Development of a plane-wave basis pseudopotential code

Byoungseon Jeon

Department of Applied Science, University of California, Davis, CA 95616

(Dated: February 26, 2010)

Contents

INTRODUCTION

Among the approaches of density functional theory, a plane wave basis pseudopotential method is favored by the simplicity and its computational efficiency, with relatively high accuracy [1, 2]. There have been many variations and the most accepted method, PAW style, is quite different from the standard one, which will be shown below. However, standard implementation can server as a reference for future work and I leave the detailed description.

For the programming, we use C/C++, employing g++ compiler from GCC. External libraries will be employed extensively for FFT and eigen-value evalualtion, using FFTW and LAPACK.

IMPLEMENTATION OF PLANE WAVE BASIS PSEUDOPOTENTIAL METHOD

In most of texts and paper, actual imlementation is not clear, because many experts focus on only small components, assuming overall loop is clear to readers. Through the despair and frustration which I had, I realized that certain comment on the whole procedure of electronic structure calculation will benefit any beginner, for *ab initio* study. Even though I will mostly describe the actual implementation, but still theoretical background is very important. Please be familiar with the theory before the implementation.

DFT and Born-Oppenheimer approximation

Main bottleneck of electronic structure calculation comes from the many body interactions of electrons. Density Functional Theory (DFT) [3] is a kind of mean-field theory, and we can replace the many body (electron) interactions with the problem of a single continuum. Kohn-Sham (KS) Ansatz [4] enables us to solve the ground state electron density as homogeneous electron gas.

As another important approximation, we can assume that the electrons do not transit from ground states. This is the adiabatic approximation, proposed by Born and Oppenheimer [2]. Because of huge mass ratio, the wave functions of electrons follow the nuclear dynamics, staying at same eigen states. Based on the Born-Oppenheimer approximation, we can decouple the Hamiltonian into several components. The corresponding Hamiltonian is given as:

$$\begin{aligned} H_{\mathrm{KS}}(\rho) \;=\; & -\sum_i \frac{\hbar^2}{2m}\nabla_i^2 + e^2\sum_i\sum_{j>i}\frac{1}{|r_i - r_j|} - e^2\sum_I\sum_i\frac{Z_I}{|R_I - r_i|} \\ & + e^2\sum_I\sum_{J>I}\frac{Z_I Z_J}{|R_I - R_J|} - \sum_I \frac{\hbar^2}{2M}\nabla_I^2 \end{aligned} \tag{1}$$

The first component is the kinetic term of the electron cloud and the second one is the electron-electron interaction. The third component is for the interaction between electrons and ions and the fourth one is the ion-ion interaction. The last one is the dynamics of ions.

Employing the electron probability density as the argument of the Hamiltonian, the Eq. (1) can be rewritten. We take only the electronic structure into account, and skip the ion-ion interaction and ion dynamics, here.

We define the electron probability density as:

$$\rho(r) = 2\sum_i |\psi(r)|^2 \tag{2}$$

The kinetic energy of non-interacting electron is:

$$T = -\frac{\hbar^2}{2m}\sum_i f_i \langle\psi_i|\nabla^2|\psi_i\rangle \tag{3}$$

Then Kohn-Sham functional can be written as:

$$E_{\mathrm{KS}}[\rho] = T[\rho] + \int \rho(r) v_{\mathrm{ext}}(r) dr + \frac{1}{2}\int\int \frac{\rho(r)\rho(r')}{|r-r'|} dr dr' + E_{\mathrm{xc}}[\rho] \tag{4}$$

Now functional is available and we know that how the total energy of Kohn-Sham formulation is built. Then how we implement this, in order to solve the problem? Before the implementation, we have to decide what basis we are going to use - there are many choices, such as plane wave, local wave, and even purely numerical. As mentioned above, we are using plane wave, which will yield overlapping integration as unit matrix. Anyway, please keep in mind - each basis has its own pro and con. Make sure why we are using plane wave basis.

Plane-wave basis

For periodic solids or crystals, the same unit-cell is repeated infinitely and plane wave basis facilitates this feature with Bloch functions.

$$\psi_{\mathbf{k}}(\mathbf{r}) = u_{\mathbf{k}}(\mathbf{r}) \exp(i\mathbf{k}\cdot\mathbf{r}) \tag{5}$$

where $u_{\mathbf{k}}(\mathbf{r})$ is periodic in the crystal lattice [5]. This means that the periodic potential and its eigenfunctions can be described using the plane wave ($\exp(i\mathbf{k}\cdot\mathbf{r})$) basis.

Plane waves are expressed as linear combination of reciprocal lattice vectors, with integer multiples, such as:

$$\vec{v} = n_1\vec{a} + n_2\vec{b} + n_3\vec{c} \tag{6}$$

For actual implementation, mostly we use those integer constants to describe each plane waves, because we know the reciprocal lattice vectors from the beginning, and this will simplify the coding and impose less burden on computing - integer number is cheaper than

floating number. Mostly the number of plane waves are defined by the cut-off energy, which implies the upper limit of kinetic energy of plane waves.

$$\frac{\hbar^2}{2m_e}|\mathbf{k}+\mathbf{G}|^2 < E_{\text{cutoff}} \tag{7}$$

In hartree unit, most of the constants will be one ($m_e = \hbar = 1$) and we can calculate the formula very easily. Correspondong plane waves inside of cutoff are found as:

```
for (i=-n;i<=n;i++) {
   for (j=-n;j<=n;j++) {
      for (k=-n;k<=n;k++) {
         km2 = mag_km2(param, ((double) i), ((double) j), ((double) k), m);
         if ( km2 < Gcut2 ) {
            kmax = std::max(km2,kmax);
            pw_el[m][0].push_back(i);
            pw_el[m][1].push_back(j);
            pw_el[m][2].push_back(k);
            pw_el[m][3].push_back(0);
            npw_++;
          }
      }
   }
}
```

Assuming certain huge grid sets, we check every index (i,j,k) of reciprocal lattice vectors and find any wave vector which is less than the cut-off energy (=`Gcut2`).

Electron-electron interaction

Electron-electron interaction is a double integral in the real space and it is very hard to solve. But mapping to reciprocal space facilitates this as a Poisson equation.

$$E_{\text{H}} = \int\int \frac{\rho(r)\rho(r')}{|r-r'|}drdr' \tag{8}$$

Hartree (electron-electron) potential is represented as a Poisson equation:

$$V_{\text{H}}(\mathbf{G}) = \frac{4\pi\rho(G)}{|G|^2} \tag{9}$$

When the coefficient of the wave functions are found, the corresponding Hartree energy is calculated as:

$$E_{\text{H}} = \sum \frac{2\pi\Omega\rho(G)^2}{|G|^2} \tag{10}$$

The mapping is done by Parseval's theorem.

$$\int\int \frac{\rho(r)\rho(r')}{|r-r'|}drdr' = \int dr\rho(r)\int dr'\frac{\rho(r')}{|r-r'|} \quad (11)$$

$$= \int dr\rho(r)\sum_{G,G'}\int \rho(G)e^{-iGr'}\frac{e^{iG'|r-r'|}}{G'^2}dr' \quad (12)$$

From the Parseval's theorem, $\rho(r)$ should be employed as conjugate, but it has only real component. For $r > r'$,

$$\int dr\rho(r)\sum_{G,G'}\int \rho(G)e^{-iGr'}\frac{e^{iG'(r-r')}}{G'^2}dr' = \sum_{G,G'}\frac{\rho(G)}{G'^2}\int dr\rho(r)e^{iG'r}\int dr'e^{-i(G+G')r'} \quad (13)$$

$$= \sum_{G,G'}\frac{\rho(G)\rho(G')}{G'^2}\Omega\delta_{G,-G'} \quad (14)$$

For $r < r'$,

$$\int dr\rho(r)\sum_{G,G'}\int \rho(G)e^{-iGr'}\frac{e^{iG'(r'-r)}}{G'^2}dr' = \sum_{G,G'}\frac{\rho(G)}{G'^2}\int dr\rho(r)e^{-iG'r}\int dr'e^{-i(G-G')r'} \quad (15)$$

$$= \sum_{G,G'}\frac{\rho(G)\rho(-G')}{G'^2}\Omega\delta_{G,G'} \quad (16)$$

Eq. (14) and (16) are equivalent each other. Using the orthogonality and the transforming coefficient, we can reach Eq. (10).

Sample code can be implemented as:

```
for (i=0;i<Npw;i++) {
  ix = pw[i][0]; iy = pw[i][1]; iz = pw[i][2];
  for (j=0;j<Npw;j++) {
    jx = pw[j][0]; jy = pw[j][1]; jz = pw[j][2];
    kx = ix - jx; ky = iy - jy; kz = iz - jz;
    if (i != j) {
      G2 = FOURPISQ * (double) (kx*kx + ky*ky + kz*kz) / (l_box*l_box);
      kx = (kx + Ngr)%Ngr; ky = (ky + Ngr)%Ngr; kz = (kz + Ngr)%Ngr;
      id = kz + Ngr*(ky + Ngr*kx);
      Hht[i][j][0] = 4.*PI*nk[id][0]/G2;
      Hht[i][j][1] = 4.*PI*nk[id][1]/G2;
      Eht += 0.5*2.*PI*omega*(nk[id][0]*nk[id][0] + nk[id][1]*nk[id][1])/G2;
    }
    else {
      Hht[i][j][0] = Hht[i][j][1] = 0.0;
    }
  }
}
```

`pw` contains the components of each plane wave, and `kx,ky,kz` will be the difference of two plane waves. `nk` is the reciprocal charge density and the calculation is very straightforward. Even though we employed $N_{\rm pw} \times N_{\rm pw}$ matrix, but this is not recommended, because some components will be redundant. We are using the difference of two plane-waves, and this operation yields many pairs of same value, such as (2,1,1,) - (1,1,1) = (1,0,0) and (-1,3,3) - (-2,3,3) = (1,0,0), in terms of plane wave index. If we employ a method based on FFT grids, then the implementation will be:

```
VH[0][0] = VH[0][1] = sys.E_HT = 0.0;
for (i=1;i<ns.ngrid3;i++) {
    double gs = G2[3][i]/FOURPISQ;
    VH[i][0] = 4.*PI*nk[i][0]/G2[3][i];
    VH[i][1] = 4.*PI*nk[i][1]/G2[3][i];
    sys.E_HT += 2.*PI*param.omega*
                (nk[i][0]*nk[i][0] + nk[i][1]*nk[i][1])/G2[3][i];
}
```

Grids and its implementation

Now this is a good point to talk about grid implementation. Basically, we don't need grid implementation for band structure code. Everything can be done using $N_{\rm pw} \times N_{\rm pw}$ matrix - but it is extemely expensive. Grid imlementation facilitate the implementation and spare computing resource for most of the Hamiltonian components, even some elements like nonlocal pseudopotential or exact exchange can not fit into grids though.

Then, why we use grids? As done above, let's assume some plane wave sets, such as (3,2,1), (2,1,0), and (1,0,-1). Each element (ij) of $N_{\rm pw} \times N_{\rm pw}$ Hamiltonian matrix is the effect of plane wave $i-j$. Then (3,2,1) - (2,1,0) = (1,1,1). Again, (2,1,0) - (1,0,-1) = (1,1,1). Now what?

If we build $N_{\rm pw} \times N_{\rm pw}$ as brute force, we are repeating some of elements, and it is futile. To save computing load and memory, we can assign this **subtraction** of two plane-waves into **grid** information. Sample code is shown below:

```
for (i=0;i<Npw;i++) {
  ix = pw[i][0]; iy = pw[i][1]; iz = pw[i][2];
  for (j=0;j<Npw;j++) {
    jx = pw[j][0]; jy = pw[j][1]; jz = pw[j][2];
    kx = ix - jx;kx = (kx+Ngr)%Ngr;
    ky = iy - jy;ky = (ky+Ngr)%Ngr;
```

```
kz = iz - jz;kz = (kz+Ngr)%Ngr;

id = kz + Ngr*(ky + Ngr*kx);
```

`kx` is the difference of two plane waves. If it is negative, then we round it - reminder: we are using periodic boundary condition - for 3D grid system, we can allocate certain id number of each grid point, using the (i,j,k) of grid position.

Now we know grid implementation will be beneficial to build Hamiltonian matrix. Then here we have one question. How big grids are required?

As shown in Fig. 2, we will begin with some group of plane waves. When we calculate the charge density, the circle (sphere in 3D) will grow twice - why? with two plane wave components (1,0,0) and (-1,0,0), charge density component can be built at (1,0,0) - (-1,0,0) = (2,0,0). We are employing the subtraction of two plane wave components, including negative value, the charge density components will grow up. Again, we are using the periodic boundary condition, at the charge density at negative grids will be rounded up into right side of grids. To prevent grid conflict, the grid size should be larger than the amount of rounded. Let $N_{\mathrm{grid-max}}$ is the maximum plane wave index at certain coordinate. Finally the grid size at the coordinate should be larger than $2N_{\mathrm{grid-max}} + 1$.

Here I want to mention several feature of the grids.

- The minimum grid size is given as $2N_{\mathrm{grid-max}} + 1$ but mostly we do not use the exact value. FFT is optimized for the power of 2,3,5 and others, depending on the library/solver, we tune the value. If the minimum size is calculated as 61, then 64 will yield much better performance.

- Excluding the charge density (or even plane wave) circle (sphere in 3D) from the grids in the simulation box, the grid components around center have zero, in real and imaginary value. You may think the grid method is wasting lots of memory, even we don't need them. But when we tranform the grid into real space, using FFT, then all grids will have certain value.

- Basically the purpose of the grid method is not from $N_{\mathrm{pw}} \times N_{\mathrm{pw}}$ matrix usage. It is from charge density calculation. In reciprocal space, charge density is calculated as:

$$\rho(\mathbf{G} - \mathbf{G}') = |\mathbf{G}\rangle\langle\mathbf{G}'| \tag{17}$$

which is N_{pw}^2 calculation. But if we use appropriate mapping as

$$\langle \mathbf{r}|\rho(\mathbf{G}-\mathbf{G}')|\mathbf{r}'\rangle = \langle \mathbf{r}|\mathbf{G}\rangle\langle \mathbf{G}'|\mathbf{r}'\rangle = \beta\beta^* \tag{18}$$

it is linear order now. But FFT $\beta = \langle \mathbf{r}|\mathbf{G}\rangle$ takes $N_{\mathrm{pw}} \log N_{\mathrm{pw}}$ and this is the dominant computing load.

Pseudopotential

To reduce the degree of the freedoms of the problem, we simplify the all-electron interactions and an ion as valence electrons and a pseudized ion. For the valence electron interaction problems, the effect from the core electrons could be neglected and we can merge them into the ion core, making a pseudo-ion. Then the potential field between the valence electrons and the ion should be rebuilt and the pseudopotential is required. There are tons of materials about pseudopotential generation and its theoretical background but we skip them because they are beyond our work.

Shortly, we use GTH [6]/HGH [7] dual space separable pseudopotential. They provide analytic functional form for real/reciprocal space respectively and are implemented simply.

Ionic local potential is a function of an error function

$$V_{\mathrm{loc}}(r) = -\frac{Z_{\mathrm{ion}}\mathrm{erf}(r/r_{\mathrm{loc}})}{r}. \tag{19}$$

Transforming to reciprocal space, we can find the expression with wave vector, using the similar approach from Chapter G.5 of Ref. [8].

$$\begin{aligned}
\int \frac{\mathrm{erf}(\alpha r)}{r} e^{i\mathbf{G}\mathbf{r}} d\mathbf{r} &= \int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} r^2 \frac{\mathrm{erf}(\alpha r)}{r} e^{iGr\cos\theta} dr d\theta d\phi && (20)\\
&= 2\pi \int_0^{\infty}\int_{-1}^{1} r\,\mathrm{erf}(\alpha r) e^{iGrw} dw dr && (21)\\
&= \frac{2\pi}{iG}\int_0^{\infty} \mathrm{erf}(\alpha r)\Big\{e^{iGr} - e^{-iGr}\Big\} dr && (22)
\end{aligned}$$

Using integration by parts and the derivative of error function,

$$\frac{\partial \mathrm{erf}(r)}{\partial r} = \frac{2}{\sqrt{\pi}} e^{-r^2} \tag{23}$$

$$\begin{aligned}
\int \frac{\mathrm{erf}(\alpha r)}{r} e^{i\mathbf{G}\mathbf{r}} d\mathbf{r} &= \frac{4\sqrt{\pi}\alpha}{G^2}\Big\{\int_0^{\infty} e^{iGr} e^{-\alpha^2 r^2} dr + \int_0^{\infty} e^{-iGr} e^{-\alpha^2 r^2} dr\Big\} && (24)\\
&= \frac{4\sqrt{\pi}\alpha}{G^2}\int_{-\infty}^{\infty} e^{iGr} e^{-\alpha^2 r^2} dr && (25)\\
&= \frac{4\pi}{G^2} e^{-G^2/4\alpha^2} && (26)
\end{aligned}$$

Other local pseudopotential at real space is the polynomial by Gaussian functions.

$$V_{\rm loc}(r) = \exp\Big(-\frac{1}{2}(r/r_{\rm loc})^2\Big)\Big[C_1 + C_2(r/r_{\rm loc})^2 + \dots\Big] \tag{27}$$

With Fourier transform, the pseudopotential at reciprocal space is

$$\begin{aligned}\int \exp\Big(-\frac{1}{2}(r/r_{\rm loc})^2\Big)\exp(i\mathbf{G}\mathbf{r})d\mathbf{r} &= \int \exp\Big(-\frac{1}{2r_{\rm loc}^2}(r - ir_{\rm loc}^2\mathbf{G})^2 - r_{\rm loc}^2G^2/2\Big)d\mathbf{r} &(28)\\ &= \sqrt{8\pi^3}\frac{r_{\rm loc}^3}{\Omega}\exp\Big(-r_{\rm loc}^2G^2/2\Big) &(29)\end{aligned}$$

Using that $x^n f(x)$ is transformed into $i^n d^n F(G)/dG^n$,

$$\begin{aligned}\int \exp\Big(-\frac{1}{2}(r/r_{\rm loc})^2\Big)\Big(\frac{r}{r_{\rm loc}}\Big)^2\exp(i\mathbf{G}\mathbf{r})d\mathbf{r} &= -\frac{d^2}{dG^2}\sqrt{8\pi^3}\frac{r_{\rm loc}}{\Omega}\exp\Big(-r_{\rm loc}^2G^2/2\Big) &(30)\\ &= -\frac{d}{dG}\sqrt{8\pi^3}\frac{r_{\rm loc}}{\Omega}\exp\Big(-r_{\rm loc}^2G^2/2\Big)(-r_{\rm loc}^2\mathbf{G}) &(31)\\ &= \sqrt{8\pi^3}\frac{r_{\rm loc}^3}{\Omega}\exp\Big(-r_{\rm loc}^2G^2/2\Big)(3 - r_{\rm loc}^2\mathbf{G}^2) &(32)\end{aligned}$$

Here, $\frac{\partial \mathbf{G}}{\partial G} = 3$.

Using the above expression, we can build $N_{\rm pw} \times N_{\rm pw}$ matrix as:

```
for (i=0;i<Npw;i++) {
  ix = pw[i][0]; iy = pw[i][1]; iz = pw[i][2];
  for (j=0;j<Npw;j++) {
    jx = pw[j][0]; jy = pw[j][1]; jz = pw[j][2];
    kx = ix - jx; ky = iy - jy; kz = iz - jz;
    if (i != j) {
      G2 = FOURPISQ * (double) (kx*kx + ky*ky + kz*kz) / (l_box*l_box);
      kx = (kx + Ngr)%Ngr; ky = (ky + Ngr)%Ngr; kz = (kz + Ngr)%Ngr;
      id = kz + Ngr*(ky + Ngr*kx);
      A = exp(-G2*rs*rs*0.5);
      Hloc[i][j][0] = pow(2.*PI,1.5)*pow(rs,3.)*A*
        (c1 + c2*(3.-G2*rs*rs))/omega - 4.*PI*A/G2/omega;
      Hloc[i][j][1] = 0.0;
      Eloc += 0.5*omega*(Hloc[i][j][0]*nk[id][0] + Hloc[i][j][1]*nk[id][1]);
    }
    else {
      Hloc[i][j][0] = Hloc[i][j][1] = 0.0;
    }
  }
}
```

However, as done in Hartree matrix, this $N_{\rm pw} \times N_{\rm pw}$ matrix is not welcomed due to repeating of same elements.

Exchange-correlation terms are also calculated using their fittings, which is based on Local Density Approximation (LDA) [9]. The pseudopotential terms are constant during the self-consistent loop. After the loop reaches the convergence, the corresponding local energy is calculated as:

$$E_{\text{local}} = \Omega \sum_I \sum V_{\text{I,local}}(G) S_I(G) \rho(G) \tag{33}$$

Non-local energy is obtained as:

$$E_{\text{nl}} = \sum \sum_{ij} \langle \psi | p_i^l \rangle h_{ij} \langle p_j^l | \psi \rangle \tag{34}$$

Exchange-Correlation of electron-electron interaction

Unlike Linear Combinations of Atomic Orbitals (LCAO), plane wave method lacks in the description of the exact exchange energy of the electron-electron interaction. Besides, additional description of the correlation term is necessary. To compensate these components, many methods have been developed, such as LDA and GGA. We employ LDA with GTH/HGH pseudopotential, and the potential is described as:

$$V_{\text{xc}}(\mathbf{r}) = \left[\epsilon_{\text{xc}} + n \frac{\partial \epsilon_{\text{xc}}}{\partial n} \right] \tag{35}$$

ϵ_{xc} is given as a rational polynomial.

$$\epsilon_{\text{xc}} = -\frac{a_0 + a_1 r_s + a_2 r_s^2 + a_3 r_s^3}{b_1 r_s + b_2 r_s^2 + b_3 r_s^3 + b_4 r_s^4} \tag{36}$$

r_s is the Wigner-Seitz radius of the local electron probability density. The coefficients can be found at the Ref. [6]. The exchange-correlation energy is calculated as:

$$E_{\text{xc}} = \int \epsilon_{\text{xc}} \rho(r) dr \tag{37}$$

```
for (i=0;i<ns.ngrid3;i++) {
  if (nr[i][0] > 0.0) {
    rs = pow(3./4./PI/nr[i][0], 1./3.);
    asum  = a[0] +    a[1]*rs +    a[2]*rs*rs +    a[3]*rs*rs*rs;
    aasum = a[1] + 2.*a[2]*rs + 3.*a[3]*rs*rs;
    bsum  = b[0]*rs + b[1]*rs*rs + b[2]*rs*rs*rs + b[3]*pow(rs,4.0);
    bbsum = b[0] + 2.*b[1]*rs + 3.*b[2]*rs*rs + 4.*b[3]*rs*rs*rs;
    exc = -asum/bsum;
```

```
    ndedn = rs*(aasum*bsum - asum*bbsum)/bsum/bsum/3.;
    Wxc[i][0] = exc + ndedn; Wxc[i][1] = 0.0;
    sys.E_xc += exc*nr[i][0]*dv;
  }
}
```

`Wxc` becomes the Hamiltonian matrix in real space. For reciprocal space component, we transfer with FFT.

Building Hamiltonian

Now we know all the main components of Kohn-Sham energy: electron-electron, electron-ion, and exchange-correlation (ion-ion is handled by Ewald sum, as done in the classical MD). Then how we build Hamiltonian? Especially, what the hell are we doing with those components?

Basic idea is variation - we know the total energy of Kohn-Sham formulation. This energy should be minium, by ground state. Therefore, the gradient of the total energy should be zero. Then here we have a question - gradient by what?

The answer is the electron density (or wave function). We build Hamiltonian components as the partial derivation of total energy in terms of electron density (or wave function). Double-check that the partial derivation of Eq. 10 is Eq. 9.

Once we know wave functions, each Hamiltonian can be calculated. Summing all the Hamiltonian components, we finally build Hamiltonian matrix, which will be $N_{\text{pw}} \times N_{\text{pw}}$. Solving this matrix, we will have eigen-values (=band energies) and eigen-vectors (new wave functions). By inherent characteristics of Schrödinger equation, we will repeat this procedure until the criterion is met. This is called self-consistent loop.

Self-consistent loop and Fourier Transform

Born-Oppenheimer (adiabatic) approximation enables us to solve quantum equations with classical description of electrons and ions. The Hamiltonian of electronic structure enables us to solve the eigen states of electrons but it is an implicit form. To solve this implicit equation, we employ self-consistent loop. The initial wave functions are configured as random numbers or the orbital data of LCAO. Solving the given Hamiltonian, we update

the eigenstate of the electrons, which will be the input of next loop. Repeating this loop, we can check the convergence of the system and finally reach the ground state of the electronic structure.

The electron probability density is calculated by the square of wave functions but its calculation order is N^2 and computing load is very intensive. To avoid this problem, we map electron wave functions into real space. Then electron probability density at real space is a linear sum of real space electron wave functions.

$$\rho(\mathbf{G}) = |\psi\rangle\langle\psi| \tag{38}$$

$$\rho(\mathbf{r}) = \langle\mathbf{r}|\psi\rangle\langle\psi|\mathbf{r}\rangle = (\langle\psi|\mathbf{r}\rangle)^*\langle\psi|\mathbf{r}\rangle \tag{39}$$

The mapping between real and reciprocal space is handled by FFT (Fast Fourier Transform), which has $N\log N$ computing order. Once real space electron probability is achieved, it will be remapped into reciprocal space using FFT again.

When the density is updated, we use mixing - interpolating the new and the old density, we can guarantee the convergence through the self consistent loop. Broyden [10–13] mixing provides better convergence rate than linear mixing, and it is favored in many built-in codes.

The illustration of the self-consistent loop is shown in Fig. 1.

Broyden mixing

A simple way to solve the implicit equation is to use self-consistent loop. During the iteration, input density is mixed with the previous output density, stabilizing the convergence. For mixing, linear mixing provides the simplest approach. However, the convergence might be quite slow and even it might not reach convergent plateau by the accumulation of round-off error.

$$\rho_{\text{in}}^m = (1-\alpha)\rho_{\text{out}}^{m-1} + \alpha\rho_{\text{in}}^{m-1} \tag{40}$$

Mostly α is favored as < 0.3 or even much smaller (~ 0.001).

To overcome the convergence speed, multi-dimensional Newton Raphson method has been developed and implemented [10].

Let's define a functional

$$\mathbf{F}(\mu) = \nu - \mu = 0 \tag{41}$$

By Taylor series expansion,

$$\mathbf{F}(\mu) \approx \mathbf{F}(\mu^m) + \frac{\partial \mathbf{F}}{\partial \mu}(\mu^m)(\mu - \mu^m) + \text{H.O.T} \tag{42}$$

Here we define a Jacobian matrix,

$$\frac{\partial \mathbf{F}}{\partial \mu}(\mu^m) = \mathbf{J}^m \tag{43}$$

From equation (41), we know the following equation is valid.

$$\frac{\partial \mathbf{F}}{\partial \mu}(\mu^m) + \mathbf{J}^m(\mu - \mu^m) \approx 0 \tag{44}$$

Then we can solve the equation by

$$\mu = \mu^m - (\mathbf{J}^m)^{-1}\mathbf{F}(\mu^m) \tag{45}$$

But solving the inverse matrix is really pain, even the size of vectors is the number of employed plane waves ($\sim O(3-4)$). Avoiding this pain, we employ iterations.

Jacobian matrix can be approximated as:

$$\mathbf{J}^m = \frac{\partial \mathbf{F}}{\partial \mu}(\mu^m) \approx \frac{\Delta \mathbf{F}^m}{\Delta \mu^m} \tag{46}$$

Then we have the variation relation of the functional and the input.

$$\Delta \mathbf{F}^m = \mathbf{J}^m \Delta \mu^m \tag{47}$$

Each component is updated every iteration. To solve the multi-dimensional Newton Raphson, we suggest the following sets of equations.

As shown above, we have the following constraint.

$$\mathbf{g} = \Delta \mathbf{F}^m - \mathbf{J}^m \Delta \mu^m = 0 \tag{48}$$

Jacobian matrix will be updated too. When we reach the convergence, the difference of Jacobian matrix will be minimized with Frobenious norm [14],

$$Q = |\mathbf{J}^m - \mathbf{J}^{m-1}|^2 \tag{49}$$

Then we configure new functional with Lagrangian multiplier.

$$K = Q + \lambda^{\mathrm{T}}\mathbf{g} = Q + \sum_i \lambda_i g_i \tag{50}$$

The right-hand side is for vector component notation. To find the minimum, we differentiate with $\mathbf{J}^m$.

$$\frac{\partial K}{\partial \mathbf{J}^m} = 2(\mathbf{J}^m - \mathbf{J}^{m-1}) - \lambda^{\mathrm{T}}\Delta\mu^m = 0 \tag{51}$$

Using the equation (46),

$$\frac{\Delta\mathbf{F}^m}{\Delta\mu^m} = \mathbf{J}^{m-1} + \frac{\lambda^{\mathrm{T}}\Delta\mu^m}{2} \tag{52}$$

Then the Lagrangian multiplier is calculated as:

$$\lambda_i = \{2\Delta F_i^m - \sum_j 2J_{ij}^{m-1}\Delta\mu_j^m\} / \sum_j (\Delta\mu_j^m)^2 \tag{53}$$

Then

$$\mathbf{J}^m = \mathbf{J}^{m-1} + \frac{[\Delta\mathbf{F}^m - \mathbf{J}^{m-1}\Delta\mu^m](\Delta\mu^m)^{\mathrm{T}}}{\Delta(\mu^m)^{\mathrm{T}}\Delta\mu^m} \tag{54}$$

Let $\mathbf{G}$ be the inverse of $\mathbf{J}$. With Broyden's second method, then the update of $\mathbf{G}$ will be

$$\mathbf{G}^m = \mathbf{G}^{m-1} + \frac{[\Delta\mu^m - \mathbf{G}^{m-1}\Delta\mathbf{F}^m](\Delta\mathbf{F}^m)^{\mathrm{T}}}{(\Delta\mathbf{F}^m)^{\mathrm{T}}\Delta\mathbf{F}^m} \tag{55}$$

As initial guess, we employ a constant matrix such as

$$\mathbf{G}^1 = -\alpha\mathbf{I} \tag{56}$$

α can be chosen as [0.3, 0.8].

We found the gradient of the functional and can employ this formulation for the density of the Hamiltonian.

$$\mathbf{F}^m = \rho_{\mathrm{out}}^m - \rho_{\mathrm{in}}^m \tag{57}$$

while $\mu^m = \rho_{\mathrm{in}}^m$. Finally,

$$\Delta\mathbf{F}^m = \mathbf{F}^m - \mathbf{F}^{m-1} \tag{58}$$

$$\Delta\rho^m = \rho_{\mathrm{in}}^m - \rho_{\mathrm{in}}^{m-1} \tag{59}$$

$$\mathbf{G}^m = \mathbf{G}^{m-1} + \frac{[\Delta\rho^m - \mathbf{G}^{m-1}\Delta\mathbf{F}^m](\Delta\mathbf{F}^m)^{\mathrm{T}}}{(\Delta\mathbf{F}^m)^{\mathrm{T}}\Delta\mathbf{F}^m} \tag{60}$$

$$\rho^{m+1}_{i,\mathrm{in}} = \rho^m_{i,\mathrm{in}} - \sum_j \mathbf{G}^m_{ij}\mathbf{F}^m_j \tag{61}$$

Even though this method accelerates the self-consistent loop, memory burden is not negligible, because of G matrix. To reduce memory load, we can decompose the matrix into several vector components [12, 13]. We followed the summary from Singh et al.

$$\rho^{m+1}_{i,\mathrm{in}} = \rho^m_{i,\mathrm{in}} - \mathbf{G}^1_{ii}\mathbf{F}^m_i - \sum_{j=2}^{m} \mathbf{U}^j_i \mathbf{V}^{T,j}_k \mathbf{F}^m_k \tag{62}$$

$\mathbf{U}$ and $\mathbf{V}$ are vectors of decomposed $\mathbf{G}$ matrix and defined as:

$$\mathbf{U}^i = -\mathbf{G}^1(\mathbf{F}^i - \mathbf{F}^{i-1}) + \rho^i_{\mathrm{in}} - \rho^{i-1}_{\mathrm{in}} - \sum_{j=2}^{i-1} \mathbf{V}^{T,j}(\mathbf{F}^i - \mathbf{F}^{i-1})\mathbf{U}^j \tag{63}$$

$$\mathbf{V}^{T,i} = \frac{(\mathbf{F}^i - \mathbf{F}^{i-1})^T}{(\mathbf{F}^i - \mathbf{F}^{i-1})^T(\mathbf{F}^i - \mathbf{F}^{i-1})} \tag{64}$$

This means that n kinds of $\mathbf{U}$ and $\mathbf{V}$ vectors are required for n self-consistent loops. But the number of self-consistent loops are much smaller than the number of plane waves, mostly, and this method results in the reduction of the required memory and memory access-time.

Finite temperature smearing

At ground state, electrons fill the energy states from the bottom and the top-filled state is called as Fermi surface. Physically, filled/unfilled states are differentiated as one or two/none of electrons. However, this strict rule may result in divergence of the self-consistent loop. For purely numerical stability, we employ finite temperature smearing.

At certain finite temperature condition, top electrons will move into unoccupied state, and the distribution of electrons will smear into smooth curves from unit-step function. Using this scheme, the partial fraction of electrons at each band is described as:

$$f_i = \frac{2}{1 + \exp((E_i - E_{\mathrm{fermi}})/k_B T)} \tag{65}$$

The numerator 2 is for non-spin polarization states. For the spin-polarized state, the numerator becomes 1. The sum of all the fractions should be the number of valence electrons.

Using bisection method, we can find the corresponding E_{fermi} - this is not the exact definition of Fermi energy. It is just for numerical convergence.

```
do {
  fsum = 0;
  for (m=0;m<NKPT;m++) {
    for (i=0;i<NBAND;i++) {
      fsum += param.weight[m]*2./(1. + exp((ns.Eband[i][m] - Efermi)/kbT));
    }
  }
  Ecrit = fsum - Nel;
  if (fsum  > Nel) {
    Emax = Efermi;
    old_E = Efermi;
    Efermi = 0.5*(Efermi + Emin);
  }
  else {
    Emin = Efermi;
    old_E = Efermi;
    Efermi = 0.5*(Efermi + Emax);
  }
} while (fabs(old_E - Efermi) > 1.E-20);
for (m=0;m<NKPT;m++) {
  for (i=0;i<NBAND;i++) {
    frc[i][m] =  2./(1. + exp((ns.Eband[i][m] - Efermi)/kbT));
  }
}
```

As shown above, we begin with arbitrary Fermi energy (`Efermi`) - we may start with the highest band energy. When the total number of electrons is available, we can compare with actual number of electrons then decide to increase or decrease the Fermi energy. Repeating this procedure several times, we can determine an appropriate numerical Fermi energy (this is not physical Fermi energy value), and assign occupation number into every band per k-point.

Special point integration

To find the physical property or the Fermi energy surface, the integration on the Brillouin zone is required. From Bloch wave function, the wave functions are illustrated with plane wave basis. The plane wave can be sampled in the unit-cell (or Brillouin zone) along the K-points. Using this, several special integration point methods have been developed and

Monkhorst-Pack method [15] is one of the popular methods.

Even though this method may require lots of samplings, we can avoid computing cost using the symmetry of periodic crystals. With group theory, we can figure out the symmetry of the system and can reduce the number of required sampling points. However, the implementation of symmetry by the special points in the Irreducible Brillouin Zone (IBZ) requires point symmetry operation when the partial electron probability densities are summed for the total probability density, to cancel out the partial effect from the corresponding special point.

$$\rho_{\text{total}}(\mathbf{G}) = \sum_k f_k \rho_k(\mathbf{G}) \tag{66}$$

$$= f_{k_1}\rho_{k_1}(\mathbf{G}) + f_{k_1}\rho_{k_2}(\mathbf{G}) \ldots f_{k_N}\rho_{k_N}(\mathbf{G}) \tag{67}$$

$$= f_{k_1}\rho_{k_1}(\mathbf{G}) + f_{k_1}\rho_{R_2\cdot k_1}(\mathbf{G}) \ldots f_{k_N}\rho_{R_N\cdot k_N}(\mathbf{G}) \tag{68}$$

$$\neq N f_{k_1}\rho_{k_1}(\mathbf{G}) \tag{69}$$

```
for (iz=0;iz<ns.nfft[2];iz++) {
  for (iy=0;iy<ns.nfft[1];iy++) {
    for (ix=0;ix<ns.nfft[0];ix++) {
      i = ix + ns.nfft[0]*(iy + ns.nfft[1]*iz);
      for (j=0;j<sym_op.nsym;j++) {
        sym_calc(ns, sym_op, j, ix, iy, iz, id, w);
        for (m=0;m<ns.nkpt;m++) {
          for(n=0;n<ns.nband;n++) {
            for (k=0;k<8;k++) {
              id_grid = id[k] + n*ns.ngrid3;
              rout_m[i] += w[k]*coeff*occ.frac[m][n]*occ.weight[m]*
                (pw.cr[m][id_grid][0]*pw.cr[m][id_grid][0] +
                 pw.cr[m][id_grid][1]*pw.cr[m][id_grid][1]);
            }
          }
        }
      }
    }
  }
}
```

Symmetry operation

In band structure code, symmetry plays a very important role. For each problem or simulation cell, an appropriate set of symmetry operation should be figured out and employed. There exit 220 symmetry groups and each group has its own point symmetry operation vectors including non-symmorphic translation vectors. As an example, FCC structure has 48 point symmetry operation vectors and they have to be implemented.

First, for multiple K-point sets, we can figure out **irreducible K-points**, using those symmetry operations. This will reduce the number of required K-points and decrease the required computing load linearly.

Second, when we mix the wave functions to calculate the charge density, symmetry operations should be done. This means that we have to blend the phonon waves from the equilivalent symmetric points. This is the way we have same eigen values/vectors at the equivalent K-points, unless we reduce K-points.

Tetrahedron integration

For special point integration methods, we sum up the value at each special point with its corresponding weight. However, due to convergence issue, mostly this method requires broadening by finite temperature, which invokes artificial energy increase. To avoid the finite temperature implementation and achieve higher accuracy, tetrahedron decomposition method is favored [16–18].

Basically, the Brillouin zone is decomposed into multiples of tetrahedron, while the vertices of each tetrahedron correspond to the integration points. At each band and integration point, an eigenvalue will be available after solving the Hamiltonian. Using linear interpolation (or with Blöchl correction [18]), Fermi energy energy is found and the weight of each integration point is calculated, as shown in Fig. **??**.

$$w_{nj} = \frac{1}{V_G} \int_{V_G} d\mathbf{k} w_j(\mathbf{k}) f(\epsilon_n(\mathbf{k})) \tag{70}$$

Even though the computational cost might be higher but it promises better accuracy than the Monkhorst-Pack integration with smearing.

However, not only the tetrahedron and Monkhorst-Pack method, most of the integration method should be employed with care. If too few points are employed, it can mislead to

the wrong interpretation. As shown in the Fig. **??**, even tetrahedron method may yield misleading results, depending on the splitting orientation. This kind of problem can be overcome by finer mesh, but still care is required.

CONCLUSION

We have implemented the finite element technique to integrate Brillouin zone or unit-cell in the plane-wave basis pseudopotential method, in order to replace the conventional tetrahedron method. The cubic element may not be available in certain Brillouin zone, considering the symmetry. Then tetra or prism element might be employed, while the basic strategy is consistent. Even mixing heterogeneous elements is available and also it is quite common in engineering applications.

Acknowledgments

I appreciate Prof. Juerg Hutter at University of Zurich for valuable help about pseudopotential. Also I thank Prof. Koun Shirai at Osaka University, Prof. Jorge Kohanoff at Queen's University, Belfast, and Prof. Jos Thijssen at Delft University of Technology for valuable advice. This work was carried out partially under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. Also this work was partially supported by NERI-C. The employed source codes can be downloaded at `http://crash.das.ucdavis.edu/~bjeon`.

[1] R. M. Martin, *Electronic Structure: Basic Theory and Practical Methods* (Cambridge, 2004).

[2] J. Kohanoff, *Electronic Structure Calculations for Solids and Molecules: Theory and Computational Methods* (Cambridge, 2006).

[3] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[4] W. Kohn and L. J. Sham, Physical Review **140**, 1133 (1965).

[5] C. Kittel, *Introdunction to Solid State Physics* (John Wiley & Sons, Inc., 1996).

[6] S. Goedecker, M. Teter, and J. Hutter, Phys. Rev. B **54**, 1703 (1996).

[7] C. Hartwigsen, S. Goedecker, and J. Hutter, Phys. Rev. B **58**, 3641 (1998).

[8] E. Kaxiras, *Atomic and Electronic Structure of Solids* (Cambridge, 2003).

[9] D. M. Ceperley and B. J. Alder, Physical Review Letters **45**, 566 (1980).

[10] C. G. Broyden, Mathematics of Computation **19**, 577 (1965).

[11] D. J.E. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Inc., 1983).

[12] G. P. Srivastava, Journal of Physics A: Mathematical and General **17**, L317 (1984).

[13] D. Singh, H. Krakauer, and C. S. Wang, Physical Review B **34**, 8391 (1986).

[14] P. Bendt and A. Zunger, Physical Review B **26**, 3114 (1982).

[15] H. J. Monkhorst and J. D. Pack, Physical Review B **13**, 5188 (1976).

[16] O. Jepson and O. Anderson, Solid State Communications **9**, 1763 (1971).

[17] G. Lehmann and M. Taut, Phys. stat. sol. (b) **54**, 469 (1972).

[18] P. E. Blöchl, O. Jepsen, and O. K. Andersen, Physical Review B **49**, 16223 (1994).

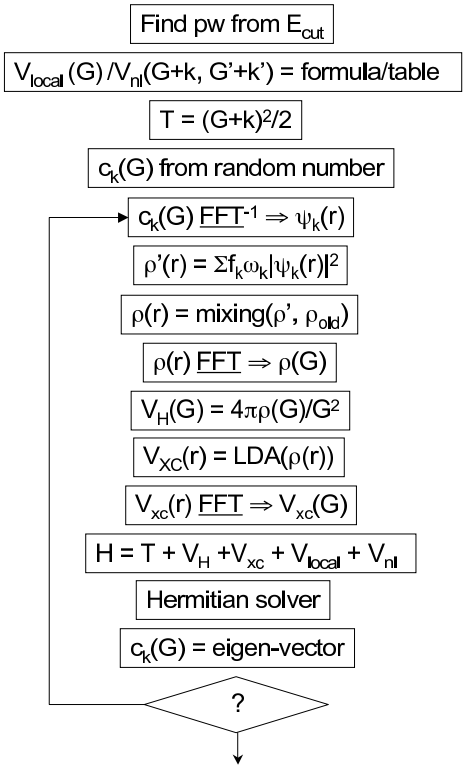

FIG. 1: Flow of electronic structure calculation with plane wave basis pseudopotential method

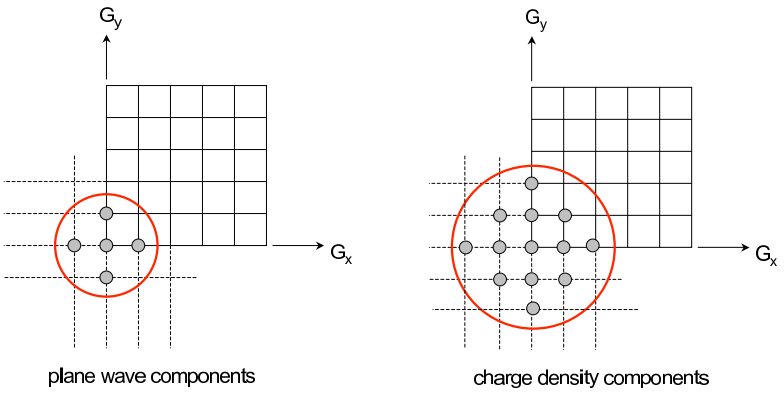

FIG. 2: Plane wave components and charge density components in grids.