ESopt Manual

ver 1.2.1.E

August 2011 (English version)

Division of Frontier Materials Science,
Department of Materials Engineering Science,
Graduate School of Engineering Science,
Osaka University

ESopt is a program package for the electronic structure calculation based on the density functional theory (DFT). The program adopting the plane-wave expansion method is originated from the package developed in the Institute for Solid State Physics (ISSP), University of Tokyo, which is called "opt". This open software is provided as "ISSP FPSOPT" by ISSP. Our ESopt, which is a revised version of opt, was produced by choosing functions selectively and by applying the Fortran 95 scheme. We also added another diagonalizing routine to the original version.

The original "opt" has two characteristic features.

- High readability of the source code
- Realization of a CG (conjugate-gradient method) to optimize all degrees of freedom of the wavefunction.

The former advantage is succeeded in ESopt. To solve the memory allocation problem by Fortan 77, some routines became lengthy and hard for easy reading in the original program. This point is avoided using Fortran 90. The latter specialty was developed by Dr. Tadashi Ogitsu, who is now in the Lawrence Livermore National Laboratory (LLNL), US. Compared to the ordinal CG method using the band-by-band minimization in the band structure calculation, the method by Ogitsu perform the simultaneous optimization of all degrees of freedom for the Kohn-Sham wavefunction. Thus, "opt" is a special program, whose implementation realizes the original idea of the CG method as it is.

This manual is provided in order to achieve the next purpose. We give the basic equations and the real implementation of the steps in this manual. By comparing the mathematical representation and the program code, we can give beginners a chance to perform the computation, understanding the calculation steps in their mind. For this purpose, we do not intend to describe the theory completely, but we select the minimum contents. However, we introduce some specific examples showing real implementation adopted in "opt" aiming at deep understanding of the readers.

ver. 1.0.0 The first Fortran 90 version adopting only Sopt functions optimizing atomic positions.

ver. 1.1.0 The revised version realizing Allopt functions performing stress calculations and cell optimization.

ver. 1.2.0 The version combining the band-structure calculation.

1 Expression of the total energy

In this section, we give the expression of the total energy in the plane-wave expansion method. We describe the formula when the norm-conserving pseudo-potential

is adopted. This allows readers to understand practical calculation methods in the DFT simulation. We define $\mathbf{R}_I$, Z_I, $\hat{V}_I^{pseudo}(\mathbf{r}-\mathbf{R}_I)$ as the atomic coordinate, the ionic charge, and the pseudo-potential of the I-th atom.

We utilize the next expression for the total energy of a crystal with the volume Ω.

$$E_{total} = E_{kin} + E_{el-el} + E_{xc} + E_{el-ion} + E_{ion-ion}\,. \tag{1}$$

Here, E_{kin} is the kinetic energy of electrons, E_{el-el} represents a part of the electron-electron interaction called the Hartree energy, which is the classical Coulomb energy for the electron charge density. We also have the exchange-correlation energy E_{xc}, the static electron-ion Coulomb interaction E_{el-ion}, and the ion-ion repulsive Coulomb energy $E_{ion-ion}$. We assume that the ion core gives the spherical potential.

Each contribution has the next expression in the real space.

$$\begin{aligned}
E_{kin} &= \sum_{\mathbf{k},n,\sigma} \int_\Omega d\mathbf{r}\ \phi^*_{\mathbf{k},n,\sigma}(\mathbf{r}) \left(-\frac{1}{2}\nabla^2\right) \phi_{\mathbf{k},n,\sigma}(\mathbf{r})\,, && (2)\\
E_{el-el} &= \frac{1}{2}\iint_\Omega d\mathbf{r}\, d\mathbf{r}'\, \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,, && (3)\\
E_{xc} &= \int_\Omega d\mathbf{r}\, \varepsilon_{xc}(n(\mathbf{r}))n(\mathbf{r})\,, && (4)\\
E_{el-ion} &= \sum_{\mathbf{k},n,\sigma}\sum_I \int_\Omega d\mathbf{r}\, \phi^*_{\mathbf{k},n,\sigma}(\mathbf{r})\hat{V}_I^{pseudo}(\mathbf{r}-\mathbf{R}_I)\phi_{\mathbf{k},n,\sigma}(\mathbf{r})\,, && (5)\\
E_{ion-ion} &= \frac{1}{2}\sum_{I,J} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}\,. && (6)
\end{aligned}$$

Here, $n(\mathbf{r})$ is the single-particle density given as,

$$\begin{aligned}
n(\mathbf{r}) &= \sum_{\sigma=\uparrow,\downarrow} n_\sigma(\mathbf{r})\,, && (7)\\
n_\sigma(\mathbf{r}) &= \sum_{\varepsilon_{\mathbf{k},n,\sigma}\le E_F} |\phi_{\mathbf{k},n,\sigma}(\mathbf{r})|^2\,, && (8)
\end{aligned}$$

The Kohn-Sham orbital represented by a wave function, $\phi_{\mathbf{k},n,\sigma}(\mathbf{r})$, is given for the n-th band with the spin σ at the sampled k point, $\mathbf{k}$. The energy of the orbital is given by $\varepsilon_{\mathbf{k},n,\sigma}$. The Fermi energy E_F is for the imaginative independent Fermion system.

2 Dual space formalism

In this package, each wavefunction is expanded in the plane wave basis by adopting the periodic boundary condition. The Bloch theorem and the Fourier transformation allows us to represent the wavefunction in a series expansion of plane waves as,

$$\phi_{\mathbf{k},n,\sigma}(\mathbf{r}) = \sum_{\mathbf{G}} \phi_{\mathbf{k},n,\sigma}(\mathbf{G}) \exp\left(i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}\right)\,. \tag{9}$$

Here, the summation with respect to $\mathbf{G}$ is taken on the reciprocal lattice.

In a real calculation of atoms and molecules, we often consider a periodic structure of atoms and/or molecules, allowing the periodic boundary condition. By this way, we can accelerate the simulation using the speed of the plane wave expansion method. We will explein some extra limitations of the method in the latter part.

We call the unit of the periodic structure "the unit cell". Introducing a large finite integer N, we assume that the unit cells of N^3 are periodically aligned in a lattice. The repeating unit cells are obaying the total periodic boundary condition. Thus we have double periodicity. We introduce a volume Ω_{cell} as the volume of the unit cell. The total system has the volume $\Omega = N^3\Omega_{cell}$.

We consider the Fourier transformation. We will introduce a cutoff so that a finite descrete sum of $\mathbf{G}$ will appear. To derive formulae of the tansilation, however, it is better to derive the equations for continuum system with coordinates $\mathbf{r}$ as continuum variables. Therefore, in eq. (**??**), the summation of $\mathbf{G}$ is an infinite summation. The electron charge density should have the same periodicity as the crystal. Thus we have the Fourier transform of the density.

$$n_\sigma(\mathbf{G}) = \frac{1}{\Omega_{cell}} \int_{\Omega_{cell}} d\mathbf{r} n_\sigma(\mathbf{r}) \exp\left(-i\mathbf{G}\cdot\mathbf{r}\right) , \tag{10}$$

$$n_\sigma(\mathbf{r}) = \sum_{\mathbf{G}} n_\sigma(\mathbf{G}) \exp\left(i\mathbf{G}\cdot\mathbf{r}\right) , \tag{11}$$

In the calculation step, both of the real space and the reciprocal space (or the momentum space) are utilized. The integration for the exchange-correlation energy is usually performed in the real space. the kinetic energy is much easily evaluated in the momentum space. To perform both of numerical integration, we adopt two representations in the real and reciprocal spaces. The method to perform these integration by transforming information from each other is called "the dual space formalism". This idea with the fast Fourier transformation (FFT) allows us to reduce the calculation steps.

The solution of the Kohn-Sham equation is represented as the wavefunction $\phi_{\mathbf{k},n,\sigma}(\mathbf{G})$ in the reciprocal space. The single-particle density given by the wavefunctions determines each energy term in the density functional. To realize the calculation, we should have representations of the density both in the real and reciplocal spaces. The procedure is given as follows.

To have a numerics, $\mathbf{G}$ is restricted in a finite region in the whole reciplocal space. More precisely, the cutoff energy E_{cut} is introduced for the plane-wave expansion. The Fourier coefficient of the wavefunction $\phi_{\mathbf{k},n,\sigma}(\mathbf{G})$ is assumed to be zero, when $|\mathbf{G}|^2 > E_{cut}$. The energy cutoff determines a subspace of the reciprocal space. We define a finite reciprocal mesh, in which $|\mathbf{G}|^2 \leq E_{cut}$ holds. (Fig. **??**) Owing to the discrete Fourier transformation, a corresponding lattice structure is given in the real space. This latter mesh is called the real mesh. Points on the real mesh is denoted as $\mathbf{r}_j$. Now we have the next procedure.

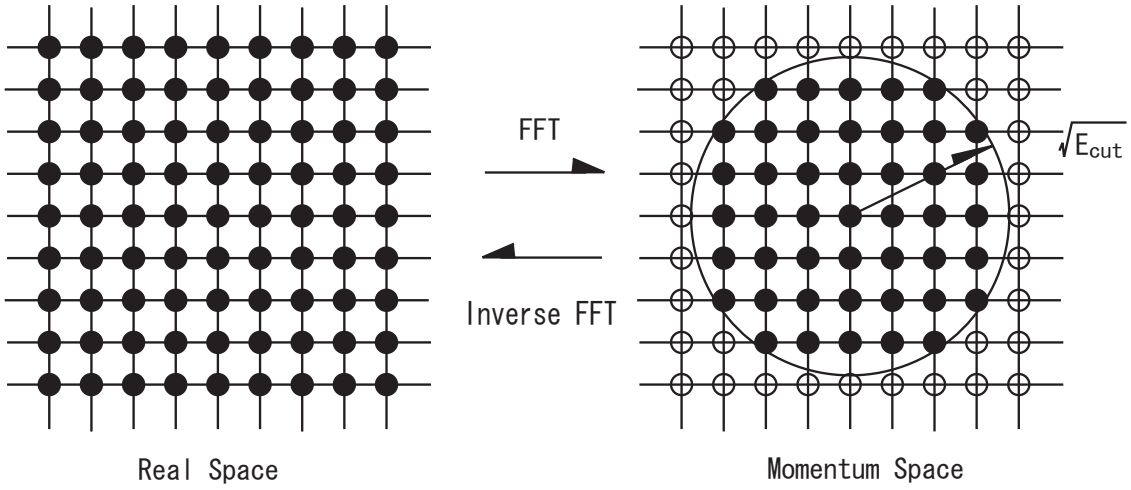

図 1: A schematic viewgraph explaining the real mesh and the reciprocal mesh transformed by the Fourier transformation. In the reciprocal space (the momentum space), the wavefunction has a finite coefficient for the plane wave with $\mathbf{G}$ satisfying $|\mathbf{G}| \leq \sqrt{E_{cut}}$.

1. The wavefunction $\phi_{\mathbf{k},n,\sigma}(\mathbf{G})$ in the reciprocal space is inverse-Fourier-transformed to have another expression $\phi_{\mathbf{k},n,\sigma}(\mathbf{r}_j)$ given by Eq. (**??**) in the real mesh.

2. Using $\phi_{\mathbf{k},n,\sigma}(\mathbf{r}_j)$, the electron density $n_\sigma(\mathbf{r}_i)$ is calculated using Eq. (**??**).

3. The Fourier representation of the density $n_\sigma(\mathbf{r}_i)$ is given by FFT as a series expansion.

In the last step, we utilize,

$$n_\sigma(\mathbf{G}) = \sum_j n_\sigma(\mathbf{r}_j) \exp\left(-i\mathbf{G}\cdot\mathbf{r}_j\right) . \tag{12}$$

3 Expression of the energy

Following the procedure given in the last section, we have $\{\phi_{\mathbf{k},n,\sigma}(\mathbf{G})\}$, $\{\phi_{\mathbf{k},n,\sigma}(\mathbf{r}_j)\}$, $\{n_\sigma(\mathbf{r}_j)\}$, $\{n_\sigma(\mathbf{G})\}$. The actual calculation is done using the formulae given in this section. Each contribution for the energy is given by an expression per a unit cell.

3.1 Kinetic energy

This term is obtained in the reciprocal space. The kinetic energy $\bar{E}_{kin}$ per a unit cell is,

$$\bar{E}_{kin} = \sum_{\mathbf{k},n,\sigma} \sum_{\mathbf{G},\mathbf{G}'} \int_{\Omega_{cell}} d\mathbf{r}\, \frac{|\mathbf{k}+\mathbf{G}|^2}{2} \exp\left(i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}\right) \phi^*_{\mathbf{k},n,\sigma}(\mathbf{G}')\phi_{\mathbf{k},n,\sigma}(\mathbf{G})$$

$$= \frac{\Omega_{cell}}{2} \sum_{\mathbf{k},n,\sigma} \sum_{\mathbf{G}} |\mathbf{k}+\mathbf{G}|^2 |\phi_{\mathbf{k},n,\sigma}(\mathbf{G})|^2 \,. \tag{13}$$

This expression may be used as it is for the discrete representation by rewriting the sumation with a finite sumation with respect to $\mathbf{G}$. The calculation of the kinetic energy is possible by evaluating an approximated expression using the finite difference method, since the total calculation cost will not increase so much. The expression and the treatment is much easy in the momentum space .

3.2 Hartree energy

This term is also evaluated in the reciprocal space. We define $\bar{E}_{el-el}$ as the Hartree energy per a unit cell.

$$\begin{aligned}
\bar{E}_{el-el} &= \frac{1}{N^3} E_{el-el} \\
&= \frac{1}{2} \int_{\Omega_{cell}} d\mathbf{r} \int_{\Omega} d\mathbf{r}' \, \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \\
&= \frac{1}{2} \int_{\Omega_{cell}} d\mathbf{r} \int_{\Omega} d\mathbf{s} \frac{1}{|\mathbf{s}|} \sum_{\mathbf{G},\mathbf{G}',\sigma,\sigma'} n_\sigma(\mathbf{G}) n_\sigma(\mathbf{G}') \exp\left(i\mathbf{G}\cdot\mathbf{r} + i\mathbf{G}'\cdot(\mathbf{r}-\mathbf{s})\right) \\
&= \frac{\Omega_{cell}}{2} \sum_{\mathbf{G},\sigma,\sigma'} n_\sigma(\mathbf{G}) n_\sigma(-\mathbf{G}) \int_{\Omega} d\mathbf{s} \frac{1}{|\mathbf{s}|} \exp\left(i\mathbf{G}\cdot\mathbf{s}\right).
\end{aligned}$$

Here, the divergence appearing when $\mathbf{G} \to \mathbf{0}$ also appears in $E_{ion-ion}$, E_{el-ion} and totally cancelation of these three diverging terms happens. Thus, the diverging term is omitted in this stage. For $\bar{E}_{el-el}$, the term for $\mathbf{G} = \mathbf{0}$ corresponds to this divergence. Thus, we remove this term at $\mathbf{G} = \mathbf{0}$. Furthermore, when $N \to \infty$, if $\mathbf{G} \neq \mathbf{0}$

$$\int_{\Omega} d\mathbf{s} \frac{1}{|\mathbf{s}|} \exp\left(i\mathbf{G}\cdot\mathbf{s}\right) = \frac{4\pi}{G^2}.$$

We can derive $n_\sigma(-\mathbf{G}) = n_\sigma^*(\mathbf{G})$, since the density $n(\mathbf{r})$ is real. If we write $n(\mathbf{G}) = \sum_\sigma n_\sigma(\mathbf{G})$, then we have the next expression.

$$\bar{E}_{el-el} = \frac{\Omega_{cell}}{2} {\sum_{\mathbf{G}}}' \frac{4\pi |n(\mathbf{G})|^2}{G^2} \,. \tag{14}$$

Here, $\sum'_{\mathbf{G}}$ specifies a summation except for $\mathbf{G} = \mathbf{0}$.

When the evaluation is done in the real space, the calculation requires a six dimensional integral and $O(N_{mesh}^2)$ numerical steps with N_{mesh} as the number of real-mesh points. When it is in the reciprocal space, the deterministic process is the FFT steps of $O(N_{mesh} \log N_{mesh})$ numerical steps. Thus, the calculation is easily done in the reciprocal space.

3.3 Exchange-correlation energy

There are several known expressions for the exchange-correlation energy density ε_{xc}. In this package, we adopted the Perdew-Wang expression (PW92). Compared to the Perdew-Zunger expression (PZ81), PW92 applies the spin-polarization dependence by Vosko *et al.* and it is given by weighted parametrization considering the error in each calculation point of the Monte-Carlo calculation Basically, these parameters are determined refering the Diffusion Monte-Carlo calculation of the uniform electron gas system.

$$\bar{E}_{xc} = \int_{\Omega_{cell}} d\mathbf{r} \varepsilon_{xc}\left(n(\mathbf{r})\right) n(\mathbf{r}) \,. \tag{15}$$

The exchange-correlation energy density ε_{xc} may be separated into the exchange contribution ε_x and hte correlation energy density ε_c.

$$\varepsilon_{xc} = \varepsilon_x + \varepsilon_c \,. \tag{16}$$

In ESopt, only the spin-paramagnetic states are treated. The exchange energy density is given as

$$\begin{aligned} \varepsilon_x(r_s) &= -\frac{3}{4\pi r_s}\left(\frac{9\pi}{4}\right)^{\frac{1}{3}} \\ r_s &= -\left(\frac{3}{4\pi n}\right)^{\frac{1}{3}} , \end{aligned} \tag{17}$$

while expression for the correlation energy density is omitted.

3.4 Electron-ion interaction

The charge of ions are treated as the classical charge distribution. For each charge of an ion is described by the pseudo-potential. In this package, E_{el-ion} is calculated by the potential $\hat{V}_I^{pseudo}$. The effect of core electrons is included in the pseudo-potential, so that the state of the valence electrons is explicitly determined in the pseudo-potential method. In other word, the charge distribution of core electrons is fixed. In this package, the energy of the pseudo-potential is given as a summation of the local potential and the non-local potential.

$$\begin{aligned} E_{el-ion} &= E_{el-ion}^{local} + E_{el-ion}^{non-local} \\ &= \sum_I \int_{\Omega_c ell} d\mathbf{r}\, n(\mathbf{r}) V_I^{local}\left(|\mathbf{r}-\mathbf{R}_I|\right) \\ &+ \sum_{\mathbf{k},n,\sigma} \sum_{I,l\neq l_0} \int_{\Omega_c ell} d\mathbf{r}\, \phi^*_{\mathbf{k},n,\sigma}(\mathbf{r}) \delta\hat{V}_I^l(\mathbf{r}-\mathbf{R}_I) \phi_{\mathbf{k},n,\sigma}(\mathbf{r}) \,. \end{aligned} \tag{18}$$

Here, l denotes the angular momentum. The potential adopted in the package, one of components with $l = 0, 1$ or 2 is included in the local potential, where the

specified local component is denoted as l_0. The other components are treated by the non-local potential.

$$V_I^{local}(r) = V_I^{core} + \delta V_I^{l_0} \,. \tag{19}$$

Here, V_I^{core} is the potential by the core electrons.

In the reciprocal space, E_{el-ion} is given as,

$$\begin{aligned} E_{el-ion} &= \Omega_{cell} {\sum_{\mathbf{G}}}' \sum_I S_I(\mathbf{G}) n(\mathbf{G}) V_I^{local}(\mathbf{G}) + \Omega_{cell} n(\mathbf{0}) \sum_I \alpha_I \\ &+ \Omega_{cell} \sum_{\mathbf{k},n,\sigma} \sum_{\mathbf{G},\mathbf{G}'} \sum_{I,l\neq l_0} S_I(\mathbf{G}-\mathbf{G}') \phi^*_{\mathbf{k},n,\sigma}(\mathbf{G}) \delta\hat{V}_I^l(\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}') \phi_{\mathbf{k},n,\sigma}(\mathbf{G}') \,. \end{aligned} \tag{20}$$

$$S_I(\mathbf{G}) = \exp\left(-i\mathbf{G}\cdot\mathbf{R}_I\right) \,. \tag{21}$$

Here, $V_I^{local}(\mathbf{G})$ is the Fourier component of the local potential of the I-th atom, and $S_I(\mathbf{G})$ is the structure factor.

We noted that divergence in E_{el-el}, $E_{ion-ion}$, and E_{el-ion} are perfectly cancelled, when they are summed. Introduction of the pseudo-potential method, however, causes a residual value and the expression of the total energy has a shift. This shift given as the limit of $\mathbf{G} \to \mathbf{0}$ for the expression of the energy is obtained as the difference between the local component E_{el-ion}^{local} in the limit of $\mathbf{G} \to \mathbf{0}$ and the limit of the Coulomb potential $-4\pi Z_I/\Omega_{cell}G^2$ for the I-th atom. The value is given as α_I,

$$\alpha_I = \lim_{\mathbf{G}\to\mathbf{0}} \left(S_I(\mathbf{G}) V_I^{local}(\mathbf{G}) - \frac{4\pi Z_I}{\Omega_{cell} G^2} \right) \,. \tag{22}$$

The energy of the non-local component is calculated by $O(N^2)$ operation steps for N plane waves. The value is reduced to $O(N)$ by adopting an approximation given by Kleinman and Bylander, which is called the K-B approximation. We adopt the formula by the K-B approximation given as,

$$\delta\hat{V}^{non-local} = \sum_{l,m} \frac{|\delta\hat{V}^l \phi_{l,m}\rangle\langle\phi_{l,m}\delta\hat{V}^l|}{\langle\phi_{l,m}|\delta\hat{V}^l|\phi_{l,m}\rangle} \,. \tag{23}$$

Here $\phi_{l,m}$ denotes a pseudo wave function with the angular momentum l with a magnetic quantum number m. The eq. (**??**) is exact, if the wavefunctions $\phi_{l,m}$ form the complete set. In the K-B approximation, since the functions are given by the pseudo wavefunction, the expression becomes an approximated formula in general. In a large number of systems, this approximation is known not to cause any bad effect.

However, in some systems, a side effect by the K-B approximation is known to appear. in a general expression, the non-local potential is given as a local expression in the radial direction, which is called the semi-local potential. When the K-B approximation is adopted, non-locality arises also for the radial direction. Therefore,

the Wronsky theorem becomes inapplicable. Thus, for the K-B approximation, the potential may have a false bound state, which may be energetically lower than a true bound state. This abnormal bound state is called the ghost state.

In the K-B approximation, the non-local potential $\delta\hat{V}^l(\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}')$ is given by the next expression.

$$\begin{aligned}
\delta\hat{V}^l(\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}') &= \frac{1}{\Omega}\int_\Omega d\mathbf{r}\, \exp\left(-i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}\right)\delta\hat{V}^l(r)\hat{P}_l \exp\left(i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}\right) \\
&= \frac{4\pi}{\Omega}(2l+1)P_l\left(\cos\theta_{\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}'}\right) \\
&\quad \times\int_0^\infty d\mathbf{r}\, r^2 j_l(|\mathbf{k}+\mathbf{G}|r)j_l(|\mathbf{k}+\mathbf{G}'|r)\delta\hat{V}^l(r) \\
&= \frac{4\pi}{\Omega}(2l+1)P_l\left(\cos\theta_{\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}'}\right)\frac{\chi_I^l(\mathbf{k}+\mathbf{G})\chi_I^l(\mathbf{k}+\mathbf{G}')}{X_I^l}\,, \quad (24) \\
\cos\theta_{\mathbf{k}+\mathbf{G},\mathbf{k}+\mathbf{G}'} &= \frac{(\mathbf{k}+\mathbf{G})\cdot(\mathbf{k}+\mathbf{G}')}{|\mathbf{k}+\mathbf{G}||\mathbf{k}+\mathbf{G}'|}\,, \quad (25) \\
\chi_I^l(\mathbf{k}+\mathbf{G}) &= \int_0^\infty d\mathbf{r}\, r^2 j_l(|\mathbf{k}+\mathbf{G}|r)\delta\hat{V}^l(r)R_l(r)\,, \quad (26) \\
X_I^l &= \int_0^\infty d\mathbf{r}\, r^2\delta\hat{V}^l(r)|R_l(r)|^2\,. \quad (27)
\end{aligned}$$

Here, $R_l(r)$ is the radial part of the pseudo wavefunction $\phi_{l,m} = Y_{l,m}(\theta,\phi)R_l(r)$ for the pseudo-potential $\delta\hat{V}^l(r)$.

3.5 Ion-ion interaction

This contribution is evaluated by the Ewald summation.

$$\begin{aligned}
E_{ion-ion} &= \frac{1}{2}\frac{1}{N_{cell}}\sum_{I,J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} \\
&= \frac{1}{2}\sum_{n,m,\mathbf{L}}\frac{Z_n Z_m}{|\mathbf{d}_n-\mathbf{d}_m-\mathbf{L}|} \\
&= \frac{1}{2}\sum_{n,m,\mathbf{L}} Z_n Z_m\left(\frac{\mathrm{erfc}(\kappa|\mathbf{d}_n-\mathbf{d}_m-\mathbf{L}|)}{|\mathbf{d}_n-\mathbf{d}_m-\mathbf{L}|}+\frac{\mathrm{erf}(\kappa|\mathbf{d}_n-\mathbf{d}_m-\mathbf{L}|)}{|\mathbf{d}_n-\mathbf{d}_m-\mathbf{L}|}\right) \quad (28)
\end{aligned}$$

Here, $\mathbf{L}$ is the translation vector for the periodic system. The coordinate $\mathbf{R}_I$ for an atom is rewritten as $\mathbf{R}_I = \mathbf{d}_n + \mathbf{L}$ with the position of the atom in a unit cell $\mathbf{d}_n$. The first term in eq. (**??**) is obtained in the real space adn the second term is calculated in the reciprocal space.

$$\begin{aligned}
f(\mathbf{G}) &= \lim_{\mu\to 0}\frac{1}{\sqrt{\Omega_{cell}}}\int d\mathbf{r}\, f(r)\exp\left(-i\mathbf{G}\cdot\mathbf{r}-\mu r\right) \\
&= \frac{4\pi}{\Omega_{cell}}\frac{e^{-(G/2\kappa)^2}}{G^2}\,. \quad (29)
\end{aligned}$$

Here, the Fourier coefficient $f(\mathbf{G})$ behaves in the limit of $\mathbf{G} \to \mathbf{0}$ as,

$$\begin{aligned} f(\mathbf{G}) &\simeq \frac{4\pi}{\Omega_{cell}} \left[1 - \left(\frac{G}{2\kappa} \right)^2 + O(G^4) \right] \frac{1}{G^2} \\ &= \frac{4\pi}{\Omega_{cell}} \left[\frac{1}{G^2} - \frac{1}{2\kappa} + O(G^2) \right] \end{aligned} \tag{30}$$

The first diverging term is cancelled with E_{el-el} and E_{el-ion} so that the second term remains. Therefore, the expression of $E_{ion-ion}$ without the divergence is given using the Ewald parameter γ_{ewald} as,

$$\begin{aligned} \gamma_{ewald} &= \sum_{n,m} Z_n Z_m \left[\frac{4\pi}{\Omega_{cell}} \left(\sum_{\mathbf{G}}{}' \frac{e^{-(G/2\kappa)^2}}{G^2} e^{i\mathbf{G}\cdot(\mathbf{d}_n - \mathbf{d}_m)} - \frac{1}{4\kappa} \right) \right. \\ &+ \left. \sum_{\mathbf{L}}{}' \frac{\mathrm{erf}(\kappa|\mathbf{d}_n - \mathbf{d}_m - \mathbf{L}|)}{|\mathbf{d}_n - \mathbf{d}_m - \mathbf{L}|} - \frac{2\kappa}{\sqrt{\pi}} \delta_{n,m} \right] . \end{aligned} \tag{31}$$

When the number of atoms in a unit cell becomes large, the calculation time for the Ewald summation becomes non-negligible. To save the calculation cost, we may optimize the numerical summation required to achieve convergence. In eq. (**??**), we have cutoffs for summations in the real and reciprocal space. They are estimated as $1/\kappa$ and 2κ. When a side of a unit cell is L, the length of a reciprocal lattice vector becomes $G_0 = 2\pi/L$. In both spaces, the maximum values of lattice vecrors required for the convergence in γ_{ewald} are given as cutoff radii. We have relations for these values as $L_{max} = LN_{Rmax} \propto 1/\kappa$ and $G_{max} \propto G_0 N_{Gmax} \propto 2\kappa$. Therefore, since the total number of the lattice points in each space is estimated as $N_{total} \propto N_{Rmax}^3 + N_{Gmax}^3 \propto \frac{1}{(\kappa L)^3} + \left(\frac{\kappa L}{\pi} \right)^3$ we can minimize the calculation steps by taking $\kappa \simeq \frac{\pi}{L}$. This value becomes $\kappa \simeq \frac{\pi}{(L_x L_y L_z)^{1/3}}$, when the cell is a rectangular parallelepiped.

4 Hellmann-Feynman Force

In this section, we give the expression of the inter-atomic force. The force F_I acting on the I-th atom is given by taking the derivative of the total energy with respect to $\mathbf{R}_I$.

$$\begin{aligned} \mathbf{F}_I &= -\frac{dE}{d\mathbf{R}_I} \\ &= -\sum_i \left[\langle \phi_i | \left(\frac{d}{d\mathbf{R}_I} H \right) | \phi_i \rangle + \left(\frac{d}{d\mathbf{R}_I} \langle \phi_i | \right) H | \phi_i \rangle + \langle \phi_i | H \left(\frac{d}{d\mathbf{R}_I} | \phi_i \rangle \right) \right] \end{aligned} \tag{32}$$

Here, $i(= (\mathbf{k}, n, \sigma))$ is a set of quantum numbers specifying a Kohn-Sham wavefunction. If the Kohn-Sham Hamiltonian is written as H, we have the total energy in a simple expression of $E = \sum_i \langle \phi_i | H | \phi_i \rangle$. In general, we have contribution from

the wave function for the derivative, the calculation cost becomes very big. When wavefunctions ϕ_i give the orthonormalized complete set of the eigen states of H, however, the second term of eq. (**??**) becomes zero.

$$\begin{aligned} &\sum_i \left[\left(\frac{d}{d\mathbf{R}_I} \langle \phi_i | \right) H | \phi_i \rangle + \langle \phi_i | H \left(\frac{d}{d\mathbf{R}_I} | \phi_i \rangle \right) \right] \\ &= \sum_i \varepsilon_i \frac{d \langle \phi_i | \phi_i \rangle}{d\mathbf{R}_I} = 0 \end{aligned} \tag{33}$$

Thus, the expression for the inter-atomic force becomes,

$$\mathbf{F}_I = -\sum_i \langle \phi_i | \frac{dH}{d\mathbf{R}_I} | \phi_i \rangle = -\frac{\partial E}{\partial \mathbf{R}_I} \tag{34}$$

The equation (**??**) has a meaning that the physical force on the atom is given by the partial derivative of the energy with respect to the atomic coordinate, when each wavefunction is the eigen state of the Hamiltonian. This conclusion is sometimes called as the Hellmann-Feynman theorem. Using the force theorem, we can obtain the physical force acting on the atom rather easily. However, even wehn the theorem is adopted, we need to have enough accurate convergence of the total energy to the true ground state energy. In addition, obtaining the force with enough accuracy is much harder than calculating the total energy with the same accuracy. The reason is because the error in the force becomes the first order in the error in the wavefunction, while the error in the total energy is the second order.

When the wavefunction is expanded in the plane waves, we have a special feature that the atomic coordinates do not explicitely appear in the expression. In this condition, when the derivative of the energy in our method using the norm-conserving pseudo-potential is obtained, only two terms from E_{el-ion} and $E_{ion-ion}$ have finite values in the derivative.

$$\mathbf{F}_I = -\frac{\partial}{\partial \mathbf{R}_I} \left(E_{el-ion} + E_{ion-ion} \right) . \tag{35}$$

Here, $\dfrac{\partial E_{el-ion}}{\partial \mathbf{R}_I}$ is given by two contributions from the local component and the non-local component.

5 Quantum Stress

The expression of the internal stress is given by the variational method similarly to the force. Let's consider an electronic state with a unit cell with the volume Ω_{cell}. The total energy is given as $E(\{\psi_{\mathbf{k},i}(\mathbf{r})\}, \{\mathbf{R}_I\})$. The variational principle tells that the ground state energy is given as the minimum of $E(\{\psi_{\mathbf{k},i}(\mathbf{r})\}, \{\mathbf{R}_I\})$ by optimizing the variables, $\{\psi_{\mathbf{k},i}(\mathbf{r})\}, \{\mathbf{R}_I\}$. To obtain the internal stress, we need

to consider the symmetrized strain tensor ε by eliminating the lotational degrees of freedom as the second order tensor. By the scaling relation with respect to the infinitesimal change in ε, we have the next expression.

$$\begin{aligned} \mathbf{R}_I &\rightarrow \mathbf{R}'_I = (1+\varepsilon)\mathbf{R}_I, \\ \psi_{\mathbf{k},i}(\mathbf{r}) &\rightarrow \psi_{\mathbf{k},i}(\mathbf{r}') = \det(1+\varepsilon)^{\frac{1}{2}}\psi_{\mathbf{k},i}(\mathbf{r}). \end{aligned}$$

The ground state energy shift as,

$$E_{tot} \rightarrow E(\varepsilon) = E_{tot} + \Delta E(\varepsilon).$$

This shift $\Delta E(\varepsilon)$ is expanded in a series with respect to ε. The stress tensor σ is given as the expansion coefficient in the first-order term.

$$\begin{aligned} \Delta E(\varepsilon) &= -\mathrm{Tr}(\sigma\varepsilon)\Omega_{cell} + O(\varepsilon^2), \\ \sigma_{\alpha\beta} &= -\frac{1}{\Omega_{cell}}\frac{\partial E(\varepsilon)}{\partial \varepsilon_{\alpha\beta}}\bigg|_{\varepsilon\rightarrow 0}. \end{aligned} \tag{36}$$

When the strain and the stress become isotropic, we have the well-known expression of $P = -\dfrac{dE}{d\Omega_{cell}}$.

The strain tensor $\varepsilon_{\alpha\beta}$ is a symmetric tensor and $\mathbf{G}$ behaves as $(1-\varepsilon)\mathbf{G}$ in the first order. Moreover, Ω_{cell} changes as $\det(1+\varepsilon)\Omega_{cell}$. Since $\Omega_{cell}n(\mathbf{G})$ and $S_I(\mathbf{G})$ are invariant, we have the next expression.

$$\sigma_{\alpha\beta} = \sigma^{kin}_{\alpha\beta} + \sigma^{el-el}_{\alpha\beta} + \sigma^{xc}_{\alpha\beta} + \sigma^{local}_{\alpha\beta} + \sigma^{non-local}_{\alpha\beta} + \sigma^{\alpha}_{\alpha\beta} + \sigma^{ewald}_{\alpha\beta}.$$

$$\sigma^{kin}_{\alpha\beta} = 2\sum_{\mathbf{k},\mathbf{G},i} |\psi_i(\mathbf{k}+\mathbf{G})|^2(\mathbf{k}+\mathbf{G})_\alpha(\mathbf{k}+\mathbf{G})_\beta,$$

$$\sigma^{el-el}_{\alpha\beta} = -\frac{1}{2}{\sum_{\mathbf{G}}}' \frac{4\pi|n(\mathbf{G})|^2}{|\mathbf{G}|^2}\left(\frac{2\mathbf{G}_\alpha\mathbf{G}_\beta}{|\mathbf{G}|^2} - \delta_{\alpha\beta}\right),$$

$$\sigma^{xc}_{\alpha\beta} = \delta_{\alpha\beta}\frac{1}{\Omega_{cell}}\int_{\Omega_{cell}} d\mathbf{r} n(\mathbf{r})(\mu_{xc}(n) - \varepsilon_{xc}(n)),$$

$$\sigma^{local}_{\alpha\beta} = -{\sum_{\mathbf{G},I}}' n^*(\mathbf{G})S_I(\mathbf{G})\frac{\partial V^{local}_I(\mathbf{G})}{\partial\varepsilon_{\alpha\beta}},$$

$$\sigma^{non-local}_{\alpha\beta} = -2\sum_{\mathbf{k},i}\sum_{\mathbf{G},\mathbf{G}'}\sum_{I,l} S_I(\mathbf{G}-\mathbf{G}')\psi^*_{\mathbf{k},i}(\mathbf{G})\psi_{\mathbf{k},i}(\mathbf{G}')\frac{\partial\delta\hat{V}^l_I(\mathbf{k}+\mathbf{G},\mathbf{k}'+\mathbf{G}')}{\partial\varepsilon_{\alpha\beta}}$$

$$\sigma^{\alpha}_{\alpha\beta} = \delta_{\alpha\beta}n^*(\mathbf{0})\sum_I \alpha_I,$$

$$\sigma^{ewald}_{\alpha\beta} - \frac{1}{\Omega_{cell}}\frac{\partial\gamma_{ewald}}{\partial\varepsilon_{\alpha\beta}}.$$

Here, we introduced a relative coordinate $\mathbf{q}_I$ for the atomic coordinate, which behaves $\mathbf{R}_I(t) = (1+\varepsilon)\mathbf{q}_I(t)$.

6 Structure of the package

The structure of ESopt has a form explained in this section at present, although an essential change might happen in future. When `ESopt.tar.gz` is unpacked, you have a directory `ESopt`. In this directory, you have subdirectories named `INPUT_DATA`, `mknon`, `commons`, `params`. We explain contents of these directories.

6.1 ESopt

In this main directory, program files (`*.f90`) and `makefile` are contained as well as some script files to create working directories. In `ESopt`, input files should be prepared in the directory `INPUT_DATA`. Then, you have to do `make` in the main directory Excecution of the calculation is done in the working directory, whose name is specified by a line of `makefile`.

6.2 INPUT_DATA

In this directory, all input files are stored. Users have to create files the main input file `CORD` and some controlling parameters in `*.CNTL`. In `CORD`, the structure of the material and the conditions for the calculation are given by parameters, which are required also in the compiling step. The control files `*.CNTL` determine a parameter set, which are read from these files at run time. These controlling parameters may change the action of the excecutable file called "opt". In this directory we have also the pseudo-potential parameter files and data files specifying the sampling k-point and k vectors (or lines in the Brillouin zone) used in the band structure calculation.

6.3 mknon

In this directory, there are sub-programs to determine a parameter set required in the actual computation. Because the programs detemine internal parameters, the user is just requested to define essential input parameters. The first sub-program `kbexe` obtains the mesh for FFT, the number of Fourier components, and the cutoff using the size of the unit cell. The number of states as well as the total charge is given by a summation of the valence electrons in each atom.

The subprogram outputs some of parameters specific to a system in a data file `parameters.data`, which is written in a Fortran unformatted form, Others are stored in `input.dat`, which is also an unformatted file. The second subprogram `mknon` obtaines the FFT mesh in a rectangular distribution in the momentum space and a list of the Fourier components in the cutoff radius by defining correspondence to the mesh points, and outputs the arrays in `gpt_list`, whic is also unformatted. These sub-programs run automatically, when the user executes `make`.

6.4 commons and params

In this package, variables and arrays are transfered among subroutines by the following rule. The essential arrays are defined as the global arrays in a module by the definition of the public array, which are refered in each subroutine. Some of variables are written in a common statement. This is because our revision of the original Fortran 77 package is not complete. Using parameters determined by `mknon`, the size of each array is obtained to use in the allocation of the global array. The data `mknon/parameters.data` is read by `parameters.f90`, while arrays are given by `globalarray.f90`. Some variables commonly used in subroutines are written as a header file stored in `commons`. In `params`, some parameters including the LDA parameters, which are independent of material, are stored.

7 Overview of the action

A typical procedure to perform computation using ESopt is the following.

1. By creating `INPUT_DATA/CORD`, the user set the system parameters including atomic coordinates, atomic species, and the unit cell, as well as the essential conditions for the cutoff radius, and the k-sampling method.

2. The user should check and write *destination*, which determines the name of the working directory and is written in the line `DESTDIR=`*destination* at the head of `makefile`. In this instruction, we omit a description how to handle the machne dependence.

3. Do `make`.

4. Next, `cd` *destination*. Set the control files, `SOPT.CNTL`, `ALLOPT.CNTL`, `FILE_IO.CNTL`, for the execution properly.

5. Run the prgram by `./opt`, or by submitting job by `qsub qsub` using the job scheduling system, SGE. In the last action, the first `qsub` is the command, while the second `qsub` is a script prepared when you run `make`.

8 Inputs and outputs

As explained in the last section, the main steps are 1) prepare `INPUT_DATA/CORD`, `makefile`, 2) `make`, 3) change your directory to the working directory, 4) prepare `*.CNTL`. Then the preparation for the run is completed. Now instruction of inputs and explanation of outputs are given.

8.1 CORD

The format of `INPUT_DATA/CORD` are the following. This example describes Si in the diamond structure.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ <<<<<<<<<<<<<<<<<<  Basic Inputs  >>>>>>>>>>>>>>>>>>>>>>>>>>>+
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#Atom coordinates relative to unit vectors,       Atom,mode.
 0.250000000D+00  0.250000000D+00  0.250000000D+00  Si o
 0.500000000D+00  0.500000000D+00  0.500000000D+00  Si o
#^              ^  ^              ^  ^              ^  ^^ ^
```

This part determines the atomic coordinates in the internal coordinate, which is the relative coordinate with respect to the unit vectors. The input field starts from `#` and ends at the second `#`. The number of lines gives the number of atoms. Each component of the relative coordinate has to be written by numbers filling an area from `^` to `^`. This is a fixed format. The atomic species are given by two characters identifying the lower-case letter and the chapital letter. The user specify with the value as `Si`. A small letter in `mode` specifies the motion of the atom, where the atom with `o` is optimized and that with `f` is fixed. The number of valence electrons and the pseudo potential parameters like l_0 for each element is written in `INPUT_DATA/TABLE_OF_ELEMENTS`.

```
================<< Other basic inputs >>=========================


 ------------------------------------------------------------------
 | unit_vec:  Primitive vectors of unit cell in real space [Ang]|
 | kp_file:   Sampling type of primitive F.B.Z.                  |
 |                      [file name with 6 char.]                 |
 | E_cut:     Cutoff energy for plane-wave [Ry]                  |
 | band_calc: Choice [yes|no|metal|bands|mband]                  |
 | f_crt:     Critical force [Ry/a.u.] for judging convergency.  |
 | f_tol:     Parameter for judging convergency of electronic    |
 |            deg. freedom.(Relative )                           |
 | calc_mode: Calculation mode.(bas[ic] or ext[ended])           |
 | ideb:      ideb = 1; Running verbose mode.                    |
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


 &basic_input
 unit_vec = 2.715, 0.000, 2.715,
```

```
           2.715, 2.715, 0.000,
           0.000, 2.715, 2.715,
kp_file   = 'DP10GM',
E_cut     = 8.0,
band_calc = 'metal',
f_crt     = 5.0D-03,
f_tol     = 1.0D-8,
calc_mode = 'bas',
ideb      = 0
&end
```

This section is given by the namelist format of Fortran.

- `unit_vec` : The unit vectors for the unit cell are given in

$$\begin{matrix} a_{1x} & a_{1y} & a_{1z} \\ a_{2x} & a_{2y} & a_{2z} \\ a_{3x} & a_{3y} & a_{3z} \end{matrix}$$

  In this program, the reciprocal lattice vectors are defined by

$$\mathbf{a}_i = (a_{ix}, a_{iy}, a_{iz}), \tag{37}$$

  and obtained as

$$\mathbf{b}_1 = \frac{\mathbf{a}_2 \times \mathbf{a}_3}{2\pi \mathbf{a}_1 \cdot \mathbf{a}_2 \times \mathbf{a}_3}, \mathbf{b}_2 = \frac{\mathbf{a}_3 \times \mathbf{a}_1}{2\pi \mathbf{a}_1 \cdot \mathbf{a}_2 \times \mathbf{a}_3}, \mathbf{b}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{2\pi \mathbf{a}_1 \cdot \mathbf{a}_2 \times \mathbf{a}_3}. \tag{38}$$

- `kp_file` : This line specifies the file for the k-point list used in the sampling. In this package, the list files are prepared and written in the input manually. `'DP10GM'` is for the cubic system using only the symmetry of the point group O.

- `E_cut` is the cutoff energy E_{cut} in the atomic unit, Ry.

- `band_calc` is a switch to change the calculation scheme. If it is `'metal'`, the occupation number around the Fermi level is smeared with the Fermi distribution function. If `'bands'` is selected, the band structure calculation using the mode `'bas'`.

- `f_crt` : The convergence criteria for the interatomic force.

- `f_tol` : The convergence criteria for the CG iteration in the electronic state calculation.

- `calc_mode` : The switch for the calculation mode. If it is `'bas'`, optimization of the atomic position is possible. If it is `'ext'`, optimization of the unit cell is possible with a constant pressure mode. `ideb` : A parameter controling the output (1 for the verbose mode, 0 for the ordinal mode.)

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+<<<<<<<<<<<<<< Mode dependant parameters >>>>>>>>>>>>>>>>>>>>+
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

==============<< Case 1;  band = yes or metal. >>=============

 ---------------------------------------------------------------
 | pwidth: Width of graduall cutoff for occupation number[Ry].|
 | incw:   Eigen states is sorted in every incw CG steps.     |
 | ncband: Number of un-occupied states.                      |
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 &METALD
 pwidth=0.005,
 incw=10,
 ncband=4
 &END
```

The part "Case 1" gives parameters for the mode `'metal'`. `pwidth` determines the width for the Fermi distribution. `incw` is the number of an interval between the neighboring steps, when the program checks the order and the occupation of the energy eigen states around the Fermi level.

```
====<<  Case 2;  cmode = ext. (Variable cell simulation.) >>====

 ___[ Constant-cutoff is mimiced with following parameters. ]___
 |                                                             |
 | vke:    Kinetic energy is scaled in vke where               |
 |         |k+G|^2 is beyond E_cut.                            |
 | sigma:  Above scaling is faded in over sigma.               |
 | defmax: Maximum deformation of unit-cell allowed in this    |
 |         simulation.                                         |
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Original
 vke = 2.0,
 sigma = 0.05,
 defmax = 3*0.05
```

```
&mimicd
vke = 2.0,
sigma = 0.10,
defmax = 3*0.05
&end
```

The part "Case 2" controls a function to reduce a possible sudden change in the list of the G vectors, when the constant-pressure molecular dynamics is done. This part is not in use for the present package.

8.2 SOPT.CNTL

"Sopt" performs the fixed cell mode. The cell is not necessarily the same as the unit cell of a crystal, but may be a super cell. `INPUT_DATA/SOPT.CNTL` is written in the namelist format of Fortran. The contents are the following.

```
&soptcntl
itmaxs=1,
f_crt=0.005,
ftol = 1.0d-9,
itmaxd=200,
delta = 0.4,
ideb = 0
&end
```

- `itmaxs` : The maximum allowed steps for the structural optimization.
- `f_crt` : The convergence criterion for the structural optimization. When the maximum of the absolute value of an interatmic force becomes less than this value, the convergence is assumed to be achieved. The unit is A.U.
- `ftol` : The convergence criterion for the electronic state calculation. This value is compared with the relative change in the total energy per a CG step.
- `itmaxd` : The maximum value of CG steps in the electronic state calculation.
- `delta` : The width of the optimization step for the atomic structure. This is a proportional constant for the interatomic force.

8.3 ALLOPT.CNTL

"Allopt" performs the variable cell mode. Using this mode, the full structural optimization and the constant-pressure molecular dynamics are realized. The contents of `INPUT_DATA/ALLOPT.CNTL` are the following.

```
0. Chose a mode.

 mode = 'md|op': Molecular Dynamics or Optimization.
 In case that MD is chosen, you decide whether "Nose's constant
 temparature formula" is used or not.

 &modesw
 mode='md',
 nose = 'yes',
 &end
```

- Using functions of "Allopt", refering the interatomic force and the internal stress, we can make the simulator to perform the molecular dynamics by `mode='md'`, or the structural optimization by `mode='op'`. When `nose='yes'` is specified, the Nosé thermostat is used to control the temperature. If `nose='no'`, opt performs a molecular dynamics without temperature control.

```
0-a. Mode dependant parameters.

 nose:          In case that nose = 'yes', constant temparature simulation
                with following parameters is performed.
        unit:   The unit of temparature.[au|K|eV]
        temp:   Temperature.
        q_inv:  This value corresponds to Q^(-1).
                In case that q_inv = 0, then zeta = 0.
                See, Nose, Solid State Physics, Vol. 24, pp. 98-106(1989).
                     Nose, Mol. Phys., Vol. 52, p 255(1984).
                 q_inv = 0.05??
 &noseparam
 unit = 'K',
 temp = 100.0,
 q_inv = 0.01,
 &end

 p_crt: Convergence parameter to judge convergence of pressure in [GPa].
```

```
gamma: Friction ratio. gamma is in [0-1]. If gamma =1, MD is done.

&optparam
f_crt=1d-10,
ftol=1d-8,
p_crt=0.0001,
gamma = 0.5,
&end
```

- `unit` determines the unit of temperature, which may be in Kelvine or in electron volt (eV). The value is given by `temp`. Another parameter `q_inv` is the Nosé parameter.
- `f_crt` The convergence criterion of the interatomic force for the structural optimization. The unit is in the atomic unit.
- `ftol` The convergence criterion of the total energy of the electronic state.
- `p_crt` The convergence criterion of the pressure for the structural optimization. The unit is in Ry.
- `gamma` A damping factor of the velocity for both the atoms and the cells.

```
================================================================================
1. Define external environment.

--------------------------------------------------------------------------------
1-1. Pressure

 iso: A switch to select isotropic or anisotropic pressure.
        Case; iso = 'no', pext_3d_[i|f](1:3) is read.
        Case; iso = 'yes', pext_3d_[i|f] is not read.
 pext_ini: initial pressure
 pext_fin: final pressure

 &extpress
 iso='no',
 pext_init=0.0,
 pext_final=0.0,
 pext_3d_i=0.0,0.0,0.0,
 pext_3d_f=0.0,0.0,0.0,
 &end
```

- `iso` determines whether the isotropic pressure is applied or not.
- A set of `pext_init` and `pext_final` or another set of `pext_3d_i` and `pext_3d_f` we can specify the initial and final values of the external pressure in the molecular dynamics.

```
------------------------------------------------------------------------
1-2. Parameters concernig how the pressure is introduced and how MD runs.

 nsec_p:        External pressure is gradually enforced.
                Sectioned by nsec_p steps between pext_fin and pext_ini.
 nstp_i:        nstp_i MD steps are performed at the pext_ini.
 nstp_g:        nstp_g MD steps are performed at each pressure
                except for pext_ini.

 &prscntl
 nsec_p=10,
 nstp_i=10,
 nstp_g=1,
 &end
```

- Using `nsec_p`, `nstp_i`, and `nstp_g` we can control the change in the external pressure.

```
========================================================================
3. Parameters for controlling evolution.

 delt_t:        Time interval in femto second.
 ww:            Fictitious mass ratio [w/mass]
 itmaxa:        Maximam iterations for MD.
 itmaxd:        Maximam iterations in diagw.

 &mdparams
 delt_t=5.0,
 ww = 100000.0,
 itmaxa=500,
 itmaxd=200,
 &end
```

- `delt_t` : The time step for the molecular dynamics.

- `ww` : The imaginative (artificial) mass for the unit cell.
- `itmaxa` : The total steps of the molecular dynamics.
- `itmaxd` : The maximum steps for the CG steps in the electronic structure calculation.

```
=============================================================================
4. Initial condition.

 ekin_init:     Initial kinetic energy is scaled in this value.
 init_conf:     Switch how is the initial velocities are given.
        Case; init_conf = 'given', read from iveloc namelist in this file.
        Case; init_conf = 'rando', random vectors, generated with 'iseed',
                         is used.
 ivpoint:       Initial kinetic energy is given at ivpoint step.

 &initveloc
 ekin_init = 10.0,
 ckin_init = 0.0,
 init_conf = 'given',
 iseed = 180,
 ivpoint = 1,
 &end

-----------------------------------------------------------------------------
4-1. Define initial direction to move.
     This parameters are read only in case init_conf = 'given'.

how:  Case; how = 'rel', final conf - initial conf is used.
      Case; how = 'abs', final conf. is used as a vector.

latv_i(3,3): initial primitive vector in real space [Ang]
latv_f(3,3): final primitive vector in real space [Ang]

coordi(3,natm): initial coordinate
coordf(3,natm): final coordinate

Example for structural transformation from cubic-diamond to beta-tin
of silicon.

 &iveloc
```

```
 how = 'rel',
 latv_i =
 2.715, 0.000, 2.715,
 2.715, 2.715, 0.000,
 0.000, 2.715, 2.715,
 latv_f =
 2.715, 0.000, 2.715,
 2.715, 2.715, 0.000,
 0.000, 2.715, 2.715,
 coordi =
 0.000000000,  0.000000000,  0.000000000,
 0.500000000,  0.500000000,  0.000000000,
 coordf =
 0.000000000,  0.000000000,  0.000000000,
 0.500000000,  0.500000000,  0.000000000,
 &end
```

- This part describes how to fix the initial velocity. In ESopt, this final part of `coordi` and `coordf` are skipped.

8.4 FILE_IO.CNTL

The contents of `INPUT_DATA/FILE_IO.CNTL` are the followings.

```
=====================================================================
1. Parameter(s) to controll output.


 &file_io_cntl
 read_file='no',
 initial_file=1
 &end


intchg: interval of charge output

 &outcntl
 intchg = 10,
 intefl = 20,
 intwf = 10,
 &end
```

2. Debug Flags

```
&debug_fl
ideb = 0
&end
```

- `read_file` : This parameter specifies a way to read the result of the atomic configuration and the wavefunction determined in a former calculation. One of `'cont'`, `'wyes'`, `'cyes'`, `'syes'` or `'no'` are selected. The action for each option is the following.
  - `'cont'`: The structure given by `STRUCT/strct` and the wavefunction in `fort.98` are read.
  - `'wyes'`: The wavefunction in `fort.98` is read.
  - `'cyes'`: Structural data in `STRUCT/cell-XXX` and `STRUCT/latcrd-XXX` is read. Here, `XXX` means a number given in the format `I3.3`, which is specified by `initial_file`. The wavefunction in `fort.98` is read.
  - `'syes'`: Structural data in `STRUCT/cell-XXX` and `STRUCT/latcrd-XXX` is read. Here, `XXX` means a number given in the format `I3.3`, which is specified by `initial_file`.
  - `'no'`: The structure is given by `CORD` in `INPUT_DATA` and the wavefunction prepared in the program is used.
- `initial_file` : When `read_file='cyes'` or `read_file='syes'`, this value gives `XXX`. If `read_file` has other value, this option is neglected.
- `intchg` : The interval for the steps, when the charge density is output.
- `intefl` : The interval for the steps, when `MONIT/energy` is output. If the user wants to reduce the time cost for IO, this value should be increased.
- `intwf` : In an older version, this value specified the interval for output of `MONIT/wave000` storing the wave function. In the present version, the value is given in `fort.99`. If the user wants to use this information to continue the simulation, this file may be copied into `fort.98`.

8.5 kpoint

The contents of `kpoint` is the following.

```
number of line (i4) =   7
                      ^^^^
 G X K G L K W X
number of mesh (i4) =  10
 1.000000000  0.000000000  0.000000000
 1.000000000  0.500000000  0.500000000
number of mesh (i4) =  10
 1.000000000  0.500000000  0.500000000
 0.750000000  0.375000000  0.375000000
number of mesh (i4) =  10
 0.750000000  0.375000000  0.375000000
 0.000000000  0.000000000  0.000000000
number of mesh (i4) =  10
 0.000000000  0.000000000  0.000000000
 0.500000000  0.500000000  0.500000000
number of mesh (i4) =  10
 0.500000000  0.500000000  0.500000000
 0.750000000  0.375000000  0.375000000
number of mesh (i4) =  10
 0.750000000  0.375000000  0.375000000
 0.750000000  0.250000000  0.500000000
number of mesh (i4) =  10
 0.750000000  0.250000000  0.500000000
 0.500000000  0.000000000  0.500000000
```

The value of `number of line` determines the number of k lines in the Brillouin zone, on which the energy dispersion is obtained. The number is limited by 20. Assume here that the number is 7 as above. These k lines have to form a continuous loop in the zone. The next filled line below an empty line shows the name of the special points determining two ends of each k lines. If you want, you may use conventional names owing to their symmetry. Next, we use three lines per each k line. The number of divisions of the k line and two end points in the reciprocal lattice of $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ are given.

8.6 Outputs

In the calculation, a sub-directory named by `MONIT` is created in the working directry, and in this directory there apperas temporal information on the energy *etc.*. The total energy is in `energy`. The wavefunction may be given in `intwf`, if properly specified in the code. (This option is switched of in ESopt.) When

the structural optimization is finished, `wave999` for the wavefunction is given. The format of the wavefunction data is unformatted.

The charge density is stored in another subdirectory `CHARGE`. The file name is `chgdnsXXX`.

The major output is given in the following files.

- For Sopt, see the directory `RESULTS`.
  - `cord***` : Atomic coordinates in the real space.
  - `eig***` : The eigen values of the Kohn-Sham equation.
  - `gradient***` : The interatomic force.
  - `H***` : The diagonal elements of a $3\times$ 3 matrix given by the unit vectors, the total energy and the volume of the cell.
  - `Htot***` : Each contribution for the total energy: 1. the kinetic energy of the electron system, 2. the Hartree term, 3. the exchange-correlation term, 4. the potential energy of the local potential part, 5. the potential energy given by the non-local pseudo-potential part, 6. the α term, 7. the Ewald summation, 8. and the summation from 1 to 7.
- For Allopt, see the directory `STRUCT`.
  - `cell***` : The lattice vectors.
  - `latcrd***` : The internal coordinates.
  - `press***` : The external pressures.
  - `realcrd***.xyz`: The atomic coordinates in the real space.
  - `result***` : The total energy same as `Htot***` for Sopt, the enthalpy, the stress tensor, and the atomic forces.
  - `strct***` : The parameters of the unit cell, as `cell***`, `latcrd`, and `zeta` in this order.
  - `strn***` : The strain tensor.

9 Source codes of the program

In this section, we show some example of the implementation of each calculation procedure found in the subroutines, in order to provide information for understanding the action of the program.

9.1 Kinetic energy

The kinetic energy is given by a function `ekin(inw)` in `etotal.f90`.

```
!***********************************************************************
      double precision function ekin(inw)

      use parameters
      use globalarray

      implicit none

      include 'commons/common.h'

      integer :: ik,ib,ipw,inw
      real(8)  frm,z,fd,x

!  -----<< definition of fermi distribution function >>-----

      frm(z) = 1d0 + vke_mim/( 1d0 + exp( -1d0*beta_mim*z ) )

!  ----------<< calc. of KE (mimicing) >>----------
!* epw[Ry],gk[a.u.],nadd

      ekin = 0d0

      do ib = 1,mxband_g
         do ik = 1,nk_g
            fd = fdist_g(ib,ik)
            do  ipw = 1,mxpwk(ik)

               x = 0.5d0*( gk_g(ipw,ik) - ec_mim )

               ekin = ekin &
                    + gk_g(ipw,ik)*frm(x) &
                    * (  dreal(wvfn_g(ipw,ib,ik,inw))**2 &
                    + dimag(wvfn_g(ipw,ib,ik,inw))**2 ) * fd
            end do
         end do
      end do
```

```
      ekin = 2d0*ekin/dfloat(natm_g)/effnk

      return
      end
```

- `inw`: To realize the CG steps, the wavefunction is stored in a few arrays. This index specifies the array for the wavefunction.
- `ib`, `ik`, `ipw`: These indices represent the band index, $\mathbf{k}$, and $\mathbf{G}$. The list of $\mathbf{G}$ appearing in the planewave $\exp(i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}))$ is given for each $\mathbf{k}$ so that the energy of the plane wave is below E_{cut}.
- `gk_g(ipw,ik)`: The array to store $|\mathbf{k}+\mathbf{G}|^2$.
- `wvfn_g(ipw,ib,ik,inw)`: The array to store the wavefunction $\phi_{\mathbf{k},n}(\mathbf{G})$.

In the variable `ekin`, we have

$$E_{kin} = 2\sum_{\mathbf{k},n,\mathbf{G}} |\mathbf{k}+\mathbf{G}|^2 |\phi_{\mathbf{k},n}(\mathbf{G})|^2 \,.$$

The value of `ekin` is thus the energy per a unit cell in Ry. Here the electron mass becomes $m = 1/2$. The spin degrees of freedom is just taken into account by multiplying the factor 2. Another factor `frm(x)` represents a smearing method to achieve a constant-cutoff simulation for the constant-pressure variable-cell simulation.

9.2 Hartree energy

The Hartree energy is given by a function `eee(vcell)` in `etotal.f90`.

```
!***********************************************************************
      double precision function eee(vcell)

      use parameters
      use globalarray

      implicit none

      include 'commons/common.h'

      integer :: ig
      real(8) :: vcell

      eee = 0d0
```

```
      do ig = 2,ng_g

         eee = eee + (dreal(chgg_g(ig))**2+dimag(chgg_g(ig))**2)*gg_g(ig)

      end do

      eee = eee*4d0*pi/vcell/dfloat(natm_g)

      return
      end
```

- `chgg_g(ig)`: An array to store the Fourier transform of the charge $n(\mathbf{G})$.
- `gg_g(ig)`: An array storing $1/|\mathbf{G}|^2$.

We have the value of the Hartree energy

$$E_{el-el} = \frac{4\pi}{\Omega_{cell}} \sum_{\mathbf{G}}{}' \frac{|n(\mathbf{G})|^2}{|\mathbf{G}|^2} \,,$$

in `eee`. The value of `eee` is also given per a unit cell in Rydberg (Ry). The electron charge is thus $e^2 = 2$. The reason for the factor of Ω_{cell}^{-1} is that E_{el-el} is originally given in the double integral in the real space.

9.3 Total energy

The total energy is given by a subroutine `etotal(vcell,inw,etot)` in `etotal.f90`.

```
!************************************************************************
      subroutine etotal (vcell,inw, etot)

      use parameters
      use globalarray

      implicit none

      integer :: lmx,inw,i,j
      real(8) :: etot(8),vcell,vatm, &
           ekin,eee,exc,evl,evnl,t1,t2,dsecnd

      parameter( lmx = 2 )

      include 'commons/common.h'

      t1 = dsecnd('cpu')

      vatm = vcell/dfloat(natm_g)
```

```
etot(1) = ekin(inw)
etot(2) = eee(vcell)
etot(3) = exc(vcell)
etot(4) = evl()
etot(5) = evnl(inw)

etot(8) = etot(1)+etot(2)+etot(3)+etot(4)+etot(5)+etot(6)+etot(7)

t2 = dsecnd('cpu')

cpu_time(1) = cpu_time(1) + t2-t1

return
end
```

The energy is given for each contribution and the meaning of these parts are the followings.

- `' ekin  = ',etot(1)` : The kinetic energy.
- `' eee   = ',etot(2)` : The Hartree energy.
- `' exc   = ',etot(3)` : The exchange-correlation energy.
- `' evl   = ',etot(4)` : The local-potential contribution.
- `' evnl  = ',etot(5)` : The contribution from the non-local potential.
- `' ealp  = ',etot(6)` : The α_I term.
- `' eewd  = ',etot(7)` : The ion-ion interaction (The Ewald summation).
- `' etot  = ',etot(8)` : The total energy.

9.4 Sopt control routine

The main routine is given to switch the calculation mode by calling a subroutine.

- Allopt : The optimization of the lattice structure and the atomic structure. The constant-pressure molecular-dynamics is possible.
- Sopt : The optimization of the atomic structure with a fixed unit cell.
- DMD : The damped dynamics with a damping factor.

ESopt is provided by stopping some functions except for Sopt and limited Allopt. Here, we consider the function of `sopt`. The code below is given by omitting some extra comments.

```
!*********************************************************************
      subroutine sopt()

      use parameters
      use globalarray

      implicit real*8(a-h,o-z)

      include 'commons/lattice.h'
      include 'commons/common.h'

      logical(4) ::  exist1, exist2
      real(8) :: etot(8),delta,cpu0,cpu1

      namelist / soptcntl / itmaxs, ftol, f_crt, itmaxd, delta, ideb

!=====================================================================

      inquire(file='SOPT.CNTL',exist=exist2)
      inquire(file='../INPUT_DATA/SOPT.CNTL',exist=exist1)

      if(exist2) then
         open(10,file='SOPT.CNTL',form='formatted', &
              iostat=ios,status='old')
         read(10,soptcntl,iostat=ios)
         close(10)
      else
         if(exist1) then
            open(10,file='../INPUT_DATA/SOPT.CNTL',form='formatted',&
                 iostat=ios,status='old')
            read(10,soptcntl,iostat=ios)
            close(10)
         else
            stop 'Can not open SOPT.CNTL'
         endif
      endif

!  ----------< generate g-points >---------------

      call gptgen()

!  -----< calc. some components of total energy >----------

      call vinit()

!  -----< calc. ewald sum (independent part with coordinates of atoms ) >-----

      call  iniewd()
```

```
! Main loop starts.
      do its = 1, itmaxs

! ----------< optimization for electronic state >----------

      call getll(etot,nloop)

      call charge(0)

! ----------< calc. Hellmann-Feynman force >----------

      call force(vcell)

      call eforce(natm_g,stoa,grd,frc,fn)

      call wfile(etot,its)

! ----------< output data >----------

      call wfile0(000,its)

! ----------<  judgement of convergence >-------------

      if(ideb.eq.1) write(*,*)'fn, f_crt = ',fn,f_crt
      if(fn.lt.f_crt) exit

! ----------< move all atoms >-------------------------------

      call mvatm(delta)

      end do
! Main loop ends.

      if(fn.lt.f_crt) then
         call wfile0(999,its)
      end if

      write(*,*)'elapse time in sopt = ', cpu1-cpu0
      return

      end subroutine sopt
```

We can see that the algorithm consists from the following steps.

1. Initialization.
   (a) Input `SOPT.CNTL`.
   (b) `gptgen()` : Creation of lists for $\mathbf{G}$, *etc.*

(c) `vinit()` : Input the potential data.

(d) `iniewd()` : Initialization of parameters for the Ewald calculation.

2. The optimization loop.

(a) `getll(etot,nloop)` : The electronic structure determination process.

(b) `charge(0)` : The charge density determination step.

(c) `force(vcell)` : The force determination step.

(d) `eforce(natm_g,stoa,grd,frc,fn)` : An optimization of `force` for a special usage.

(e) `wfile(etot,its), wfile0(999,its), wfile0(000,its)` : The output routines.

(f) `mvatm(delta)` : The routine to obtain new atomic coordinates.

9.5 Subroutine for diagonalization

In the subroutine `diagw`, the CG method is applied to obtain the electronic structure.

```
      subroutine diagw (nloop,vcell,alp,etot)

      use parameters
      use globalarray

      implicit none

      include 'commons/common.h'

      integer :: nloop, inw, itmax, ik, in, ig, ist, loop, its, nwf, nnn, &
           istc, i

      real(8) ::  etot(8), vcell, alp, eps, ang, d1, d2, dnf, step, ep, e_p,&
           fret, fp, gd, avm, dmm, t1, t2, dsecnd,e2,e1

      complex(8) :: ggc,gam,s, chg1

      parameter (eps = 1d-10)
      parameter (inw = 2)
      parameter (e_p = 5d0)

! -----< calc. charge density for G-space >-----
      call schmit(inw)
```

```
      call toreal(inw)

      chg1 = chgg_g(1)
      etot(6)=alp*dreal(chg1)

! -----< calc. total energy(fp) >-----

      call etotal (vcell,inw,etot)

      fp = etot(8)

! -----< calc. force >-----

      call wforce(vcell,inw)

      call precnd(e_p)

      call schmi2(inw,1)

      itmax = max(itmaxd*2,npw_g*mxband_g*nk_g*4)

! nwf(= npw*nband*nk) is num. of wave func.

! -----< store old data of w.f. >-----

      do in = 1, mxband_g
         do ik = 1, nk_g
            do ig = 1, mxpwk(ik)
               wvfn_g(ig,in,ik,4)=wvfn_g(ig,in,ik,1)
               wvfn_g(ig,in,ik,5)=wvfn_g(ig,in,ik,1)
            end do
         end do
      end do

! -----< iteration start >-----

      ist = 1
      istc = 0
      loop = 0
! Main loop starts
      do its = 1, itmax

      s = (0d0,0d0)
      d1= 0d0
      d2= 0d0

! -----< summation of some value >-----
      nwf = mxband_g*nk_g
```

```
      do in = 1,mxband_g
         do ik = 1,nk_g
            do ig = 1,mxpwk(ik)

               s = s + dconjg(wvfn_g(ig,in,ik,1))*wvfn_g(ig,in,ik,2)
               d1= d1+dreal(wvfn_g(ig,in,ik,1))**2+dimag(wvfn_g(ig,in,ik,1))**2
               d2= d2+dreal(wvfn_g(ig,in,ik,2))**2+dimag(wvfn_g(ig,in,ik,2))**2

            end do
         end do
      end do

      ang = acos(cdabs(s)/sqrt(d1*d2))/pi
         dnf = sqrt( d1/nwf )

      if(ang.lt.0.46d0.and.ang.ge.0.385d0) then
         call schmi2(inw,2)
      elseif(ang.lt.0.385d0)then
         call schmi2(inw,1)
      endif

      call linmin(vcell,etot,fret,ist)

      if(ist.eq.0) istc=istc+1

      step=2d0*dabs(fret-fp)/(dabs(fret)+dabs(fp)+eps)

!  ----------< judgement for convergency >----------

      if((step.le.ftol.and.ist.eq.1).or.its.gt.itmaxd.or.istc.gt.3) then

       loop = loop + 1
       if(loop.gt.2) then

          etot(6)=alp*dreal(chg1)
          call efile(etot,ang,dnf,its,ftol,step,loop)
          nloop = its
          write(*,*) 'Count (its,loop)', its,loop
         return

       else

          ist = 0

       endif
      endif

!  -------------------------------------------------
```

```
ep=fret-fp

fp = fret
call wforce(vcell,inw)
call precnd(e_p)

call eigenv(avm,dmm,its,nnn)

if(ist.eq.0.or.nnn.eq.1) then
call toreal(inw)
chg1 = chgg_g(1)
call wforce(vcell,inw)
call precnd(e_p)
endif

if(its.eq.0) ist = 0
if(ist.eq.1) then
   gd = 0d0
   ggc = (0d0,0d0)

   do in = 1, mxband_g
      do ik = 1, nk_g
         do ig = 1, mxpwk(ik)
            gd = gd + dreal(wvfn_g(ig,in,ik,4))**2 &
                 + dimag(wvfn_g(ig,in,ik,4))**2
            ggc = ggc + dconjg(wvfn_g(ig,in,ik,1) &
                 - wvfn_g(ig,in,ik,4))*wvfn_g(ig,in,ik,1)
         end do
      end do
   end do

   if(gd.eq.0d0) then
      write(*,*)'warning. really gd = 0 ?'
      return
   endif

   gam = ggc/gd

   do in = 1, mxband_g
      do ik = 1, nk_g
         do ig = 1, mxpwk(ik)
            wvfn_g(ig,in,ik,4)=wvfn_g(ig,in,ik,1)
         end do
      end do
   end do

   do in = 1, mxband_g
      do ik = 1, nk_g
         do ig = 1, mxpwk(ik)
```

```
                  wvfn_g(ig,in,ik,5)=wvfn_g(ig,in,ik,4) &
                       + gam*wvfn_g(ig,in,ik,5)
               end do
            end do
         end do

         do in = 1, mxband_g
            do ik = 1, nk_g
               do ig = 1, mxpwk(ik)
                  wvfn_g(ig,in,ik,1)=wvfn_g(ig,in,ik,5)
               end do
            end do
         end do

      elseif(ist.eq.0) then

         do in = 1,mxband_g
            do ik = 1,nk_g
               do ig = 1,mxpwk(ik)
                  wvfn_g(ig,in,ik,4)=wvfn_g(ig,in,ik,1)
                  wvfn_g(ig,in,ik,5)=wvfn_g(ig,in,ik,1)
               end do
            end do
         end do

         ist = 1

      endif

      nnn = mod(its,intefl)

      if(nnn.eq.0) then
         if(ideb.eq.1) write(*,*) 'efile in'
         call efile(etot,ang,dnf,its,ftol,step,loop)
         if(ideb.eq.1) write(*,*) 'efile out'
      endif

      end do
! Main loop ends

      stop 'Maximum iterations exceeded (diagw).'

      end subroutine diagw
```