆格子ベクトル. 単位は任意で良い. 逆格子の形によって四面体の切り方を決めるため, それらの長さの比のみが必要であるため.

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

バンド本数

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

軌道エネルギーのメッシュ数.

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

軌道エネルギー. Fermi エネルギーを基準とすること ($\varepsilon_{\mathrm{F}} = 0$). eig2 も同様.

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

軌道エネルギー. 移行運動量の分だけグリッドをずらしたものなど.

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

積分重みの k メッシュ. nge と違っていても構わない (**補遺** 参照).

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

積分重み .

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

計算を行う虚振動数の点数

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

計算を行う複素振動数

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

オプショナル引数. MPI のコミニュケーター (MPI_COMM_WORLD など) を入れる. libtetrabz を内部で MPI/Hybrid 並列するときのみ入力する. C 言語では使用しないときには NULL を入れる.

第5章　サンプルコード(抜粋)

以下では電荷密度

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

を計算するサブルーチンを示す.

```
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
```

(次のページに続く)

(前のページからの続き)

```
  END DO
  !
END SUBROUTINE calc_rho
```

第6章　プログラムの再配布

6.1 自分のプログラムに libtetrabz を含める

libtetrabz は下記の *MIT* **ライセンス** に基づいて配布されている. これはかいつまんで言うと, 個人的 (研究室や共同研究者等のグループ) なプログラムであろうとも, 公開したり売ったりするプログラムであろうとも自由にコピペしたり改変して良いし, どのようなライセンスで配布しても構わない, と言うことである.

6.2 Autoconf を使わずに libtetrabz をビルドする

このパッケージでは Autotools (Autoconf, Aitomake, Libtool) を使って libtetrabz をビルドしている. もし再配布するソースコードに libtetrabz を含めるときに, Autoconf の使用に支障がある場合には, 以下の簡易版の Makefile を使うと良い (タブに注意).

```
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:
```

(次のページに続く)

(前のページからの続き)

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

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 ライセンス

第7章　問い合わせ先

プログラムのバグや質問は以下のフォーラムへご投稿ください.

http://sourceforge.jp/projects/libtetrabz/forums/

開発に参加したい方は以下の連絡先にて受け付けております.

東京大学物性研究所

河村光晶

mkawamura__at__issp.u-tokyo.ac.jp

第8章　補遺

8.1 逆補間

積分

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

を計算するとする. このとき,

- w は ε_k に敏感な関数 (階段関数 · デルタ関数等) であり, なるべく細かいグリッド上の ε_k が必要である.
- X_k を求めるための計算コストが ε_k の計算コストよりかなり大きい.

という場合には X_k のグリッドを補間により増やす方法が有効である. それは,

1. ε_k を細かい k グリッド上で計算する.
2. X_k を粗いグリッド上で計算し, それを補間 (線形補間, 多項式補間, スプライン補間など) して細かいグリッド上での値を得る.

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

1. 細かい k グリッドで上記の積分を行う.

$$\langle X \rangle = \sum_k^{\mathrm{dense}} X_k^{\mathrm{dense}} w_k^{\mathrm{dense}} \tag{8.3}$$

という流れで行われる.

さらに, この計算と同じ結果を得るように粗いグリッド上での積分重み w_k^{coarse} を $w_k\mathrm{dense}$ から求める**逆補間** (*[2]* の Appendix) も可能である. すなわち,

$$\sum_k^{\mathrm{dense}} X_k^{\mathrm{dense}} w_k^{\mathrm{dense}} = \sum_k^{\mathrm{coarse}} X_k^{\mathrm{coarse}} w_k^{\mathrm{coarse}} \tag{8.4}$$

が満たされる事を要請すると

$$w_k^{\mathrm{coarse}} = \sum_k^{\mathrm{dense}} F_{k'k} w_{k'}^{\mathrm{dense}} \tag{8.5}$$

となる. この場合の計算手順は,

1. 細かい k グリッド上の ε_k から w_k^{dense} を計算する.
2. 逆補間により w_k^{coarse} を求める.

3. 粗いグリッド上での X_k との積和を行う.

となる. このライブラリ内の全ルーチンはこの逆補間の機能を備えており, 軌道エネルギーの k グリッドと重み関数の k グリッドを異なる値にすると逆補間された w_k^{coarse} が出力される.

8.2 2重デルタ関数の積分

For the integration

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

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

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

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

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

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

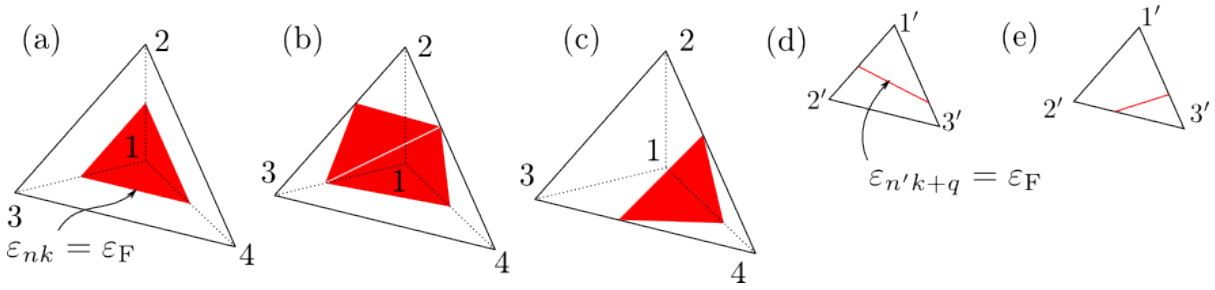

図 1: How to divide a tetrahedron in the case of $\epsilon_1 \le \varepsilon_{\mathrm{F}} \le \varepsilon_2$ (a), $\varepsilon_2 \le \varepsilon_{\mathrm{F}} \le \varepsilon_3$ (b), and $\varepsilon_3 \le \varepsilon_{\mathrm{F}} \le \varepsilon_4$ (c).

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

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

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

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

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

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

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

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)],

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

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

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

第9章　参考文献

[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).