平面波電子状態計算プログラム Quantum ESPRESSO ソースコード解説

Yasuhiro Oishi

2024 年 12 月 25 日

目 次

第 I 部
はじめに

密度汎関数理論（Density functional theory; DFT）で結晶や分子の電子状態を求めることは、系のエネルギーを電子密度の汎関数として与え、変分原理によりエネルギーの停留条件として得られる方程式を解くことに帰する。これは、Kohn-Sham 方程式と呼ばれる、位置に関する二階の偏微分方程式であり、自己無撞着（Self Consistent Field; SCF）計算と呼ばれる方法によって解かれる。

本稿では、DFT 計算のフリーソフトウェアである Quantum Espresso（QE）を対象とし、SCF 計算のコードを解読することを目指す。

なお、QE の最新版（2024 年 1 月時点）は version 7.3 だが、本コード読みでは version 5.1 を扱う。パッケージは以下のリンクからダウンロードできる：

```
https://gitlab.com/QEF/q-e/-/releases/qe-5.1.0
```

SCF 計算の主なコードは q-e-qe-5.1.0/PW/src/ に格納されている。
また、コード読みの際は以下の文献を参考にした：

- Giannozzi, Paolo, et al. ”QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials.” Journal of physics: Condensed matter 21.39 (2009): 395502.
- R.M. マーチン 著, 物質の電子状態 (下)
- 香山正憲　著, 平面波基底の第一原理計算法（2024）

第 II 部
平面波電子状態計算の理論

1　ブロッホの定理

結晶は、構成する格子の周期で同じ構造を繰り返しもった系である。ブロッホの定理は、同じ構造を繰り返しもった系において電子の波動関数がとるべき形を規定するものである。本節ではこの定理を群論に基づき示そう。

結晶内の格子点の座標を

$$\boldsymbol{R} = n_1\boldsymbol{a}_1 + n_2\boldsymbol{a}_2 + n_3\boldsymbol{a}_3 \tag{1}$$

と表す。ただし n_i は任意の整数である。ある関数に作用させると座標を $\boldsymbol{R}$ だけずらず操作（並進操作）

$$\hat{T}_{\boldsymbol{R}} f(\boldsymbol{r}) = f(\boldsymbol{r} + \boldsymbol{R}) \tag{2}$$

を定義しよう。すると、

$$\hat{T}_{\boldsymbol{R}}\hat{T}_{\boldsymbol{R'}} f(\boldsymbol{r}) = f(\boldsymbol{r} + \boldsymbol{R} + \boldsymbol{R'}) = \hat{T}_{\boldsymbol{R'}}\hat{T}_{\boldsymbol{R}} f(\boldsymbol{r}) \tag{3}$$

であるので、並進操作 (2) の集合はアーベル群をなす。アーベル群の既約表現は全て一次元である。[1] ゆえに、$\hat{T}_{\boldsymbol{R}}$ を $\psi(\boldsymbol{r})$ に作用させた結果は、任意の複素数 $C_{\boldsymbol{R}}$ を用いて

$$\hat{T}_{\boldsymbol{R}}\psi(\boldsymbol{r}) = C_{\boldsymbol{R}}\psi(\boldsymbol{r}) \tag{13}$$

と書ける。また、全ての $\hat{T}_{\boldsymbol{R}}$ は $\hat{T}_{\boldsymbol{a}_i}$ を基本単位 (生成元) として作ることができる。つまり、$\hat{T}_{\boldsymbol{R}} = (\hat{T}_{\boldsymbol{a}_1})^{n_1}(\hat{T}_{\boldsymbol{a}_2})^{n_2}(\hat{T}_{\boldsymbol{a}_3})^{n_3}$ である。よって、$\hat{T}_{\boldsymbol{R}}$ は巡回郡でもある。

ここで、結晶の巨視的な境界条件を与える。今、結晶のサイズが $N_1\boldsymbol{a}_1 \times N_2\boldsymbol{a}_2 \times N_3\boldsymbol{a}_3$ であるとする。ここで、N_i は巨視的な数字である。境界条件として、このサイズの結晶がさらに全方向に繰り返すものと考える。これはボルン・フォン・カルマン条件と呼ばれる。この条件下では、結晶内に広がった電子の波動関数は

$$\psi(\boldsymbol{r} + N_i\boldsymbol{a}_i) = \psi(\boldsymbol{r}) \tag{14}$$

を満たすと考える。上式は、並進操作を用いて

$$\hat{T}_{N_i\boldsymbol{a}_i}\psi(\boldsymbol{r}) = \psi(\boldsymbol{r}) \tag{15}$$

と書き表すことができる。よって、独立な並進操作の数は $N = N_1N_2N_3$ ということになる。群論の言葉で

[1] アーベル群は、任意の群元 G_i と G_j に対し

$$G_iG_j = G_jG_i \tag{4}$$

が常に成り立つような集合のことである。また、任意の群元 G_i に対し、同じ群の元 G によって $G^{-1}G_iG = G_j$ のように変換した元 G_j を、G_i に共役な元という。G_i に共役な元全ての集合を、G_i の類と呼ぶ。アーベル群では、(4) から明らかなように、どの元も単独で類をなす。ゆえに、類の個数は群の位数に等しい。

今、群 G の既約表現を $D^{(\alpha)}$、位数を g とする。群論における重要な定理の一つとして知られる大直交定理

$$\sum_G D^{(\alpha)*}_{ij}(G)D^{(\beta)}_{kl}(G) = \frac{g}{d_\alpha}\delta_{\alpha\beta}\delta_{ik}\delta_{jl} \tag{5}$$

から、次の指標 $\chi^{(\alpha)}(G)$ に関する直交性の定理を導出することができる：

$$\sum_G \chi^{(\alpha)*}(G)\chi^{(\beta)}(G) = g\delta_{\alpha\beta} \tag{6}$$

同じ類での元の指標は一致する。ゆえに、類 C_k に含まれる元の数を h_k とすると

$$\sum_{k=1}^{n_c} h_k\chi^{(\alpha)*}(C_k)\chi^{(\beta)}(C_k) = g\delta_{\alpha\beta} \tag{7}$$

が成り立つことがわかる。また、証明は省くがこの式から

$$\sum_{\alpha=1}^{n_r} \chi^{(\alpha)*}(C_i)\chi^{(\beta)}(C_j) = \delta_{ij}\frac{g}{h_i} \tag{8}$$

を導くことができる。これは指標に関する二番目の定理である。

(7) は、次のような幾何学的な解釈ができる。

$$(\sqrt{h_1}\chi^{(\alpha)}_1, \sqrt{h_2}\chi^{(\alpha)}_2, \cdots, \sqrt{h_{n_c}}\chi^{(\alpha)}_{n_c}) \tag{9}$$

を n_c 次元のベクトルと見做せば、(7) はベクトルの内積が直交していることを表す。n_c 次元空間において直交しているベクトルは高々n_c 個であるので、

$$n_r \leq n_c \tag{10}$$

である。同様に、(8) から

$$n_c \leq n_r \tag{11}$$

が成り立つ。よって、$n_c = n_r$ が結論される。つまり、類の数と既約表現の数は等しい。

C_k を単位元とみなすと、単位元は単独で類をなす（つまり $h_k = 1$)。単位元の指標は既約表現の次元 d_α に一致することから、(7) より

$$\sum_{k=1}^{n_c} d_\alpha^2 = g \tag{12}$$

が得られる。アーベル群では、類の数と群の位数は一致するため、$d_\alpha = 1$ とならなければならないことがわかる。

いえば、$\hat{T}_{\boldsymbol{R}}$ を元とする巡回郡の位数が有限の数 $N = N_1N_2N_3$ となる。この (15) 式と、(13) から導かれる

$$\hat{T}_{N_i\boldsymbol{a}_i}\psi(\boldsymbol{r}) = (\hat{T}_{\boldsymbol{a}_i})^{N_i}\psi(\boldsymbol{r}) = (C_{\boldsymbol{a}_i})^{N_i}\psi(\boldsymbol{r}) \tag{16}$$

を比較すると、$C_{\boldsymbol{a}_i}$ は整数 m_i を用いて

$$C_{\boldsymbol{a}_i} = \exp\left(\frac{2\pi i}{N_i}m_i\right) \tag{17}$$

と表せる。ただし m_i は $1 \leq m_i \leq N_i - 1$ を満たすものとする。

よって、並進操作 $\hat{T}_{\boldsymbol{R}}$（$\boldsymbol{R} = n_1\boldsymbol{a}_1 + n_2\boldsymbol{a}_2 + n_3\boldsymbol{a}_3$）により、波動関数は

$$\hat{T}_{\boldsymbol{R}}\psi(\boldsymbol{r}) = (\hat{T}_{\boldsymbol{a}_1})^{n_1}(\hat{T}_{\boldsymbol{a}_2})^{n_2}(\hat{T}_{\boldsymbol{a}_3})^{n_3}\psi(\boldsymbol{r}) = \exp\left\{2\pi i\left(\frac{n_1m_1}{N_1} + \frac{n_2m_2}{N_2} + \frac{n_3m_3}{N_3}\right)\right\}\psi(\boldsymbol{r}) \tag{18}$$

のような変換を受ける。ここで、既約表現を区別するラベルとして

$$\boldsymbol{k}_m = \frac{m_1}{N_1}\boldsymbol{b}_1 + \frac{m_2}{N_2}\boldsymbol{b}_2 + \frac{m_3}{N_3}\boldsymbol{b}_3 \tag{19}$$

を導入する。$\boldsymbol{b}_i$ は逆格子空間における基本ベクトルである。$\boldsymbol{k}_m$ は、逆格子空間におけるブリルアンゾーン $\boldsymbol{b}_1 \cdot (\boldsymbol{b}_2 \times \boldsymbol{b}_3)$ を各方向 $\boldsymbol{b}_i$ に沿って N_i 分割したときのメッシュの点を表している。

逆格子ベクトルと実空間の並進ベクトルの関係 $\boldsymbol{a}_i \cdot \boldsymbol{b}_j = 2\pi\delta_{ij}$ を踏まえると

$$\hat{T}_{\boldsymbol{R}}\psi(\boldsymbol{r}) = \exp(i\boldsymbol{k}_m \cdot \boldsymbol{R})\psi(\boldsymbol{r}) \tag{20}$$

と書ける。N_i はマクロな数であり、従って $\boldsymbol{k}_m$ は厳密には離散的な値を取るが、実際にはほぼ連続した値をとるとみなすことができる。そこで $\boldsymbol{k}_m$ を添字を省略して $\boldsymbol{k}$ と書くことにしよう。

(20) の関係式と並進操作の定義から

$$\psi_{\boldsymbol{k}}(\boldsymbol{r} + \boldsymbol{R}) = \exp(i\boldsymbol{k} \cdot \boldsymbol{R})\psi_{\boldsymbol{k}}(\boldsymbol{r}) \tag{21}$$

が成り立つことが示される。これはブロッホの定理と呼ばれる。$\psi(\boldsymbol{r})$ はブロッホ関数と呼ばれる。以降は、固有状態を区別するための下付き添え字 n もしくは i を表記する。

ブロッホ関数の形としては有名な表式がもう一つある。今、ブロッホ関数を用いて、ある関数

$$u_{\boldsymbol{k}n}(\boldsymbol{r}) \equiv e^{-i\boldsymbol{k}\cdot\boldsymbol{r}}\psi_{\boldsymbol{k}n}(\boldsymbol{r}) \tag{22}$$

を定義しよう。すると、これは

$$u_{\boldsymbol{k}}(\boldsymbol{r} + \boldsymbol{R}) = e^{-i\boldsymbol{k}\cdot(\boldsymbol{r}+\boldsymbol{R})}\psi_{\boldsymbol{k}}(\boldsymbol{r} + \boldsymbol{R}) = e^{-i\boldsymbol{k}\cdot(\boldsymbol{r}+\boldsymbol{R})}e^{i\boldsymbol{k}\cdot\boldsymbol{R}}\psi_{\boldsymbol{k}}(\boldsymbol{r}) = u_{\boldsymbol{k}}(\boldsymbol{r}) \tag{23}$$

となるので、格子の周期性を持った関数であることが容易にわかる。この $u_{\boldsymbol{k}n}(\boldsymbol{r})$ を用いて

$$\psi_{\boldsymbol{k}n}(\boldsymbol{r}) = e^{i\boldsymbol{k}\cdot\boldsymbol{r}}u_{\boldsymbol{k}n}(\boldsymbol{r}) \tag{24}$$

と表したのもしばしばみられるブロッホ関数の表式である。$u_{\boldsymbol{k}n}(\boldsymbol{r})$ は格子の周期性を持つので、逆格子ベクトルで展開できる。

$$\psi_{n\boldsymbol{k}}(\boldsymbol{r}) = \exp(i\boldsymbol{k} \cdot \boldsymbol{r})\sum_{\boldsymbol{G}} u_{\boldsymbol{k}n}(\boldsymbol{G})\exp(i\boldsymbol{G} \cdot \boldsymbol{r}) = \sum_{\boldsymbol{G}} u_{\boldsymbol{k}n}(\boldsymbol{G})\exp[i(\boldsymbol{k} + \boldsymbol{G}) \cdot \boldsymbol{r}] \tag{25}$$

よって、ブロッホ関数は平面波で展開して表したものとみなすことができる。$\psi_{\boldsymbol{k}n}(\boldsymbol{r})$ は、次の $|\psi_{\boldsymbol{k}n}\rangle$ の位置表示であるとする：

$$|\psi_{\boldsymbol{k}n}\rangle = \sum_{\boldsymbol{G}} C^n_{\boldsymbol{k}+\boldsymbol{G}}\,|\boldsymbol{k} + \boldsymbol{G}\rangle \tag{26}$$

$C^n_{\boldsymbol{k}+\boldsymbol{G}}$ は展開係数である。$|\boldsymbol{k}+\boldsymbol{G}\rangle$ は平面波基底であり、その位置表示は

$$\langle \boldsymbol{r}|\boldsymbol{k}+\boldsymbol{G}\rangle = \frac{1}{\sqrt{\Omega_{\rm cell}}} e^{i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}} \tag{27}$$

である。係数の $1/\sqrt{\Omega_{\rm cell}}$ は、平面波基底がユニットセル内で規格化されているとしていることに起因する。実際、

$$\langle \boldsymbol{k}+\boldsymbol{G}|\boldsymbol{k}+\boldsymbol{G'}\rangle = \int_{\rm cell} d\boldsymbol{r}\,\langle \boldsymbol{k}+\boldsymbol{G}|\boldsymbol{r}\rangle\langle \boldsymbol{r}|\boldsymbol{k}+\boldsymbol{G'}\rangle = \frac{1}{\Omega_{\rm cell}}\int_{\rm cell} d\boldsymbol{r} e^{i(\boldsymbol{G'}-\boldsymbol{G})\cdot\boldsymbol{r}} = \delta_{\boldsymbol{G'G}} \tag{28}$$

となり、確かに規格化されていることがわかる。(25) と

$$\psi_{\boldsymbol{k}n}(\boldsymbol{r}) \equiv \langle \boldsymbol{r}|\psi_{\boldsymbol{k}n}\rangle = \frac{1}{\sqrt{\Omega_{\rm cell}}}\sum_{\boldsymbol{G}} C^n_{\boldsymbol{k}+\boldsymbol{G}} e^{i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}} \tag{29}$$

とを比較すると、$\Omega_{\rm cell}^{-1/2} C^n_{\boldsymbol{k}+\boldsymbol{G}} = u_{\boldsymbol{k}n}(\boldsymbol{G})$ であるとわかる。

2　平面波を基底とした Kohn-Sham 方程式

平面波第一原理電子状態計算では、KS 方程式を解く際、波動関数や電子密度といった物理量を平面波で展開した形で表し、逆空間の問題に置き換える。今、固体のように原子が周期的に並んだ系での電子の固有状態を求めたいとする。すなわち、周期 $\boldsymbol{R}$ を持つポテンシャル $\hat{V}_{\rm eff}(\boldsymbol{r})$ の下での、時間に依存しない一電子系のシュレーディンガー方程式を解くことを考える：

$$\hat{H}\,|\psi_{i\boldsymbol{k}}\rangle = \varepsilon_{i\boldsymbol{k}}\,|\psi_{i\boldsymbol{k}}\rangle\,, \qquad \hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + \hat{V}_{\rm eff}(\boldsymbol{r}) \tag{30}$$

平面波で展開した $|\psi_{i\boldsymbol{k}}\rangle$ を Kohn-Sham 方程式に代入し、左から $\langle \boldsymbol{k}+\boldsymbol{G}|$ を掛けると、以下の行列方程式が得られる。

$$\sum_{G'}\left\langle \boldsymbol{k}+\boldsymbol{G}\,\middle|\,\hat{H}\,\middle|\,\boldsymbol{k}+\boldsymbol{G'}\right\rangle C^i_{\boldsymbol{k}+\boldsymbol{G'}} = \varepsilon_{i\boldsymbol{k}} C^i_{\boldsymbol{k}+\boldsymbol{G}} \tag{31}$$

$H_{\boldsymbol{k}+\boldsymbol{G},\boldsymbol{k}+\boldsymbol{G'}}$ の具体的な形を見てみよう。運動エネルギー項は、

$$\begin{aligned}\left\langle \boldsymbol{k}+\boldsymbol{G}\,\middle|\,\left(-\frac{\hbar^2}{2m}\nabla^2\right)\,\middle|\,\boldsymbol{k}+\boldsymbol{G'}\right\rangle &= \int_{\Omega_{\rm cell}} d\boldsymbol{r}\,\langle \boldsymbol{k}+\boldsymbol{G}|\boldsymbol{r}\rangle\left(-\frac{\hbar^2}{2m}\nabla^2\right)\langle \boldsymbol{r}|\boldsymbol{k}+\boldsymbol{G'}\rangle \\ &= \frac{1}{\Omega_{\rm cell}}\frac{\hbar^2}{2m}|\boldsymbol{k}+\boldsymbol{G'}|^2\int_{\Omega_{\rm cell}} d\boldsymbol{r} e^{i(\boldsymbol{G'}-\boldsymbol{G})\cdot\boldsymbol{r}} = \frac{\hbar^2}{2m}|\boldsymbol{k}+\boldsymbol{G'}|^2\delta_{\boldsymbol{G'G}}\end{aligned} \tag{32}$$

KS 方程式の有効ポテンシャルは、ハートレー項 $V_{\rm H}(\boldsymbol{r})$ と交換相関項 $V_{\rm xc}(\boldsymbol{r})$、さらに擬ポテンシャル項で構成される。擬ポテンシャル項はさらに射影演算子を含む非局所項 $\hat{V}^{\rm PS}_{\rm NL}(\boldsymbol{r})$ と含まない局所項 $\hat{V}^{\rm PS}_{\rm L}(\boldsymbol{r})$ に分けられる。有効ポテンシャルのうち、$\hat{V}^{\rm PS}_{\rm NL}(\boldsymbol{r})$ を除いた部分を $V_{\rm L}(\boldsymbol{r})$ と定義すると、その行列要素は

$$\begin{aligned}\langle \boldsymbol{k}+\boldsymbol{G}|V_{\rm L}(\boldsymbol{r})|\boldsymbol{k}+\boldsymbol{G'}\rangle &= \frac{1}{\Omega_{\rm cell}}\int_{\Omega_{\rm cell}} d\boldsymbol{r} e^{-i(\boldsymbol{k}+\boldsymbol{G})}\Big(\sum_{\boldsymbol{G''}} V_{\rm L}(\boldsymbol{G''})e^{i\boldsymbol{G''}\cdot\boldsymbol{r}}\Big)e^{i(\boldsymbol{k}+\boldsymbol{G'})} \\ &= \frac{1}{\Omega_{\rm cell}}\sum_{\boldsymbol{G''}} V_{\rm L}(\boldsymbol{G''})\int_{\Omega_{\rm cell}} d\boldsymbol{r} e^{i(\boldsymbol{G''}-\boldsymbol{G}+\boldsymbol{G'})} = V_{\rm L}(\boldsymbol{G}-\boldsymbol{G'})\end{aligned} \tag{33}$$

のように与えられる。

3　擬ポテンシャル法

4　全エネルギーの波数空間表示

SCF 計算が一通り回り電子密度が決まると、全エネルギー $E_{\rm tot}$ が求まる。$E_{\rm tot}$ の波数空間での表現は

$$
\begin{aligned}
E_{\rm tot} &= \sum_{n}^{occ}\sum_{\boldsymbol{k}_i} w_{n\boldsymbol{k}_i} \sum_{\boldsymbol{G}} \frac{\hbar^2}{2m}|\boldsymbol{k}_i+\boldsymbol{G}|^2 + \Omega_{\rm cell}\sum_{\boldsymbol{G}\neq 0} V^{\rm PS}_{\rm local}(\boldsymbol{G})n(-\boldsymbol{G}) + \frac{N_c}{\Omega_{\rm cell}}\sum_a \alpha_a \\
&+ \sum_{n}^{occ}\sum_{\boldsymbol{k}_i} w_{n\boldsymbol{k}_i} \sum_a \sum_l C^{-1}_{a,l} \sum_{m=-l}^{+l} |\sum_{\boldsymbol{G}} C^n_{\boldsymbol{k}_i+\boldsymbol{G}} A_{a,lm}(\boldsymbol{k}_i+\boldsymbol{G})|^2 + \frac{1}{2}4\pi e^2\Omega_{\rm cell}\sum_{\boldsymbol{G}\neq 0}\frac{|n(\boldsymbol{G})|^2}{|\boldsymbol{G}|^2} \\
&+ \Omega_{\rm cell}\sum_{\boldsymbol{G}} \varepsilon_{\rm xc}(\boldsymbol{G})n(-\boldsymbol{G}) + E_{\rm ewald}
\end{aligned}
\tag{34}
$$

証明は第一部で示した文献に譲る。

5　SCF 計算の大まかな流れ

KS 方程式中に現れる有効ポテンシャルには、電子密度が含まれており、KS 方程式の解として得られる波動関数から得られる電子密度と一致していなければならない。以下に、SCF 計算の手順を示す。

1. 初期の電子密度 $n^{\rm in}(\boldsymbol{r})$ から有効ポテンシャルを決める。
2. Kohn-Sham 方程式の固有値と固有状態を決定する。
3. 占有された固有状態の 2 乗の BZ 内積分から、電子密度 $n^{\rm out}(\boldsymbol{r})$ を決定する。
4. $n^{\rm in}(\boldsymbol{r})$ と $n^{\rm out}(\boldsymbol{r})$ を比べ、収束条件を満たしてなけらば、$n^{\rm in}(\boldsymbol{r})$ と $n^{\rm out}(\boldsymbol{r})$ を用いて電子密度を更新する。

上記のプロセスを、電子密度が収束条件を満たすまで繰り返す。

- `run_pwscf.f90`: scf 計算の起点
- `init_run.f90`: 初期の電子密度と有効ポテンシャルを決定
- `electrons.f90`: scf ループ。サブルーチン electrons_scf が scf サイクルを動かしている。
- `v_of_rho.f90`: 有効ポテンシャルを更新
- `c_bands.f90`: 与えられた Hamiltonian に対して対角化
- `sum_band.f90`: 電子密度を計算
- `mix_rho.f90`: in と out の電子密度を混合。次の iteration の入力の電子密度として使われる。(iteration 前後の電子密度の差分から、収束判定も行っている？)

6　固有状態計算アルゴリズム

第一原理電子状態計算では、Kohn-Sham（KS）方程式と呼ばれる二階偏微分方程式を解く。QE においてこのプロセスが記述されているのが `c_bands.f90` である。

■ Davidson 法によるハミルトニアンの対角化

各 $\boldsymbol{k}$ に対し、次の一般化固有値問題を解くことを考える：

$$\hat{H}\left|\psi_i\right\rangle = \epsilon_i \hat{S}\left|\psi_i\right\rangle, \qquad i = 1, \ldots, N_G \tag{35}$$

$\hat{H}$ は $N_G \times N_G$ のハミルトニアン行列、$\hat{S}$ は $N_G \times N_G$ の重なり積分 (overlap matrix) の行列を表し、その行列要素は

$$\left\langle \psi_i \middle| \hat{S} \middle| \psi_j \right\rangle = \delta_{ij} \tag{36}$$

で与えられる。

N_G が大きくなると対角化計算が容易ではなくなってくる。しかし、すべての固有状態が必要ではなく、通常はユニットセルに含まれる電子数のオーダー程度である。今、$nbnd$ 個の固有状態を得たいとする。もし、$nbnd \ll N$ であるとすると、以下で述べる反復法が有効になる。この方法では、ある行列 A をベクトル $\boldsymbol{b}^{(0)}$ に繰り返し作用させ

$$\mathcal{K} = \left\{ \boldsymbol{b}^{(0)}, A\boldsymbol{b}^{(0)}, \cdots, A^{n-1}\boldsymbol{b}^{(0)} \right\} \tag{37}$$

という空間（Krylov 部分空間）を作り、この空間における基底を用いて固有値問題を解く。この方法により、求めたい $nbnd$ 個の解を、元の基底の次元 N よりも少ない数の基底を使って近似的に計算することが可能になる。

QE では、反復法の一つである davidson 法と呼ばれる方法がデフォルトとして採用されている。以下、davidson 法のアルゴリズムを示す。まず、試行ベクトル $\left|\psi_i^{(0)}\right\rangle$ により張られる空間

$$\mathcal{K} = \left\{ \left|\psi_1^{(0)}\right\rangle, \left|\psi_2^{(0)}\right\rangle, \cdots, \left|\psi_{nbnd}^{(0)}\right\rangle \right\} \tag{38}$$

と近似的な固有値 $\epsilon_i^{(0)} = \left\langle \psi_i^{(0)} \middle| H \middle| \psi_i^{(0)} \right\rangle$ が予め決められているとする。これを元に、次の修正ベクトル (correction vector) と呼ばれるベクトル

$$\left|\delta\psi_i^{(0)}\right\rangle = D^{-1}(H - \epsilon_i^{(n)} S)\left|\psi_i^{(0)}\right\rangle \tag{39}$$

を導入する。ここで、D は

$$D = H_{mm}\delta_{mm'} - \epsilon_i^{(n)} I \tag{40}$$

で与えられる。[2] 修正ベクトル $\left|\delta\psi_i^{(0)}\right\rangle$ から、$\left|\tilde{\psi}_i^{(1)}\right\rangle = \left|\psi_i^{(0)}\right\rangle$, $\left|\tilde{\psi}_{i+nbnd}^{(1)}\right\rangle = \left|\delta\psi_i^{(0)}\right\rangle$ として、$2nbnd$ 次元の空間

$$\mathcal{K} = \left\{ \left|\tilde{\psi}_1^{(1)}\right\rangle, \left|\tilde{\psi}_2^{(1)}\right\rangle, \cdots, \left|\tilde{\psi}_{nbnd}^{(1)}\right\rangle, \left|\tilde{\psi}_{nbnd+1}^{(1)}\right\rangle, \left|\tilde{\psi}_{nbnd+2}^{(1)}\right\rangle, \cdots, \left|\tilde{\psi}_{2nbnd}^{(1)}\right\rangle \right\} \tag{42}$$

を作る。この拡張された空間において、ハミルトニアン行列 $\tilde{H}_{ij}^{(1)} = \left\langle \tilde{\psi}_i^{(1)} \middle| H \middle| \tilde{\psi}_j^{(1)} \right\rangle$ を構築し、対角化により $nbnd$ 個の固有値と固有ベクトル $\left|\psi_i^{(1)}\right\rangle$ を求める。

こうして求まった固有ベクトルをもとに再び修正ベクトルを作り、部分空間 (42) に新たな基底として加え、ハミルトニアン行列（$3nbnd \times 3nbnd$）を作り、対角化により固有値と固有ベクトルを求める。これを、固有値が収束するまで繰り返す。

---

[2]

$$\left|R_i^{(n)}\right\rangle = (H - \epsilon_i^{(n)} S)\left|\psi_i^{(n)}\right\rangle \tag{41}$$

は残差ベクトル（residual vector）と呼ばれる。

7　高速フーリエ変換による実空間と逆空間の行き来

平面波を基底にとったときの電子状態計算では、ハミルトニアンや全エネルギー、電子密度などが波数空間で表現される。しかし、大規模な計算を実行するにあたり、計算負荷を減らすには、実際には実空間メッシュ点での値を求め、逆空間との間でフーリエ変換を通じて行き来することが必要となる。このとき、高速フーリエ変換（Fast Fourier Transform; FFT）が使われる。

FFT は離散フーリエ変換であり、次のような実空間メッシュ点ごとにデータが算出される：

$$\boldsymbol{r}_n = \frac{n_1}{M_1}\boldsymbol{a}_1 + \frac{n_2}{M_2}\boldsymbol{a}_2 + \frac{n_3}{M_3}\boldsymbol{a}_3 \tag{43}$$

（n_i は整数で、$0 \leq n_i \leq M_i - 1$）ここで、M_i（$i = 1, 2, 3$）は、平行六面体のユニットセルを張る基本並進ベクトル $\boldsymbol{a}_i$ の方向におけるメッシュの分割数である。FFT は、実空間メッシュ点の数 $M = M_1 M_2 M_3$ と同数の逆格子点

$$\boldsymbol{G}_m = m_1\boldsymbol{b}_1 + m_2\boldsymbol{b}_2 + m_3\boldsymbol{b}_3 \tag{44}$$

（m_i は整数で、$0 \leq m_i \leq M_i - 1$）において、

$$f(\boldsymbol{G}_m) = \frac{1}{M_1 M_2 M_3} \sum_{n_1=0}^{M_1-1} \sum_{n_2=0}^{M_2-1} \sum_{n_3=0}^{M_3-1} f(\boldsymbol{r}_n) \exp\left(-i\frac{2\pi n_1 m_1}{M_1}\right) \exp\left(-i\frac{2\pi n_2 m_2}{M_2}\right) \exp\left(-i\frac{2\pi n_3 m_3}{M_3}\right) \tag{45}$$

のように計算される。逆 FFT は

$$f(\boldsymbol{r}_n) = \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \sum_{m_3=0}^{M_3-1} f(\boldsymbol{G}_m) \exp\left(i\frac{2\pi n_1 m_1}{M_1}\right) \exp\left(i\frac{2\pi n_2 m_2}{M_2}\right) \exp\left(i\frac{2\pi n_3 m_3}{M_3}\right) \tag{46}$$

である。

メッシュ点の数 $M = M_1 M_2 M_3$ は、計算する系のユニットセルとカットオフエネルギーが与えられるとプログラムが自動的に決める。

第 III 部
ソースコードを覗く

以下のソースコードは SCF 計算において重要なプロセスを担う部分であり、`q-e-qe-5.1.0/PW/src/` に格納されている。

8　scf 計算の起点

Listing 1: PW/src/pwscf.f90

```
8   !----------------------------------------------------------------------------
9  PROGRAM pwscf
10  !----------------------------------------------------------------------------
11  !
12  ! ... Main program calling one instance of Plane Wave Self-Consistent Field
        code
13  !
14  USE environment, ONLY : environment_start
15  USE mp_global, ONLY : mp_startup
```

```
16   USE read_input, ONLY : read_input_file
17   USE command_line_options, ONLY: input_file_
18   !
19   IMPLICIT NONE
20   INTEGER :: exit_status
21   !
22   !
23   CALL mp_startup ( )
24   CALL environment_start ( 'PWSCF' )
25   !
26   CALL read_input_file ('PW', input_file_ )
27   !
28   ! ... Perform actual calculation
29   !
30   CALL run_pwscf ( exit_status )
31   !
32   CALL stop_run( exit_status )
33   CALL do_stop( exit_status )
34   !
35   STOP
36   !
37 END PROGRAM pwscf
```

30 行目でコールしているサブルーチンである run_scf が、SCF 計算，もしくは Non-SCF 計算のメインプログラムを呼び出している。内容は次のとおりである：

Listing 2: PW/src/run_pwscf.f90

```
8  !----------------------------------------------------------------------------
9  SUBROUTINE run_pwscf ( exit_status )
10   !----------------------------------------------------------------------------
11   !
12   ! ... Run an instance of the Plane Wave Self-Consistent Field code
13   ! ... MPI initialization and input data reading is performed in the
14   ! ... calling code - returns in exit_status the exit code for pw.x,
15   ! ... returned in the shell. Values are:
16   ! ... * 0: completed successfully
17   ! ... * 1: an error has occurred (value returned by the errore() routine)
18   ! ... * 2-127: convergence error
19   ! ... * 2: scf convergence error
20   ! ... * 3: ion convergence error
21   ! ... * 128-255: code exited due to specific trigger
22   ! * 255: exit due to user request, or signal trapped,
23   ! or time > max_seconds
24   ! ... (note: in the future, check_stop_now could also return a value
25   ! ... to specify the reason of exiting, and the value could be used
26   ! .. to return a different value for different reasons)
27   ! ... Will be eventually merged with NEB
28   !
29   USE io_global, ONLY : stdout, ionode, ionode_id
30   USE parameters, ONLY : ntypx, npk, lmaxx
```

```
31  USE cell_base, ONLY : fix_volume, fix_area
32  USE control_flags, ONLY : conv_elec, gamma_only, lscf
33  USE control_flags, ONLY : conv_ions, istep, nstep, restart, lmd, lbfgs
34  USE force_mod, ONLY : lforce, lstres, sigma, force
35  USE check_stop, ONLY : check_stop_init, check_stop_now
36  USE mp_images, ONLY : intra_image_comm
37  USE qmmm, ONLY : qmmm_initialization, qmmm_shutdown, &
38                               qmmm_update_positions, qmmm_update_forces
39  !
40  IMPLICIT NONE
41  INTEGER, INTENT(OUT) :: exit_status
42  !
43  !
44  exit_status = 0
45  IF ( ionode ) WRITE( unit = stdout, FMT = 9010 ) ntypx, npk, lmaxx
46  !
47  IF (ionode) CALL plugin_arguments()
48  CALL plugin_arguments_bcast( ionode_id, intra_image_comm )
49  !
50  ! ... needs to come before iosys() so some input flags can be
51  ! overridden without needing to write PWscf specific code.
52  !
53  CALL qmmm_initialization()
54  !
55  ! ... convert to internal variables
56  !
57  CALL iosys()
58  !
59  IF ( gamma_only ) WRITE( UNIT = stdout, &
60     & FMT = '(/,5X,"gamma-point␣specific␣algorithms␣are␣used")' )
61  !
62  ! call to void routine for user defined / plugin patches initializations
63  !
64  CALL plugin_initialization()
65  !
66  CALL check_stop_init()
67  !
68  CALL setup ()
69  !
70  CALL qmmm_update_positions()
71  !
72  CALL init_run()
73  !
74  ! ... dry run: code will stop here if called with exit file present
75  ! ... useful for a quick and automated way to check input data
76  !
77  IF ( check_stop_now() ) THEN
78     CALL punch( 'config' )
79     exit_status = 255
80     RETURN
81  ENDIF
82  !
83  main_loop: DO
```

```
84      !
85      ! ... electronic self-consistency or band structure calculation
86      !
87      IF ( .NOT. lscf) THEN
88         CALL non_scf ()
89      ELSE
90         CALL electrons()
91      END IF
92      !
93      ! ... code stopped by user or not converged
94      !
95      IF ( check_stop_now() .OR. .NOT. conv_elec ) THEN
96         IF ( check_stop_now() ) exit_status = 255
97         IF ( .NOT. conv_elec ) exit_status = 2
98         CALL punch( 'config' )
99         RETURN
100     ENDIF
101     !
102     ! ... ionic section starts here
103     !
104     CALL start_clock( 'ions' )
105     conv_ions = .TRUE.
106     !
107     ! ... recover from a previous run, if appropriate
108     !
109     !IF ( restart .AND. lscf ) CALL restart_in_ions()
110     !
111     ! ... file in CASINO format written here if required
112     !
113     IF ( lmd ) CALL pw2casino()
114     !
115     ! ... force calculation
116     !
117     IF ( lforce ) CALL forces()
118     !
119     ! ... stress calculation
120     !
121     IF ( lstres ) CALL stress ( sigma )
122     !
123     ! ... send out forces to MM code in QM/MM run
124     !
125     CALL qmmm_update_forces(force)
126     !
127     IF ( lmd .OR. lbfgs ) THEN
128        !
129        if (fix_volume) CALL impose_deviatoric_stress(sigma)
130        !
131        if (fix_area) CALL impose_deviatoric_stress_2d(sigma)
132        !
133        ! ... ionic step (for molecular dynamics or optimization)
134        !
135        CALL move_ions()
136        !
```

```
137         ! ... then we save restart information for the new configuration
138         !
139         IF ( istep < nstep .AND. .NOT. conv_ions ) &
140            CALL punch( 'config' )
141         !
142      END IF
143      !
144      CALL stop_clock( 'ions' )
145      !
146      ! ... exit condition (ionic convergence) is checked here
147      !
148      IF ( conv_ions ) EXIT main_loop
149      !
150      ! ... receive new positions from MM code in QM/MM run
151      !
152      CALL qmmm_update_positions()
153      !
154      ! ... terms of the hamiltonian depending upon nuclear positions
155      ! ... are reinitialized here
156      !
157      IF ( lmd .OR. lbfgs ) CALL hinit1()
158      !
159   END DO main_loop
160   !
161   ! ... save final data file
162   !
163   IF ( .not. lmd) CALL pw2casino()
164   CALL punch('all')
165   !
166   CALL qmmm_shutdown()
167   !
168   IF ( .NOT. conv_ions ) exit_status = 3
169   RETURN
170   !
171 9010 FORMAT( /,5X,'Current␣dimensions␣of␣program␣PWSCF␣are:', &
172             & /,5X,'Max␣number␣of␣different␣atomic␣species␣(ntypx)␣=␣',I2,&
173             & /,5X,'Max␣number␣of␣k-points␣(npk)␣=␣',I6,&
174             & /,5X,'Max␣angular␣momentum␣in␣pseudopotentials␣(lmaxx)␣=␣',i2)
175   !
176 END SUBROUTINE run_pwscf
```

83〜159 行にわたる DO ループで、与えられた原子配置に対して SCF 計算を行う。90 行目でコールされるサブルーチン electrons() で SCF 計算が走る。electron() は PW/src/electrons.f90 に記述されている。117 行目では各原子にかかるヘルマン・ファインマン力が計算される。力の計算は構造最適化計算において必要である。121 行目でセルにかかる応力の計算がなされる。

electrons.f90 では、一番目のサブルーチン electrons() をまずは見て、146 行目で electrons_scf() をコールする。このサブルーチン electrons_scf() が、SCF 計算のループを動かしており、ループとなる DO 文は次のようにはじまる：

Listing 3: PW/src/electrons.f90

```
DO idum = 1, niter
   !
   IF ( check_stop_now() ) THEN
      conv_elec=.FALSE.
      CALL save_in_electrons (iter, dr2, et )
      GO TO 10
   END IF
   iter = iter + 1
   !
   WRITE( stdout, 9010 ) iter, ecutwfc, mixing_beta
   !
   CALL flush_unit( stdout )
   !
   ! ... Convergence threshold for iterative diagonalization is
   ! ... automatically updated during self consistency
   !
   IF ( iter > 1 ) THEN
      !
      IF ( iter == 2 ) ethr = 1.D-2
      ethr = MIN( ethr, 0.1D0*dr2 / MAX( 1.D0, nelec ) )
      ! ... do not allow convergence threshold to become too small:
      ! ... iterative diagonalization may become unstable
      ethr = MAX( ethr, 1.D-13 )
      !
   END IF
   !
   first = ( iter == 1 )
   !
   ! ... deband = - \sum_v <\psi_v | V_h + V_xc |\psi_v> is calculated a
   ! ... first time here using the input density and potential ( to be
   ! ... used to calculate the Harris-Weinert-Foulkes energy )
   !
   deband_hwf = delta_e()
   !
   ! ... save input density to rhoin
   !
   call scf_type_COPY( rho, rhoin )
   !
   scf_step: DO
      !
      ! ... tr2_min is set to an estimate of the error on the energy
      ! ... due to diagonalization - used only for the first scf iteration
      !
      tr2_min = 0.D0
      !
      IF ( first ) tr2_min = ethr*MAX( 1.D0, nelec )
      !
      ! ... diagonalization of the KS hamiltonian
      !
      IF ( lelfield ) THEN
         CALL c_bands_efield ( iter )
      ELSE
```

```
481             CALL c_bands( iter )
482         END IF
```

481 行でコールされる c.bands() において、KS 方程式の固有状態計算が行われる。

9 固有状態計算（c_bands.f90）

サブルーチン c_bands() は、ハミルトニアンの対角化を行いエネルギー固有値と固有ベクトルを各 $\boldsymbol{k}$ 点において計算するプロセスのドライバーである。ハミルトニアンの各項と、初期の波動関数データを読み込み、計算が回る。

c_bands.f90 において、66〜128 行が $\boldsymbol{k}$ に関する DO ループ（ik=1, nks）である。nks は各 pool が担当する $\boldsymbol{k}$ 点の数である。106 行目で diag_bands(iter, ik, avg_iter) をコールし、さらにそのサブルーチン内の 217〜225 行目で、diag_bands_gamma() と diag_bands_k() をコールする。前者が $\boldsymbol{k}$ 点のサンプリングがガンマ点のみの場合であり、そうでない場合は後者がコールされる。さらにその内側で繰り返し対角化法の計算が行われる。

Quantum ESPRESSO では、繰り返し対角化法として、ダビッドソン (Davidson) 法と共役勾配法（Conjugate Gradient）が実装されている。デフォルトは前者となっており、ここではダビッドソン法について見ていく。diag_bands_k() だと、455〜484 行でダビッドソンループが記述されており、具体的な計算は、サブルーチン regterg() と cegterg() [3] に記述されている。前者はガンマ点のみの計算に対して最適化されたものである。ここでは後者の cegterg() を見ていく。なお、これら 2 つのサブルーチンは q-e-qe-5.1.0/PW/src/ 内の `regterg.f90` と `cegterg.f90` に含まれている[4]。

Listing 4: PW/src/cegterg.f90

```
12 !----------------------------------------------------------------------------
13 SUBROUTINE cegterg( npw, npwx, nvec, nvecx, npol, evc, ethr, &
14                     uspp, e, btype, notcnv, lrot, dav_iter )
15   !----------------------------------------------------------------------------
16   !
17   ! ... iterative solution of the eigenvalue problem:
18   !
19   ! ... ( H - e S ) * evc = 0
20   !
21   ! ... where H is an hermitean operator, e is a real scalar,
22   ! ... S is an overlap matrix, evc is a complex vector
23   !
```

cegterg() で用いられる変数リスト

[3] regterg = real eigen iterative generalized
cegterg = complex eigen iterative generalized

[4] qe の最近のバージョンだと、PW/src/ではなく、KS_Solvers/Davidson の方に格納されている。

変数	型	意味
npw	整数	平面波基底の数
npwx	整数	平面波の数の最大数
nvec	整数	求める固有状態の数。c_band() の nbnd から引き継がれる。
nvecx	整数	
npol	整数	スピン偏極の数
ethr	実数	収束の閾値
evc(npwx,npol,nvec)	実数配列	固有ベクトル
uspp	論理変数	ウルトラソフト擬ポテンシャル法の場合に真
btype(nvec)	整数型の配列	占有バンドだと 1 で、非占有バンドだと 0
lrot	論理変数	
e(nvec)	実数配列	バンド nvec における固有値
dav_iter	整数	ダビッドソン法の反復回数
notcnv	整数	収束しなかった固有値の数
nbase	整数	各イタレーション毎の拡張された空間の次元

cegterg() を上から見ていこう。まず、104〜105 行目でローカル変数 kdim,kdmx にそれぞれ npw,npwx が代入されることである。さらに、143 145 行目で、ローカル変数 notconv,nbase,conv にそれぞれ nvec,nvec, そして.FALSE. が代入される。

9.1 初期ベクトルの処理とハミルトニアンと固有ベクトルの積の演算

以下の部分では、初期の固有ベクトルと、それにハミルトニアンを掛け算するプロセスが記述されている。

Listing 5: PW/src/cegterg.f90

```
143   notcnv = nvec
144   nbase = nvec
145   conv = .FALSE.
146   !
147   IF ( uspp ) spsi = ZERO
148   !
149   hpsi = ZERO
150   psi = ZERO
151   psi(:,:,1:nvec) = evc(:,:,1:nvec)
152   !
153   ! ... hpsi contains h times the basis vectors
154   !
155   CALL h_psi( npwx, npw, nvec, psi, hpsi )
156   !
157   ! ... spsi contains s times the basis vectors
158   !
159   IF ( uspp ) CALL s_psi( npwx, npw, nvec, psi, spsi )
```

151 行目で、何らかの方法で推定された初期の固有ベクトルの配列 evc を、配列 psi に代入する。

$$\mathcal{K} = \left\{ \left|\psi_1^{(0)}\right\rangle, \left|\psi_2^{(0)}\right\rangle, \cdots, \left|\psi_{nvec}^{(0)}\right\rangle \right\} \tag{47}$$

$\left|\psi_i^{(0)}\right\rangle$ は列ベクトルとして表せ、

$$\left|\psi_i^{(0)}\right\rangle = \begin{bmatrix} \left\langle \boldsymbol{k}+\boldsymbol{G}_1 \middle| \psi_i^{(0)} \right\rangle \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_2 \middle| \psi_i^{(0)} \right\rangle \\ \vdots \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_{npw} \middle| \psi_i^{(0)} \right\rangle \end{bmatrix} = \begin{bmatrix} C^i_{\boldsymbol{k}+\boldsymbol{G}_1} \\ C^i_{\boldsymbol{k}+\boldsymbol{G}_2} \\ \vdots \\ C^i_{\boldsymbol{k}+\boldsymbol{G}_{npw}} \end{bmatrix} \tag{48}$$

である。つまり、配列 psi はスピン部分を除くと、npw×nvec 次元のデータである。

この psi に対し、155 行目でコールされるサブルーチン hpsi（spsi）で H（S）と基底ベクトル psi の積が出力される。

$$\left\{ H\left|\psi_1^{(0)}\right\rangle, H\left|\psi_2^{(0)}\right\rangle, \cdots, H\left|\psi_{nvec}^{(0)}\right\rangle \right\} \tag{49}$$

なお、hpsi（spsi）は q-e-qe-5.1.0/PW/src/ の中にある `h_psi.f90`（`s_psi.f90`）に記述されている。ここで、h_psi() を見てみよう：

Listing 6: PW/src/h_psi.f90

```
8  !----------------------------------------------------------------------------
9  SUBROUTINE h_psi( lda, n, m, psi, hpsi )
10   !--------------------------------------------------------------------------
11   !
12   ! ... This routine computes the product of the Hamiltonian
13   ! ... matrix with m wavefunctions contained in psi
14   !
15   ! ... input:
16   ! ... lda leading dimension of arrays psi, spsi, hpsi
17   ! ... n true dimension of psi, spsi, hpsi
18   ! ... m number of states psi
19   ! ... psi
20   !
21   ! ... output:
22   ! ... hpsi H*psi
23   !
```

cegterg() から、各変数が引き継がれる。具体的には、npwx が lda に、npw が n に、nvec が m に引き渡す。hpsi を計算し、cegterg() に返す。

まず初めに、ハミルトニアンの運動エネルギー項と波動関数の積が

$$\begin{pmatrix} \frac{\hbar^2}{2m}|\boldsymbol{k}+\boldsymbol{G}_1|^2 & & & \\ & \frac{\hbar^2}{2m}|\boldsymbol{k}+\boldsymbol{G}_2|^2 & 0 & \\ & & \ddots & \\ & 0 & & \frac{\hbar^2}{2m}|\boldsymbol{k}+\boldsymbol{G}_{npw}|^2 \end{pmatrix} \times \begin{pmatrix} \left\langle \boldsymbol{k}+\boldsymbol{G}_1 \middle| \psi_{ibnd}^{(0)} \right\rangle \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_2 \middle| \psi_{ibnd}^{(0)} \right\rangle \\ \vdots \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_{npw} \middle| \psi_{ibnd}^{(0)} \right\rangle \end{pmatrix} \tag{50}$$

のように計算され、その記述は以下のとおりである：

Listing 7: PW/src/h_psi.f90

```
8   ! ... Here we apply the kinetic energy (k+G)^2 psi
9   !
10  DO ibnd = 1, m
11     hpsi (1:n, ibnd) = g2kin (1:n) * psi (1:n, ibnd)
12     hpsi (n+1:lda,ibnd) = (0.0_dp, 0.0_dp)
13     IF ( noncolin ) THEN
14        hpsi (lda+1:lda+n, ibnd) = g2kin (1:n) * psi (lda+1:lda+n, ibnd)
15        hpsi (lda+n+1:lda*npol, ibnd) = (0.0_dp, 0.0_dp)
16     END IF
17  END DO
```

上記の運動エネルギーと波動関数の積の計算部分に後続する箇所で、ポテンシャル項と波動関数の積を計算するサブルーチンをコールしている。112 行でコールされる vloc_psi_gamma() と、121 行でコールされる vloc_psi_k() では、有効ポテンシャルの局所項（ハートレー項＋交換相関項＋擬ポテンシャルの局所項）と波動関数の積が計算される。前者がガンマ点のみの計算に対し最適化されたものである。ここでは後者の方を見ていく：

Listing 8: PW/src/vloc_psi.f90

```
8   !
9   !----------------------------------------------------------------------
10  SUBROUTINE vloc_psi_k(lda, n, m, psi, v, hpsi)
11    !----------------------------------------------------------------------
12    !
13    ! Calculation of Vloc*psi using dual-space technique - k-points
14    !
```

局所ポテンシャルと波動関数の積を波数空間で表すと次のようになる。

$$\begin{pmatrix} V_{\mathrm{L}}(0) & V_{\mathrm{L}}(\boldsymbol{G}_1-\boldsymbol{G}_2) & \dots & V_{\mathrm{L}}(\boldsymbol{G}_1-\boldsymbol{G}_{npw}) \\ V_{\mathrm{L}}(\boldsymbol{G}_2-\boldsymbol{G}_1) & V_{\mathrm{L}}(0) & \dots & V_{\mathrm{L}}(\boldsymbol{G}_2-\boldsymbol{G}_{npw}) \\ \vdots & \vdots & \ddots & \vdots \\ V_{\mathrm{L}}(\boldsymbol{G}_{npw}-\boldsymbol{G}_1) & V_{\mathrm{L}}(\boldsymbol{G}_{npw}-\boldsymbol{G}_2) & \dots & V_{\mathrm{L}}(0) \end{pmatrix} \times \begin{pmatrix} \left\langle \boldsymbol{k}+\boldsymbol{G}_1 \middle| \psi^{(0)}_{ibnd} \right\rangle \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_2 \middle| \psi^{(0)}_{ibnd} \right\rangle \\ \vdots \\ \left\langle \boldsymbol{k}+\boldsymbol{G}_{npw} \middle| \psi^{(0)}_{ibnd} \right\rangle \end{pmatrix} \tag{51}$$

(51) の行列積した後の列ベクトルにおいて、m 行目の成分は

$$\sum_{n=1}^{npw} V_{\mathrm{L}}(\boldsymbol{G}_m-\boldsymbol{G}_n) \left\langle \boldsymbol{k}+\boldsymbol{G}_n \middle| \psi^{(0)}_i \right\rangle \tag{52}$$

である。

運動エネルギーの場合と異なり、非対角成分に有限の要素が現れ、(51) のような積は直接に行おうとすると演算量が膨大である。そこで、高速フーリエ変換を活用し、下のコードに示すような流れで計算量を減らす工夫がなされている：

Listing 9: PW/src/vloc.f90

```
254   ! the local potential V_Loc psi. First bring psi to real space
255   !
256   DO ibnd = 1, m, incr
257      !
258      IF( dffts%have_task_groups ) THEN
259         !
260         tg_psic = (0.d0, 0.d0)
261         ioff = 0
262         !
263         DO idx = 1, dffts%nogrp
264
265            IF( idx + ibnd - 1 <= m ) THEN
266!$omp parallel do
267               DO j = 1, n
268                  tg_psic(nls (igk(j))+ioff) = psi(j,idx+ibnd-1)
269               ENDDO
270!$omp end parallel do
271            ENDIF
272
273            ioff = ioff + dffts%tg_nnr
274
275         ENDDO
276         !
277         CALL invfft ('Wave', tg_psic, dffts)
278         !
279      ELSE
280         !
281         psic(:) = (0.d0, 0.d0)
282         psic (nls (igk(1:n))) = psi(1:n, ibnd)
283         !
284         CALL invfft ('Wave', psic, dffts)
285         !
286      ENDIF
287      !
288      ! fft to real space
289      ! product with the potential v on the smooth grid
290      ! back to reciprocal space
291      !
292      IF( dffts%have_task_groups ) THEN
293         !
294!$omp parallel do
295         DO j = 1, dffts%nr1x*dffts%nr2x*dffts%tg_npp( me_bgrp + 1 )
296            tg_psic (j) = tg_psic (j) * tg_v(j)
297         ENDDO
298!$omp end parallel do
299         !
300         CALL fwfft ('Wave', tg_psic, dffts)
301         !
302      ELSE
303         !
304!$omp parallel do
305         DO j = 1, dffts%nnr
```

```
306           psic (j) = psic (j) * v(j)
307         ENDDO
308 !$omp end parallel do
309         !
310         CALL fwfft ('Wave', psic, dffts)
311         !
312       ENDIF
313       !
314       ! addition to the total product
315       !
316       IF( dffts%have_task_groups ) THEN
317         !
318         ioff = 0
319         !
320         DO idx = 1, dffts%nogrp
321           !
322           IF( idx + ibnd - 1 <= m ) THEN
323 !$omp parallel do
324             DO j = 1, n
325               hpsi (j, ibnd+idx-1) = hpsi (j, ibnd+idx-1) + tg_psic( nls(igk(j))
                      + ioff )
326             ENDDO
327 !$omp end parallel do
328           ENDIF
329           !
330           ioff = ioff + dffts%nr3x * dffts%nsw( me_bgrp + 1 )
331           !
332         ENDDO
333         !
334       ELSE
335 !$omp parallel do
336         DO j = 1, n
337           hpsi (j, ibnd) = hpsi (j, ibnd) + psic (nls(igk(j)))
338         ENDDO
339 !$omp end parallel do
340       ENDIF
341       !
342     ENDDO
343     !
344     IF( dffts%have_task_groups ) THEN
345       !
346       DEALLOCATE( tg_psic )
347       DEALLOCATE( tg_v )
348       !
349     ENDIF
350     dffts%have_task_groups = use_tg
351     !
352     RETURN
353 END SUBROUTINE vloc_psi_k
354 !
```

ここで行っている計算の手順を理解するには、次に示す式変形を見る必要がある。

$$
\begin{aligned}
V_{\mathrm{L}}(\boldsymbol{r})\psi_{i\boldsymbol{k}}(\boldsymbol{r}) &= \sum_{\boldsymbol{G}} V_{\mathrm{L}}(\boldsymbol{G})e^{i\boldsymbol{G}\cdot\boldsymbol{r}} \times \frac{1}{\sqrt{\Omega_{\mathrm{cell}}}} \sum_{\boldsymbol{G'}} C^{i}_{\boldsymbol{k}+\boldsymbol{G'}} e^{i(\boldsymbol{k}+\boldsymbol{G'})\cdot\boldsymbol{r}} \\
&= \sum_{\boldsymbol{G}} \Big(\sum_{\boldsymbol{G'}} V_{\mathrm{L}}(\boldsymbol{G}) \frac{1}{\sqrt{\Omega_{\mathrm{cell}}}} C^{i}_{\boldsymbol{k}+\boldsymbol{G'}} e^{i\boldsymbol{G'}\cdot\boldsymbol{r}} \Big) e^{i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}} \\
&= \sum_{\boldsymbol{G}} \Big(\sum_{\boldsymbol{G'}} \frac{1}{\sqrt{\Omega_{\mathrm{cell}}}} V_{\mathrm{L}}(\boldsymbol{G}-\boldsymbol{G'}) C^{i}_{\boldsymbol{k}+\boldsymbol{G'}} \Big) e^{i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}}
\end{aligned}
\tag{53}
$$

最後の式の括弧に現れたのはまさに (51) の各要素 (52) であり、最左辺の $\boldsymbol{G}$ によるフーリエ展開係数とみなすことができる。これより

$$
\sum_{\boldsymbol{G'}} V_{\mathrm{L}}(\boldsymbol{G}-\boldsymbol{G'}) C^{i}_{\boldsymbol{k}+\boldsymbol{G'}} = \frac{1}{\Omega_{\mathrm{cell}}} \int_{\Omega_{\mathrm{cell}}} d\boldsymbol{r} \Big(\sqrt{\Omega_{\mathrm{cell}}} V_{\mathrm{L}}(\boldsymbol{r}) \psi_{i\boldsymbol{k}}(\boldsymbol{r}) \Big) e^{-i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}}
\tag{54}
$$

となる。つまり、(51) の各要素 (52) は、局所ポテンシャルと波動関数の積を実空間で計算したものを FFT によって得られることを意味している。$V_{\mathrm{L}}(\boldsymbol{r})$ と $\psi_{i\boldsymbol{k}}(\boldsymbol{r})$ はそれぞれ、逆 FFT で得られる。

上述の手順が記述されたソースコードを確認しよう。簡単のため、FFT の並列化で参照しない方を見ていく。162〜342 行目のバンドに関する DO ループの内側で処理が行われる。282 行目で、逆空間表示の固有ベクトルの配列 psi がローカルの配列 psic に代入され、284 行目で逆 FFT により実空間表示の固有ベクトルが psic として得られる。304〜308 行目で実空間での局所ポテンシャルと固有ベクトル（波動関数）の積が計算される。これが 310 行目で FFT により逆空間に移される。335〜339 行目で、運動エネルギー項と局所ポテンシャル項の和がとられ、hpsi に代入される。hpsi はもともと運動エネルギー項のみ持っていたが、ここで局所ポテンシャル項との和として上書きされる。

ここまでが、ハミルトニアンと波動関数の積の演算の内容である。では、繰り返し対角化の演算の話に戻ろう。

9.2 繰り返し対角化のループ

Listing 10: PW/src/cegterg.f90

```
160   !
161   ! ... hc contains the projection of the hamiltonian onto the reduced
162   ! ... space vc contains the eigenvectors of hc
163   !
164   hc(:,:) = ZERO
165   sc(:,:) = ZERO
166   vc(:,:) = ZERO
167   !
168   CALL ZGEMM( 'C', 'N', nbase, nbase, kdim, ONE, &
169               psi, kdmx, hpsi, kdmx, ZERO, hc, nvecx )
170   !
171   CALL mp_sum( hc( :, 1:nbase ), intra_bgrp_comm )
172   !
173   IF ( uspp ) THEN
174      !
175      CALL ZGEMM( 'C', 'N', nbase, nbase, kdim, ONE, &
176                  psi, kdmx, spsi, kdmx, ZERO, sc, nvecx )
177      !
178   ELSE
```

```
179     !
180     CALL ZGEMM( 'C', 'N', nbase, nbase, kdim, ONE, &
181                 psi, kdmx, psi, kdmx, ZERO, sc, nvecx )
182     !
183  END IF
184  !
185  CALL mp_sum( sc( :, 1:nbase ), intra_bgrp_comm )
186  !
187  IF ( lrot ) THEN
188     !
189     DO n = 1, nbase
190        !
191        e(n) = REAL( hc(n,n) )
192        !
193        vc(n,n) = ONE
194        !
195     END DO
196     !
197  ELSE
198     !
199     ! ... diagonalize the reduced hamiltonian
200     !
201     CALL cdiaghg( nbase, nvec, hc, sc, nvecx, ew, vc )
202     !
203     e(1:nvec) = ew(1:nvec)
204     !
205  END IF
```

168、175、180 行目で呼び出される ZGEMM において、hpsi と spsi に代入された $H\left|\psi_i^{(0)}\right\rangle$ と $S\left|\psi_i^{(0)}\right\rangle$ にブラ $\left\langle\psi_i^{(0)}\right|$ を掛けることでハミルトニアンと重なり積分の行列要素が計算され、配列 hc と sc に代入される。ZGEMM というのは、LAPACK(Linear Algebra PACKage) と呼ばれる線形代数の演算のパッケージにおけるプログラムの一つであり、複素行列の積を計算する。[5] ここでは、行列 (49) と $\left|\psi_i^{(0)}\right\rangle$ に共役なベクトルを入力として、以下の行列

$$\begin{pmatrix} \left\langle\psi_1^{(0)}|H|\psi_1^{(0)}\right\rangle & \cdots & \left\langle\psi_1^{(0)}|H|\psi_{nbase}^{(0)}\right\rangle \\ \vdots & & \vdots \\ \left\langle\psi_{nbase}^{(0)}|H|\psi_1^{(0)}\right\rangle & \cdots & \left\langle\psi_{nbase}^{(0)}|H|\psi_{nbase}^{(0)}\right\rangle \end{pmatrix} \tag{55}$$

を返す（S の方も同様）。

189〜205 行目で初期の近似的な固有値と固有ベクトルが決定される。lrot が真であれば、試行ベクトル $\left|\psi_i^{(0)}\right\rangle$ により

$$\epsilon_i^{(0)} = \left\langle\psi_i^{(0)}\right|H\left|\psi_i^{(0)}\right\rangle \tag{56}$$

が固有値として配列 e(n) に代入され、固有ベクトルは単位行列として配列 vc に代入される。lrot が偽であれば、201 行目でコールされる cdiaghg にて行列 (55) を対角化し、得られた固有値 ew が e(n) に代入される。

[5] 具体的な計算内容は、`https://netlib.org/lapack/explore-html-3.6.1/dc/d17/` のサイトから、Modules → Level3 complex16 にアクセスして確認してほしい。

Listing 11: PW/src/cegterg.f90

```
206 !
207  ! ... iterate
208  !
209  iterate: DO kter = 1, maxter
210     !
211     dav_iter = kter
212     !
213     CALL start_clock( 'cegterg:update' )
214     !
215     np = 0
216     !
217     DO n = 1, nvec
218        !
219        IF ( .NOT. conv(n) ) THEN
220           !
221           ! ... this root not yet converged ...
222           !
223           np = np + 1
224           !
225           ! ... reorder eigenvectors so that coefficients for unconverged
226           ! ... roots come first. This allows to use quick matrix-matrix
227           ! ... multiplications to set a new basis vector (see below)
228           !
229           IF ( np /= n ) vc(:,np) = vc(:,n)
230           !
231           ! ... for use in g_psi
232           !
233           ew(nbase+np) = e(n)
234           !
235        END IF
236        !
237     END DO
```

217〜237 行目では、前回のイタレーションで得られた固有ベクトル vc(:,:)[6]について、固有値が収束しなかった固有ベクトルを左から並べ、収束した固有ベクトルがその後に並ぶように処理している。収束しなかった固有値の数 notconv は、イタレーションごとに 368 行で計算される。

Listing 12: PW/src/cegterg.f90

```
238     !
239     nb1 = nbase + 1
240     !
241     ! ... expand the basis set with new basis vectors ( H - e*S )|psi> ...
242     !
243     IF ( uspp ) THEN
244        !
245        CALL ZGEMM( 'N', 'N', kdim, notcnv, nbase, ONE, spsi, &
246                    kdmx, vc, nvecx, ZERO, psi(1,1,nb1), kdmx )
```

[6]npw 個の要素を持つ列ベクトルが左からずらっと nbase 個並んでいる。

```
247        !
248     ELSE
249        !
250        CALL ZGEMM( 'N', 'N', kdim, notcnv, nbase, ONE, psi, &
251                    kdmx, vc, nvecx, ZERO, psi(1,1,nb1), kdmx )
252        !
253     END IF
254     !
255     DO np = 1, notcnv
256        !
257        psi(:,:,nbase+np) = - ew(nbase+np)*psi(:,:,nbase+np)
258        !
259     END DO
260     !
261     CALL ZGEMM( 'N', 'N', kdim, notcnv, nbase, ONE, hpsi, &
262                 kdmx, vc, nvecx, ONE, psi(1,1,nb1), kdmx )
263     !
264     CALL stop_clock( 'cegterg:update' )
```

$kter$ 回目のイタレーションにおいて、一つ前までのイタレーションで、notcnv 個の固有ベクトル $\left|\psi_i^{(kter-1)}\right\rangle$ が配列 vc に格納されている。これらの固有ベクトルに行列 $(H-eS)$ を掛けることで notcnv 個の新たに追加する規定ベクトルを作る。

詳細には、245 行目で $S\left|\psi_i\right\rangle$ が計算され、255〜257 行目でこれと近似固有値の積 $-eS\left|\psi_i\right\rangle$ が計算される。この結果は配列 psi(1,1,nb1) に格納される（nb1=nbase+1）。261 行目の ZGEMM では $-e_iS\left|\psi_i\right\rangle$ に $H\left|\psi_i\right\rangle$ を加えた結果が返される。ここまでで、

$$(H-e_iS)\left|\psi_i^{(kter-1)}\right\rangle \qquad (i=nbase+1,\cdots,nbase+notcnv) \tag{57}$$

が得られたことになる。

Listing 13: PW/src/cegterg.f90

```
265     !
266     ! ... approximate inverse iteration
267     !
268     CALL g_psi( npwx, npw, notcnv, npol, psi(1,1,nb1), ew(nb1) )
269     !
```

ここでは、(57) に行列

$$\langle \boldsymbol{k}+\boldsymbol{G}|D|\boldsymbol{k}+\boldsymbol{G'}\rangle = \frac{\delta_{\boldsymbol{GG'}}}{\langle \boldsymbol{k}+\boldsymbol{G}|H-\epsilon S|\boldsymbol{k}+\boldsymbol{G}\rangle} \tag{58}$$

を掛けて psi を上書きする。この行列は、$H-\epsilon S$ の逆行列の近似である。

$$\left|\delta\psi_i^{(kter-1)}\right\rangle = D(H-\epsilon_iS)\left|\psi_i^{(kter-1)}\right\rangle \tag{59}$$

Listing 14: PW/src/cegterg.f90

```
270     ! ... "normalize" correction vectors psi(:,nb1:nbase+notcnv) in
271     ! ... order to improve numerical stability of subspace diagonalization
272     ! ... (cdiaghg) ew is used as work array :
273     !
274     ! ... ew = <psi_i|psi_i>, i = nbase + 1, nbase + notcnv
275     !
276     DO n = 1, notcnv
277        !
278        nbn = nbase + n
279        !
280        IF ( npol == 1 ) THEN
281           !
282           ew(n) = ddot( 2*npw, psi(1,1,nbn), 1, psi(1,1,nbn), 1 )
283           !
284        ELSE
285           !
286           ew(n) = ddot( 2*npw, psi(1,1,nbn), 1, psi(1,1,nbn), 1 ) + &
287                   ddot( 2*npw, psi(1,2,nbn), 1, psi(1,2,nbn), 1 )
288           !
289        END IF
290        !
291     END DO
292     !
293     CALL mp_sum( ew( 1:notcnv ), intra_bgrp_comm )
294     !
295     DO n = 1, notcnv
296        !
297        psi(:,:,nbase+n) = psi(:,:,nbase+n) / SQRT( ew(n) )
298        !
299     END DO
```

276〜297 行目では、新たに追加された基底ベクトルの規格化が行われている。以上より、

$$\left|\tilde{\psi}_i^{(kter)}\right\rangle = \left|\tilde{\psi}_i^{(kter-1)}\right\rangle, \qquad \left|\tilde{\psi}_{i+notcnv}^{(kter)}\right\rangle = \left|\delta\psi_i^{(kter-1)}\right\rangle \tag{60}$$

とし、部分空間を拡張して

$$\mathcal{K} = \left\{ \left|\tilde{\psi}_1^{(kter)}\right\rangle, \cdots, \left|\tilde{\psi}_{nbase}^{(kter)}\right\rangle, \left|\tilde{\psi}_{nbase+1}^{(kter)}\right\rangle, \cdots, \left|\tilde{\psi}_{nbase+notcnv}^{(kter)}\right\rangle \right\} \tag{61}$$

を得る。

Listing 15: PW/src/cegterg.f90

```
300     !
301     ! ... here compute the hpsi and spsi of the new functions
302     !
303     !
304     CALL h_psi( npwx, npw, notcnv, psi(1,1,nb1), hpsi(1,1,nb1) )
305     !
306     IF ( uspp ) &
```

```
307         CALL s_psi( npwx, npw, notcnv, psi(1,1,nb1), spsi(1,1,nb1) )
308      !
309      ! ... update the reduced hamiltonian
310      !
311      CALL start_clock( 'cegterg:overlap' )
312      !
313      CALL ZGEMM( 'C', 'N', nbase+notcnv, notcnv, kdim, ONE, psi, &
314                  kdmx, hpsi(1,1,nb1), kdmx, ZERO, hc(1,nb1), nvecx )
315      !
316      CALL mp_sum( hc( :, nb1:nb1+notcnv-1 ), intra_bgrp_comm )
317      !
318      IF ( uspp ) THEN
319         !
320         CALL ZGEMM( 'C', 'N', nbase+notcnv, notcnv, kdim, ONE, psi, &
321                     kdmx, spsi(1,1,nb1), kdmx, ZERO, sc(1,nb1), nvecx )
322         !
323      ELSE
324         !
325         CALL ZGEMM( 'C', 'N', nbase+notcnv, notcnv, kdim, ONE, psi, &
326                     kdmx, psi(1,1,nb1), kdmx, ZERO, sc(1,nb1), nvecx )
327         !
328      END IF
329      !
330      CALL mp_sum( sc( :, nb1:nb1+notcnv-1 ), intra_bgrp_comm )
331      !
332      CALL stop_clock( 'cegterg:overlap' )
333      !
334      nbase = nbase + notcnv
335      !
336      DO n = 1, nbase
337         !
338         ! ... the diagonal of hc and sc must be strictly real
339         !
340         hc(n,n) = CMPLX( REAL( hc(n,n) ), 0.D0 ,kind=DP)
341         sc(n,n) = CMPLX( REAL( sc(n,n) ), 0.D0 ,kind=DP)
342         !
343         DO m = n + 1, nbase
344            !
345            hc(m,n) = CONJG( hc(n,m) )
346            sc(m,n) = CONJG( sc(n,m) )
347            !
348         END DO
349         !
350      END DO
351      !
352      ! ... diagonalize the reduced hamiltonian
353      !
354      CALL cdiaghg( nbase, nvec, hc, sc, nvecx, ew, vc )
355      !
```

304 行目で、新たに追加された基底ベクトル群

$$\left\{ \left| \tilde{\psi}^{(kter)}_{nbase+1} \right\rangle, \cdots, \left| \tilde{\psi}^{(kter)}_{nbase+notcnv} \right\rangle \right\} \tag{62}$$

に対してハミルトニアンを掛ける（S も同様）。その結果を入力として、313 行目の ZGEMM で以下のような行列積が計算される：

$$
\begin{aligned}
&\begin{pmatrix}
\left\langle \tilde{\psi}_1^{(kter)} | \boldsymbol{k}+\boldsymbol{G}_1 \right\rangle & \cdots & \left\langle \tilde{\psi}_1^{(kter)} | \boldsymbol{k}+\boldsymbol{G}_{npw} \right\rangle \\
\vdots & & \vdots \\
\left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} | \boldsymbol{k}+\boldsymbol{G}_1 \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} | \boldsymbol{k}+\boldsymbol{G}_{npw} \right\rangle
\end{pmatrix} \\
&\times \begin{pmatrix}
\left\langle \boldsymbol{k}+\boldsymbol{G}_1 | H | \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \boldsymbol{k}+\boldsymbol{G}_1 | H | \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\vdots & & \vdots \\
\left\langle \boldsymbol{k}+\boldsymbol{G}_{npw} | H | \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \boldsymbol{k}+\boldsymbol{G}_{npw} | H | \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle
\end{pmatrix} \\
&= \begin{bmatrix}
\left\langle \tilde{\psi}_1^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_1^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\vdots & & \vdots \\
\left\langle \tilde{\psi}_{nbase}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase}^{(0)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\left\langle \tilde{\psi}_{nbase+1}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase+1}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\vdots & & \vdots \\
\left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle
\end{bmatrix}
\end{aligned}
\tag{63}
$$

結果は配列 hc(1,1,nb1) に代入される。ここまでで、配列 hc の全体は次のような行列を表すものとなっている：

$$
\begin{bmatrix}
\left\langle \tilde{\psi}_1^{(kter-1)} \middle| H \middle| \tilde{\psi}_1^{(kter-1)} \right\rangle & \cdots & \left\langle \tilde{\psi}_1^{(kter-1)} \middle| H \middle| \tilde{\psi}_{nbase}^{(kter-1)} \right\rangle & \left\langle \tilde{\psi}_1^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_1^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\vdots & & \vdots & \vdots & & \vdots \\
\left\langle \tilde{\psi}_{nbase}^{(kter-1)} \middle| H \middle| \tilde{\psi}_1^{(kter-1)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase}^{(kter-1)} \middle| H \middle| \tilde{\psi}_{nbase}^{(kter-1)} \right\rangle & \left\langle \tilde{\psi}_{nbase}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
0 & \cdots & 0 & \left\langle \tilde{\psi}_{nbase+1}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase+1}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle \\
\vdots & & \vdots & & & \\
0 & \cdots & 0 & \left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+1}^{(kter)} \right\rangle & \cdots & \left\langle \tilde{\psi}_{nbase+notcnv}^{(kter)} \middle| H \middle| \tilde{\psi}_{nbase+notcnv}^{(kter)} \right\rangle
\end{bmatrix}
\tag{64}
$$

334 行で nbase+notcnv を nbase として更新する。336〜350 行目では、hc の対角成分が実数になり、かつエルミート行列になるように処理している。354 行目で、この拡大された行列に対し、対角化計算を行う。

358 行目以降は、固有値が収束しているかどうかで、収束判定がなされる。

Listing 16: PW/src/cegterg.f90

```
356 ! ... test for convergence
357     !
358     WHERE( btype(1:nvec) == 1 )
359        !
360        conv(1:nvec) = ( ( ABS( ew(1:nvec) - e(1:nvec) ) < ethr ) )
361        !
362     ELSEWHERE
363        !
364        conv(1:nvec) = ( ( ABS( ew(1:nvec) - e(1:nvec) ) < empty_ethr ) )
365        !
366     END WHERE
367     !
368     notcnv = COUNT( .NOT. conv(:) )
369     !
```

```
    e(1:nvec) = ew(1:nvec)
    !
    ! ... if overall convergence has been achieved, or the dimension of
    ! ... the reduced basis set is becoming too large, or in any case if
    ! ... we are at the last iteration refresh the basis set. i.e. replace
    ! ... the first nvec elements with the current estimate of the
    ! ... eigenvectors; set the basis dimension to nvec.
    !
    IF ( notcnv == 0 .OR. &
         nbase+notcnv > nvecx .OR. dav_iter == maxter ) THEN
       !
       CALL start_clock( 'cegterg:last' )
       !
       CALL ZGEMM( 'N', 'N', kdim, nvec, nbase, ONE, &
                   psi, kdmx, vc, nvecx, ZERO, evc, kdmx )
       !
       IF ( notcnv == 0 ) THEN
          !
          ! ... all roots converged: return
          !
          CALL stop_clock( 'cegterg:last' )
          !
          EXIT iterate
          !
       ELSE IF ( dav_iter == maxter ) THEN
          !
          ! ... last iteration, some roots not converged: return
          !
          !!!WRITE( stdout, '(5X,"WARNING: ",I5, &
          !!! & " eigenvalues not converged")' ) notcnv
          !
          CALL stop_clock( 'cegterg:last' )
          !
          EXIT iterate
          !
       END IF
       !
       ! ... refresh psi, H*psi and S*psi
       !
       psi(:,:,1:nvec) = evc(:,:,1:nvec)
       !
       IF ( uspp ) THEN
          !
          CALL ZGEMM( 'N', 'N', kdim, nvec, nbase, ONE, spsi, &
                      kdmx, vc, nvecx, ZERO, psi(1,1,nvec+1), kdmx )
          !
          spsi(:,:,1:nvec) = psi(:,:,nvec+1:nvec+nvec)
          !
       END IF
       !
       CALL ZGEMM( 'N', 'N', kdim, nvec, nbase, ONE, hpsi, &
                   kdmx, vc, nvecx, ZERO, psi(1,1,nvec+1), kdmx )
       !
```

```
423         hpsi(:,:,1:nvec) = psi(:,:,nvec+1:nvec+nvec)
424         !
425         ! ... refresh the reduced hamiltonian
426         !
427         nbase = nvec
428         !
429         hc(:,1:nbase) = ZERO
430         sc(:,1:nbase) = ZERO
431         vc(:,1:nbase) = ZERO
432         !
433         DO n = 1, nbase
434            !
435 ! hc(n,n) = REAL( e(n) )
436            hc(n,n) = CMPLX( e(n), 0.0_DP ,kind=DP)
437            !
438            sc(n,n) = ONE
439            vc(n,n) = ONE
440            !
441         END DO
442         !
443         CALL stop_clock( 'cegterg:last' )
444         !
445      END IF
446      !
447   END DO iterate
```

全ての固有値が収束条件を満たせば、392 行でダビッドソンループから抜ける。イタレーション回数が最大回数に達したら、403 行でループから抜ける。もし拡張されたハミルトニアンの次元が大きくなりすぎたら（nbase+notcnv ¿ nvecx）、基底の組のうちはじめから nvec 番目のベクトルを別のものと入れ替える（383 行と 409 行を参照）。後続する部分で、hpsi、spsi、hc、sc、vc も新たに組み直されている。

10　電子密度の計算（sum_bands.f90）

SCF 計算では、各ループにおいて固有状態が求まると、それを入力として電子密度が計算される。ここで、上の式からわかるように、$n(\boldsymbol{r})$ の計算には、N_G^2 回程度の積の演算を伴う。これは、高速フーリエ変換（Fast Fourier Transform; FFT）を上手く活用することで回避できる。

c_bands() が回ったのち、electron_scf() に戻り、その 502 行目で電子密度が計算される:

Listing 17: PW/src/electron.f90

```
497         !
498         ! ... the new density is computed here. For PAW:
499         ! ... sum_band computes new becsum (stored in uspp modules)
500         ! ... and a subtly different copy in rho%bec (scf module)
501         !
502         CALL sum_band()
503         !
```

sum_band.f90 のソースコードを見てほしい。前半の sum_band_gamma() は、k メッシュの指定がガンマ点のみの場合に特化したバージョンのものであり、ユーザー側からすると入力ファイルで K_POINTS{gamma} と指定することで動く。ここでは、より一般的な方である k 点を複数とった場合のバージョンである、sum_band_k() の方を見る。

sum_band_k() では、与えられた逆空間での固有ベクトルを逆 FFT により実空間に変換し、以下の式で電子密度を計算する：

$$n(\boldsymbol{r}) = \sum_{i}^{\mathrm{occ}} \sum_{\boldsymbol{k}}^{\mathrm{IBZ}} w_{i\boldsymbol{k}} |u_{i\boldsymbol{k}}(\boldsymbol{r})|^2 \tag{65}$$

ここで、$w_{i\boldsymbol{k}}$ は i 番目のバンドで IBZ 内の $\boldsymbol{k}$ 点における重み（weight）である。$u_{i\boldsymbol{k}}(\boldsymbol{r})$ は、c_band() で計算された固有状態 $\{C^i_{\boldsymbol{k}+\boldsymbol{G}}\}$ から逆 FFT により得られる、格子周期関数である。sum_band_k() において実際に (65) を計算している箇所は以下である：

Listing 18: PW/src/sum_band.f90

```
747                !
748                IF( dffts%have_task_groups ) THEN
749                   !
750 !$omp parallel default(shared), private(j,ioff,idx)
751 !$omp do
752                   DO j = 1, SIZE( tg_psi )
753                      tg_psi(j) = ( 0.D0, 0.D0 )
754                   END DO
755 !$omp end do
756                   !
757                   ioff = 0
758                   !
759                   DO idx = 1, dffts%nogrp
760                      !
761                      ! ... dffts%nogrp ffts at the same time
762                      !
763                      IF( idx + ibnd - 1 <= nbnd ) THEN
764 !$omp do
765                         DO j = 1, npw
766                            tg_psi( nls( igk( j ) ) + ioff ) = evc( j, idx+ibnd-1 )
767                         END DO
768 !$omp end do
769                      END IF
770
771                      ioff = ioff + dffts%tg_nnr
772
773                   END DO
774 !$omp end parallel
775                   !
776                   CALL invfft ('Wave', tg_psi, dffts)
777                   !
778                   ! Now the first proc of the group holds the first band
779                   ! of the dffts%nogrp bands that we are processing at the same
                         time,
780                   ! the second proc. holds the second and so on
781                   !
```

```
782                   ! Compute the proper factor for each band
783                   !
784                   DO idx = 1, dffts%nogrp
785                      IF( dffts%nolist( idx ) == me_bgrp ) EXIT
786                   END DO
787                   !
788                   ! Remember
789                   ! proc 0 has bands ibnd
790                   ! proc 1 has bands ibnd+1
791                   ! ....
792                   !
793                   IF( idx + ibnd - 1 <= nbnd ) THEN
794                      w1 = wg( idx + ibnd - 1, ik) / omega
795                   ELSE
796                      w1 = 0.0d0
797                   END IF
798                   !
799                   CALL get_rho(tg_rho, dffts%tg_npp( me_bgrp + 1 ) * dffts%nr1x *
                          dffts%nr2x, w1, tg_psi)
800                   !
801                ELSE
802                   !
803                   psic(:) = ( 0.D0, 0.D0 )
804                   !
805                   psic(nls(igk(1:npw))) = evc(1:npw,ibnd)
806                   !
807                   CALL invfft ('Wave', psic, dffts)
808                   !
809                   ! ... increment the charge density ...
810                   !
811                   CALL get_rho(rho%of_r(:,current_spin), dffts%nnr, w1, psic)
812
813                END IF
814                !
```

この箇所は、$\boldsymbol{k}$ に関する Do ループ（ik=1,nks）の内側のバンドに関する Do ループ（ibnd=1,nbnd,incr）の内側に記述されている。ノンコリニアを計算するかしないかについての IF 分（661 行）の、計算しない方の箇所に該当する。659 行において、重み $w_{i\boldsymbol{k}}$ が計算され、変数 w1 = wg(ibnd,ik) / omega に格納される。omega はユニットセルの体積である[7]。

FFT に関する並列のオプション（-nt）をユーザーが指定すれば 748 行目を参照し、そうでない場合は 801 行目を参照する。単純のため、ここでは並列でない方を見る。

805 行目で、c_bands() で計算された逆空間表示の固有ベクトルが格納されている配列 evc を、配列 psic に代入する（evc と psic は、Module/wavefunctions.f90 で定義されている）。これを 807 行でコールする invfft() で逆 FFT を行い[8]、実空間に変換する。得られた実空間表示の固有ベクトルを用いて、811 行目の get_rho() で、ノルムの 2 乗の計算がなされる。get_rho() を以下に示す：

[7]セルに関するパラメータ群は Modules/cell_base.f90 で定義されている。
[8]FFT に関するサブルーチンは、Modules/fft_interfaces.f90 に記述されている。

Listing 19: PW/src/sum_band.f90

```
560     SUBROUTINE get_rho(rho_loc, nrxxs_loc, w1_loc, psic_loc)
561
562        IMPLICIT NONE
563
564        INTEGER :: nrxxs_loc
565        REAL(DP) :: rho_loc(nrxxs_loc)
566        REAL(DP) :: w1_loc
567        COMPLEX(DP) :: psic_loc(nrxxs_loc)
568
569        INTEGER :: ir
570
571 !$omp parallel do
572        DO ir = 1, nrxxs_loc
573           !
574           rho_loc(ir) = rho_loc(ir) + &
575                         w1_loc * ( DBLE( psic_loc(ir) )**2 + &
576                                    AIMAG( psic_loc(ir) )**2 )
577           !
578        END DO
579 !$omp end parallel do
580
581     END SUBROUTINE get_rho
```

11　有効ポテンシャルの更新（v_of_rho.f90）

電子密度が収束されたと判断されていなければ、mix_rho() で更新された電子密度を用いて、KS 方程式の有効ポテンシャル $v_{\rm eff}$ が作られる。v_of_rho.f90 では、$v_{\rm eff}$ のうち、交換相関項 $v_{\rm xc}$ とハートレー項 $v_{\rm H}$ が計算される。$v_{\rm xc}$ は実空間表示の電子密度を用いて計算され、

$$v_{\rm xc}(\boldsymbol{r}) = \varepsilon_{\rm xc}(\boldsymbol{r}) + \frac{d\varepsilon_{\rm xc}}{d\rho} \tag{66}$$

である。$\varepsilon_{\rm xc}(\boldsymbol{r})$ は交換相関エネルギー密度であり、交換項と相関項の和として与えられる。

$$\varepsilon_{\rm xc} = \varepsilon_{\rm x} + \varepsilon_{\rm c} \tag{67}$$

$\varepsilon_{\rm xc}$ は、様々なものが用意されており、ユーザー側で指定する汎関数の種類によって変わる。

$v_{\rm xc}(\boldsymbol{r})$ を実際に計算している部分を見てみよう。

Listing 20: PW/src/v_of_rho.f90

```
379   etxc = 0.D0
380   vtxc = 0.D0
381   v(:,:) = 0.D0
382   rhoneg = 0.D0
383   !
384   IF ( nspin == 1 .OR. ( nspin == 4 .AND. .NOT. domag ) ) THEN
385      !
386      ! ... spin-unpolarized case
387      !
388 !$omp parallel do private( rhox, arhox, ex, ec, vx, vc ), &
389 !$omp reduction(+:etxc,vtxc), reduction(-:rhoneg)
```

```
390     DO ir = 1, dfftp%nnr
391        !
392        rhox = rho%of_r(ir,1) + rho_core(ir)
393        !
394        arhox = ABS( rhox )
395        !
396        IF ( arhox > vanishing_charge ) THEN
397           !
398           CALL xc( arhox, ex, ec, vx(1), vc(1) )
399           !
400           v(ir,1) = e2*( vx(1) + vc(1) )
401           !
402           etxc = etxc + e2*( ex + ec ) * rhox
403           !
404           vtxc = vtxc + v(ir,1) * rho%of_r(ir,1)
405           !
406        ENDIF
407        !
408        IF ( rho%of_r(ir,1) < 0.D0 ) rhoneg(1) = rhoneg(1) - rho%of_r(ir,1)
409        !
410     END DO
411 !$omp end parallel do
412     !
```

これは、v_of_rho.f90 内の一つ目のサブルーチン v_xc() の一部を示したものである。スピン偏極を考えない場合に該当する。ここでの DO ループは実空間メッシュ（ir）についてである。まず、擬ポテンシャルの内核からの寄与 rho_core と、価電子による寄与 rho%of_r の和が rho_x に代入される。398 行でコールされる xc() は Module/funct.f90 の 1162 行に記述されており、エネルギー密度について、交換項と相関項それぞれについて求める。400 行でそれらの和がとられ、配列 v に代入する。e2 は電荷素量の二乗である[9]。402 行では全エネルギーのうちの交換相関項を計算している：

$$E_{\mathrm{xc}}[n] = \int d\boldsymbol{r} \varepsilon_{\mathrm{xc}}(\boldsymbol{r}) n(\boldsymbol{r}) \tag{68}$$

こうして、有効ポテンシャルとエネルギーにおける交換相関項の実空間表示が求まった。一方、ハートレー項 v_{H} は逆空間表示の電子密度を用いて計算される。

$$v_{\mathrm{H}}(\boldsymbol{G}) = \frac{4\pi e^2 n(\boldsymbol{G})}{|\mathrm{G}|^2} \tag{69}$$

$v_{\mathrm{H}}(\boldsymbol{G})$ を計算しているのは、サブルーチン vx() のすぐ後にあるサブルーチン v_h() である。以下にその一部を示す：

Listing 21: PW/src/v_of_rho.f90

```
590 !$omp parallel do private( fac, rgtot_re, rgtot_im ), reduction(+:ehart)
591     DO ig = gstart, ngm
592        !
593        fac = 1.D0 / gg(ig)
594        !
```

[9] こうした定数は Modules/constanfs.f90 で定義されている。

```
595         rgtot_re = REAL( rhog(ig,1) )
596         rgtot_im = AIMAG( rhog(ig,1) )
597         !
598         IF ( nspin == 2 ) THEN
599            !
600            rgtot_re = rgtot_re + REAL( rhog(ig,2) )
601            rgtot_im = rgtot_im + AIMAG( rhog(ig,2) )
602            !
603         END IF
604         !
605         ehart = ehart + ( rgtot_re**2 + rgtot_im**2 ) * fac
606         !
607         aux1(1,ig) = rgtot_re * fac
608         aux1(2,ig) = rgtot_im * fac
609         !
610      ENDDO
611 !$omp end parallel do
612      !
613      fac = e2 * fpi / tpiba2
614      !
615      ehart = ehart * fac
616      !
617      aux1 = aux1 * fac
618      !
619      IF ( gamma_only ) THEN
620         !
621         ehart = ehart * omega
622         !
623      ELSE
624         !
625         ehart = ehart * 0.5D0 * omega
626         !
627      END IF
628      !
629      if (do_comp_mt) then
630         ALLOCATE( vaux( ngm ), rgtot(ngm) )
631         rgtot(:) = rhog(:,1)
632         if (nspin==2) rgtot(:) = rgtot(:) + rhog(:,2)
633         CALL wg_corr_h (omega, ngm, rgtot, vaux, eh_corr)
634         aux1(1,1:ngm) = aux1(1,1:ngm) + REAL( vaux(1:ngm))
635         aux1(2,1:ngm) = aux1(2,1:ngm) + AIMAG(vaux(1:ngm))
636         ehart = ehart + eh_corr
637         DEALLOCATE( rgtot, vaux )
638      end if
639      !
640      CALL mp_sum( ehart , intra_bgrp_comm )
641      !
642      aux(:) = 0.D0
643      !
644      aux(nl(1:ngm)) = CMPLX ( aux1(1,1:ngm), aux1(2,1:ngm), KIND=dp )
645      !
646      IF ( gamma_only ) THEN
647         !
```

```
648         aux(nlm(1:ngm)) = CMPLX ( aux1(1,1:ngm), -aux1(2,1:ngm), KIND=dp )
649         !
650      END IF
651   END IF
652   !
653   ! ... transform hartree potential to real space
654   !
655   CALL invfft ('Dense', aux, dfftp)
656   !
657   ! ... add hartree potential to the xc potential
658   !
659   IF ( nspin == 4 ) THEN
660      !
661      v(:,1) = v(:,1) + DBLE (aux(:))
662      !
663   ELSE
664      !
665      DO is = 1, nspin
666         !
667         v(:,is) = v(:,is) + DBLE (aux(:))
668         !
669      END DO
670      !
671   END IF
672   !
673   DEALLOCATE( aux, aux1 )
674   !
675   CALL stop_clock( 'v_h' )
676   !
677   RETURN
678   !
679 END SUBROUTINE v_h
```

このサブルーチンは、逆空間での電子密度 $n(\boldsymbol{G})$ の配列 rhog(ngm,nspin) を受け取り、有効ポテンシャルと全エネルギーのハートレー項を計算して返すものである。ここでの DO ループは逆格子点 $\boldsymbol{G}$ に関するものであり、ループの各イタレーションにおいて、各 $\boldsymbol{G}$ 点での計算がされ、(69) の $\boldsymbol{G}$ の和をとっていく。

(69) の計算手順を具体的に説明すると次のようになる。593 行目で $|\boldsymbol{G}|^{-2}$ が fac に代入され、595 603 行で $n(\boldsymbol{G})$ の実部と虚部が、それぞれ rgtot_re と rgtot_im に代入される。607 と 608 行で $n(\boldsymbol{G})$ の実部、虚部と fac($|\boldsymbol{G}|^{-2}$) との積が計算される。DO 文が終わった時点で、各 $\boldsymbol{G}$ 点での $n(\boldsymbol{G})/|\boldsymbol{G}|^2$ が求まったことになる。613 行で変数 fac が $4\pi e^2$ として上書きされ（tpiba2 は $(2\pi/a)^2$）、617 行で係数 $4\pi e^2$ が掛け算される。fpi は 4π を意味し、Modules/constants.f90 で定義されている。ここまでで、(69) が計算されたことになる。

有効ポテンシャルだけでなく、ここでは全エネルギーのハートレー項も計算している。その値は ehart に代入される。

$$E_{\rm H} = \frac{1}{2} 4\pi e^2 \Omega_{\rm cell} \sum_{\boldsymbol{G}\neq 0} \frac{4\pi e^2 n(\boldsymbol{G})}{|{\rm G}|^2} \tag{70}$$

DO 文内の 605 行にて、$n(\boldsymbol{G})/|\boldsymbol{G}|^2$ の和がとられ、615 行と 619 627 行で係数が掛け算され、(70) が計算されたことになる。

655 行で、逆 FFT により波数空間の $v_{\rm H}$ を実空間に変換する。659 671 行で $v_{\rm H}$ と $v_{\rm xc}$ との和を、配列 v

に代入する。ポテンシャルを実空間に変換したのは、固有状態計算のところでハミルトニアンと波動関数の積を計算する際に、高速フーリエ変換を活用するためである。その詳細はサブルーチン h_psi() の説明で触れる。

第 IV 部
MPI 並列の実装について

12　既約ブリルアンゾーン内の k 点の並列化

13　高速フーリエ変換の並列化

14　繰り返し対角化計算の並列化