OpenMX 朗読会

“The Journey in OpenMX” ver.1.0

Let’s trace the calculation process in OpenMX !!

・Review of equations used in OpenMX

・Decoding of subroutines step by step

by T. Ohwaki (NISSAN Research Center)

Kohn-Sham equaton: $\left[-\frac{\hbar^2\nabla^2}{2m}+\tilde{V}_{\text{eff}}\left(\mathbf{r}\right)\right]\psi_k\left(\mathbf{r}\right)=\varepsilon_k\psi_k\left(\mathbf{r}\right)$

$$\begin{aligned}\tilde{V}_{\text{eff}}\left(\mathbf{r}\right)&=\tilde{V}_{\text{core}}\left(\mathbf{r}\right)+\tilde{V}_{\text{Hartree}}\left(\mathbf{r},\mathbf{r}'\right)+\tilde{V}_{\text{xc}}\left(\mathbf{r}\right)\\&=\tilde{V}_{\text{nonlocal}}\left(\mathbf{r}\right)+\tilde{V}_{\text{na}}\left(\mathbf{r}\right)+\tilde{V}_{\delta\text{Hartree}}\left(\mathbf{r},\mathbf{r}'\right)+\tilde{V}_{\text{xc}}\left(\mathbf{r}\right)\end{aligned}$$ in OpenMX

$\psi_k\left(\mathbf{r}\right)=\sum_{i=1}^{N_{\text{PAO}}}C_{ki}\phi_i\left(\mathbf{r}\right)$: linear expansion of wave functions with basis set (PAO)

$$\mathbf{HC}_k=\varepsilon_k\mathbf{SC}_k \qquad \mathbf{C}_k=\begin{pmatrix}C_{k1}\\ \vdots\\ C_{kN_{\text{PAO}}}\end{pmatrix}$$

Kohn-Sham equation ⇒ Generalized eienvalue problem

The bundle of coefficients and eigenvalue can be expressed in a matrix form:

$$\mathbf{C}_k = \begin{pmatrix} C_{k1} \\ \vdots \\ C_{kN_{\mathrm{PAO}}} \end{pmatrix} \quad \Rightarrow \quad \mathbf{C} = \begin{pmatrix} C_{11} & \cdots & C_{1N_{\mathrm{PAO}}} \\ \vdots & \ddots & \vdots \\ C_{N_{\mathrm{PAO}}1} & \cdots & C_{N_{\mathrm{PAO}}N_{\mathrm{PAO}}} \end{pmatrix}$$

$$\varepsilon_k \quad \Rightarrow \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \varepsilon_{N_{\mathrm{PAO}}} \end{pmatrix}$$

In this case, $\mathbf{HC}_k = \varepsilon_k \mathbf{SC}_k \quad \Rightarrow \quad \mathbf{HC} = \boldsymbol{\varepsilon}\mathbf{SC}$

Matrix elements in equation “**HC = εSC**”

$$S_{ij} = \int \phi_i(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$$ Overlap matrix

$$H_{ij} = T_{ij} + V_{ij}^{\mathrm{na}} + V_{ij}^{\delta\mathrm{H}} + V_{ij}^{\mathrm{xc}} + V_{ij}^{\mathrm{nl}}$$ Hamiltonian matrix

$$T_{ij} = \int \phi_i(\mathbf{r})\frac{-\hbar^2\nabla^2}{2m}\phi_j(\mathbf{r})d\mathbf{r}$$ Kinetic matrix

$$V_{ij}^{\delta\mathrm{H}} = \int \phi_i(\mathbf{r})\tilde{V}_{\delta\mathrm{Hartree}}(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$$

$$= \iint \phi_i(\mathbf{r})\frac{\delta\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\phi_j(\mathbf{r})d\mathbf{r}'d\mathbf{r}$$ Difference Hartree potential matrix

$$V_{ij}^{\mathrm{na}} = \int \phi_i(\mathbf{r})\tilde{V}_{\mathrm{na}}(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$$ Neutral atomic potential matrix

$$V_{ij}^{\mathrm{xc}} = \int \phi_i(\mathbf{r})\tilde{V}_{\mathrm{xc}}(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$$ Exchange-correlation potential matrix

$$V_{ij}^{\mathrm{nl}} = \int \phi_i(\mathbf{r})\tilde{V}_{\mathrm{nonloacl}}(\mathbf{r})\phi_j(\mathbf{r})d\mathbf{r}$$ Nonlocal potential matrix

Main process of calculation in OpenMX

STEP 1: Preparation of basis sets and pseudopotentials

STEP 2: Set of real-space grids

STEP 3: Calculation of matrix elements

STEP 4: Solving generalized eigenvalue problem

STEP 5: SCF calculation（convergence problem）

STEP 6: Force calculation

STEP 7: Calculation of physical quantities

OpenMX

Program	Function	corresponding STEP
openmx.c	Main program	
readfile.c	Reading files	
Input_std.c	Reading input file for atomic structure and calculation conditions	
SetPara_DFT.c	Reading database for basis sets and pseudopotentials	
ReadPara_DFT		1
Read_PAO	Reading basis sets (PAO)	1
Read_VPS	Reading pseudopotentials (VPS)	1
V_Hart_atom	Calculation of atomic Hartree potentials	1
Vna	Calculation of neutral atomic potentials	1
FT_PAO	Fourier transformation of PAOs	1
FT_NLP	Fourier transformation of non-local parts	1
truncation.c	Preparation of real-space grids Storage of relationship between real-space grids and parameters	2
DFT.c	SCF calculation of electronic structure	3，4，5

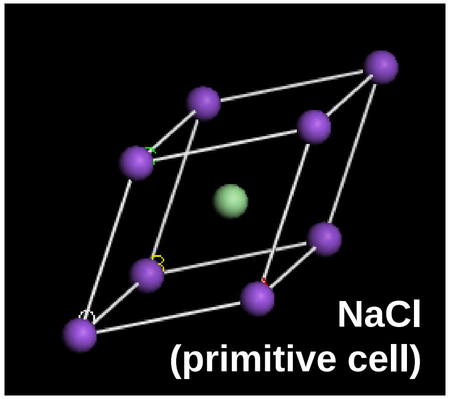
DFT.c (in case of solver = “Cluster”)

SCF calculation			DFT.c	
Hamiltonian matrix (H) elements			Set_OLP_Kin.c	Overlap integrals, kinetic-energy terms
			Set_Nonlocal.c	Non-local terms
			Set_ProExpn_VNA.c	Projector expansion method
SCF	Const. of H	iter. = 1	Set_Aden_Grid.c	Superposition of elec. densities of isolated atoms
			Set_Orbital_Grid.c	Set
			FFT_Density.c	Fourier transformation of electronic densities
			Poisson.c	Hartree potential
			Set_Hamiltonian.c	Hamiltonian matrix
			Set_Vpot.c	neutral atomic, xc, and Hartree potentials
		iter. ≥ 2	Poisson.c	δ–Hartree potentials
			Set_Hamiltonian.c	Hamiltonian matrix
			Set_Vpot.c	neutral atomic, xc, and δ–Hartree potentials
	Generalized eigenvalue problem		Cluster_DFT.c※	HC = εSC (※ in case of solver = “Band”, the subroutine is “Band_DFT_Col.c”)
	Physical quantities		Mulliken_Charge.c Band_DFT_kpath.c, etc.	Mulliken charge Band structure, etc...
	Conv. algorithm	every iter.	(check SCF-convergence)	
		In case of no conv.	Mixing_DM.c	Mixing of previous and present density matrices (DM)
			Set_Density_Grid.c	Calculation of charge-density grid from DM

Preparation of Band Calculation

"NaCl.dat"

```
Atoms.Number              2
Atoms.SpeciesAndCoordinates.Unit    Ang # Ang|AU
<Atoms.SpeciesAndCoordinates            # Unit=Ang.
  1   Na   0.00000  0.00000  0.00000    4.5 4.5
  2   Cl   2.81500  2.81500  2.81500    3.5 3.5
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit              Ang #  Ang|AU
<Atoms.UnitVectors                      # unit=Ang.
  0.00000   2.81500   2.81500
  2.81500   0.00000   2.81500
  2.81500   2.81500   0.00000
Atoms.UnitVectors>


#
# SCF or Electronic System
#

scf.XcType                 LDA         # LDA|LSDA-CA|LSDA-PW
scf.SpinPolarization       off         # On|Off
scf.ElectronicTemperature  300.0       # default=300 (K)
scf.energycutoff           150.0       # default=150 (Ry)
scf.maxIter                170         # default=40
scf.EigenvalueSolver       band        # Recursion|Cluster|Band
scf.Kgrid                  3 3 3       # means 4x4x4
scf.Mixing.Type           rmm-diisk    # Simple|Rmm-Diis|Gr-Pulay
scf.Init.Mixing.Weight     0.010       # default=0.30
scf.Min.Mixing.Weight      0.001       # default=0.001
scf.Max.Mixing.Weight      0.200       # default=0.40
scf.Mixing.History          6          # default=5
scf.Mixing.StartPulay      12          # default=6
scf.criterion             1.0e-6       # default=1.0e-6 (Hartree)
scf.system.charge          0.0         # default=0.0
```



a : (**tv**[1][1], **tv**[1][2], **tv**[1][3])
b : (**tv**[2][1], **tv**[2][2], **tv**[2][3])
c : (**tv**[3][1], **tv**[3][2], **tv**[3][3])

Solver = 3

Kspace_grid1 = 3
Kspace_grid2 = 3
Kspace_grid3 = 3

🟨 … in "Input_std.c"

Outline of Subroutine “Band_DFT_Col.c”

1. Setting $\mathbf{k}$-mesh points

2. Searching first-neighbor atoms (determination of $\mathbf{R}_n$-region)

3. Setting overlap and Hamiltonian matrices for 3-D periodic boundary condition

4. Solving generalized eigenvalue problem

5. Finding chemical potential

6. Constructing density matrix (⇒ Check of SCF-convergence)

K-points (1); Weight Factor for DOS

Kspace_grid1
Kspace_grid2
Kspace_grid3

$\Rightarrow$

knum_i
knum_j
knum_k

in “Band_DFT_Col.c”

```
 for (i=0;i<=knum_i-1;i++) {
      for (j=0;j<=knum_j-1;j++) {
        for (k=0;k<=knum_k-1;k++) {
          k_op[i][j][k]=-999;
        }
      }
   }

   for (i=0;i<=knum_i-1;i++) {
     for (j=0;j<=knum_j-1;j++) {
       for (k=0;k<=knum_k-1;k++) {
         if ( k_op[i][j][k]==-999 ) {
           k_inversion(i,j,k,knum_i,knum_j,knum_k,&ii,&ij,&ik);
           if ( i==ii && j==ij && k==ik ) {
             k_op[i][j][k]     = 1;
           }

           else {
             k_op[i][j][k]     = 2;
             k_op[ii][ij][ik]  = 0;
           }
         }
       } /* k */
     } /* j */
   } /* i */
```

i	0	1	2
ii	2	1	0

inversion

k_op[i][j][k] : weight factor

***l*. 426 in “Band_DFT_Col.c”**

K-points (2); Weight Factor for DOS

ex) case of 2-dimensional k-space (3 × 3)

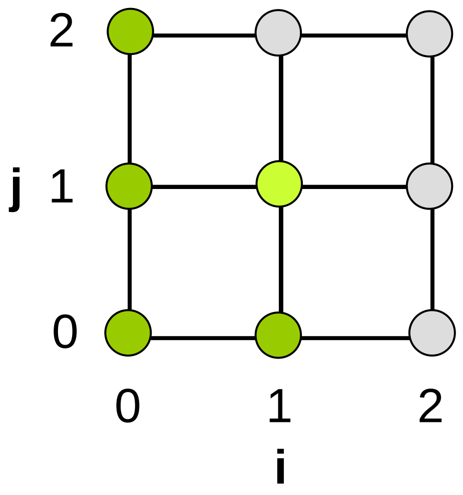
loop	(i,j)			(ii,ij)	
1	0	0		2	2
2	0	1		2	1
3	0	2		2	0
4	1	0		1	2
5	1	1		1	1
6	1	2			
7	2	0			
8	2	1			
9	2	2			

k_op = 2

k_op = 0

k_op = 1
(Γ point)

skipped
because **k_op**[i][j] ≠ –999



K-points (3); K-Grids by Regular Mesh

from *l*. 467 in "Band_DFT_Col.c"

```
/* set T_KGrids1,2,3 and T_k_op */

   T_knum = 0;
   for (i=0; i<knum_i; i++){

     if (knum_i==1)  k1 = 0.0;
     else            k1 = -0.5 + (2.0*(double)i+1.0)/(2.0*(double)knum_i)
                          + Shift_K_Point;

     for (j=0; j<knum_j; j++){

       if (knum_j==1)  k2 = 0.0;
       else            k2 = -0.5 + (2.0*(double)j+1.0)/(2.0*(double)knum_j)
                            - Shift_K_Point;

       for (k=0; k<knum_k; k++){

         if (knum_k==1)  k3 = 0.0;
         else            k3 = -0.5 + (2.0*(double)k+1.0)/(2.0*(double)knum_k)
                              + 2.0*Shift_K_Point;

         if (0<k_op[i][j][k]){

           T_KGrids1[T_knum] = k1;
           T_KGrids2[T_knum] = k2;
           T_KGrids3[T_knum] = k3;
           T_k_op[T_knum]    = k_op[i][j][k];

           T_knum++;
         }
       }
     }
   }
```

K-grids with non-zero weight (1-D array)

1-dimensionalization of **k_op**[i][j][k]

(After loop processing) total num. of k-points with non-zero weight

Before K-Loop; 1-Dimensionalizing S and H Matrices

```
 /* set S1 */

if (SCF_iter==1 || all_knum!=1){
   size_H1 = Get_OneD_HS_Col(1, CntOLP,   S1, MP, order_GA, My_NZeros, SP_NZeros, SP_Atoms);
 }

diagonalize1:

  /* set H1 */

  if (SpinP_switch==0){
    size_H1 = Get_OneD_HS_Col(1, nh[0], H1,   MP, order_GA, My_NZeros, SP_NZeros, SP_Atoms);
  }
  else if (1<numprocs0){
    size_H1 = Get_OneD_HS_Col(1, nh[0], H1,   MP, order_GA, My_NZeros, SP_NZeros, SP_Atoms);
    size_H1 = Get_OneD_HS_Col(1, nh[1], CDM1, MP, order_GA, My_NZeros, SP_NZeros, SP_Atoms);

    if (myworld1){
      for (i=0; i<size_H1; i++){
        H1[i] = CDM1[i];
      }
    }
  }
  else{
    size_H1 = Get_OneD_HS_Col(1, nh[spin], H1, MP, order_GA, My_NZeros, SP_NZeros, SP_Atoms);
  }
```

because S do not change until the next MD-step starts

Overlap matrix : **CntOLP** (= **OLP** in DFT.c) ⇒ **S1**

Hamiltonian matrix : **nh** (= **H** in DFT.c) ⇒ **H1**

Starting point in 1-D array of each atom

from *l*. 773 in "Band_DFT_Col.c"

In case of basis sets, "Na9.0-s1p1" and "Cl7.0-s1p1"...

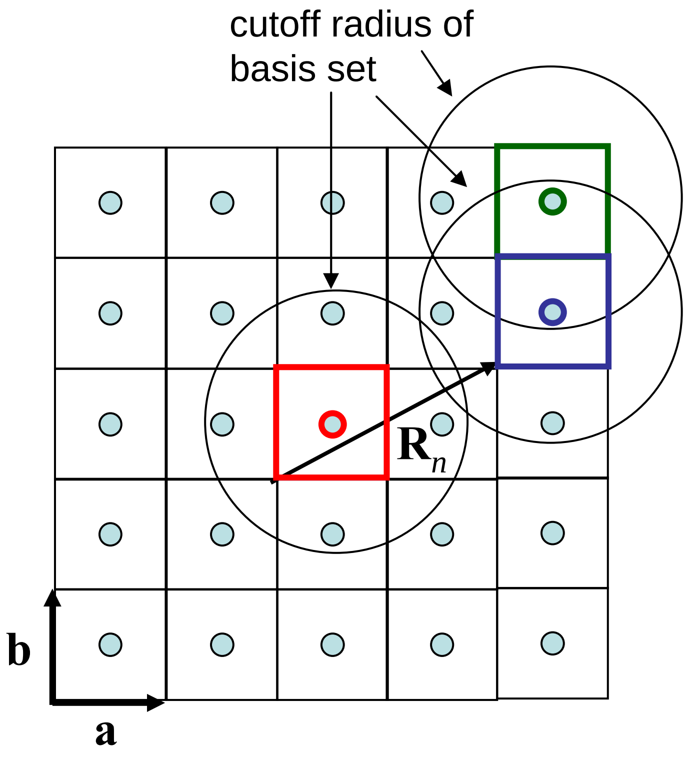
atom	1 (Na)				2 (Cl)			
PAO	1 (s)	2 (p_x)	3 (p_y)	4 (p_z)	1 (s)	2 (p_x)	3 (p_y)	4 (p_z)
1-D index	1	2	3	4	5	6	7	8
MP[atom]	1				5			

K-Loop (1); S and H for 3-D Periodic Boundary Condition

$$S_{ij}(\mathbf{k}) = \sum_{\mathbf{R}_n} e^{i\mathbf{k}\cdot\mathbf{R}_n} \underbrace{\int \phi_i(\mathbf{r})\phi_j(\mathbf{r}-\mathbf{R}_n)\,d\mathbf{r}}_{\textbf{OLP}\text{ (in DFT.c)}}$$

$$H_{ij}(\mathbf{k}) = \sum_{\mathbf{R}_n} e^{i\mathbf{k}\cdot\mathbf{R}_n} \underbrace{\int \phi_i(\mathbf{r})\bar{H}\phi_j(\mathbf{r}-\mathbf{R}_n)\,d\mathbf{r}}_{\textbf{H}\text{ (in DFT.c)}}$$

- $\mathbf{R}_n = k\mathbf{a} + l\mathbf{b} + m\mathbf{c}$
- $\mathbf{R}_n$'s are determined as first-neighbor atoms in 3-D periodic boundary system.
- $\mathbf{R}_n$ is corresponding to each first-neighbor atom.



ex) a simple case: one atom per cell

○ is a first-neighbor atom of ○
○ is not a first-neighbor atom of ○

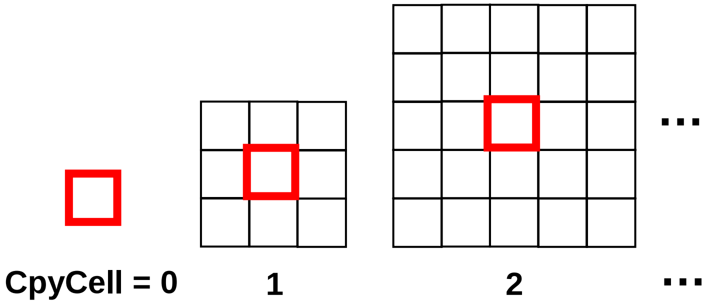
K-Loop (2); vector $\mathbf{R}_n$ (= $k\mathbf{a} + l\mathbf{b} + m\mathbf{c}$)

from *l.* 216 in "truncation.c"

```
CpyCell = 0;
   po = 0; TFNAN = 0; TSNAN = 0;

   do{

     CpyCell++;

     /***********************************************
          allocation of arrays listed above and
                Generation_ATV(CpyN);
     ***********************************************/

     TCpyCell = Set_Periodic(CpyCell,0);

     /***********************************************
       find Max_FSNAN by the physical truncation
         for allocation of natn, ncn, and Dis
     ***********************************************/

     Estimate_Trn_System(CpyCell,TCpyCell,0);

     /***********************************************
             allocation of natn, ncn, and Dis
     ***********************************************/

     Allocate_Arrays(3);
```

while po = 0 (see the next page)

generation of **atv**'s (= $\mathbf{R}_n$)



$$\left\{\begin{aligned} k\mathbf{a} &= \textbf{atv}[n][1] \\ l\mathbf{b} &= \textbf{atv}[n][2] \\ m\mathbf{c} &= \textbf{atv}[n][3] \end{aligned}\right.$$

$$\left\{\begin{aligned} k &= \textbf{atv\_ijk}[n][1] \\ l &= \textbf{atv\_ijk}[n][2] \\ m &= \textbf{atv\_ijk}[n][3] \end{aligned}\right.$$

```
    /**********************
     find TFNAN and TSNAN
    **********************/

    TFNAN2 = TFNAN;
    TSNAN2 = TSNAN;

   Trn_System(MD_iter,CpyCell,TCpyCell,0,1);

    if ( TFNAN==TFNAN2 && TSNAN==TSNAN2 && (Solver==1 || Solver==5
                                         || Solver==6 || Solver==8) ) po++;
    else if (TFNAN==TFNAN2 && (Solver==2 || Solver==3 || Solver==4
                                         || Solver==7 || Solver==9) ) po++;
    else if (CellNN_flag==1)                                          po++;

    Free_truncation(CpyCell,TCpyCell,0);

 } while (po==0);
```

- the index of $\mathbf{R}n$ between GAth and LBth atoms "**ncn**[GA][LB]"
- the number of first-neighbor atoms of *m*th atom "**FNAN**[m]"

- By increasing "CpyCell", one can search some point where the num. of first-neighbor atoms does not change moreover

⇒ finding range of $\mathbf{R}_n$

K-Loop (3); S and H matrices

from *l*. 809 in "Band_DFT_Col.c"

```
for (kloop0=0; kloop0<num_kloop0; kloop0++){

   kloop = S_knum + kloop0;

   k1 = T_KGrids1[kloop];
   k2 = T_KGrids2[kloop];
   k3 = T_KGrids3[kloop];

k = 0;
   for (AN=1; AN<=atomnum; AN++){
     GA_AN = order_GA[AN];
     wanA = WhatSpecies[GA_AN];
     tnoA = Spe_Total_CNO[wanA];
     Anum = MP[GA_AN];

     for (LB_AN=0; LB_AN<=FNAN[GA_AN]; LB_AN++){
       GB_AN = natn[GA_AN][LB_AN];
       Rn = ncn[GA_AN][LB_AN];
       wanB = WhatSpecies[GB_AN];
       tnoB = Spe_Total_CNO[wanB];
       Bnum = MP[GB_AN];

       l1 = atv_ijk[Rn][1];
       l2 = atv_ijk[Rn][2];
       l3 = atv_ijk[Rn][3];
       kRn = k1*(double)l1 + k2*(double)l2 + k3*(double)l3;

       si = sin(2.0*PI*kRn);
       co = cos(2.0*PI*kRn);
```

$$\mathbf{k} = 2\pi\frac{\mathbf{b}\times\mathbf{c}}{\Delta V}k_1 + 2\pi\frac{\mathbf{c}\times\mathbf{a}}{\Delta V}k_2 + 2\pi\frac{\mathbf{a}\times\mathbf{b}}{\Delta V}k_3$$

$$\left(\Delta V = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})\right)$$

$$\mathbf{R}_n = k\mathbf{a} + l\mathbf{b} + m\mathbf{c}$$

$$\therefore \mathbf{k}\cdot\mathbf{R}_n = 2\pi\left(k_1 k + k_2 l + k_3 m\right)$$

$$\sum_{\mathbf{R}_n} \boxed{e^{i\mathbf{k}\cdot\mathbf{R}_n}} \int d\mathbf{r}\cdots$$

```
        for (i=0; i<tnoA; i++){

          for (j=0; j<tnoB; j++){

            H[Anum+i][Bnum+j].r += H1[k]*co;
            H[Anum+i][Bnum+j].i += H1[k]*si;

            k++;

          }


          if (SCF_iter==1 || all_knum!=1){

            k -= tnoB;

            for (j=0; j<tnoB; j++){

              S[Anum+i][Bnum+j].r += S1[k]*co;
              S[Anum+i][Bnum+j].i += S1[k]*si;

              k++;
            }
          }

        }
      }
    }
```

$$H_{ij}(\mathbf{k}) = \boxed{\sum_{\mathbf{R}_n}} e^{i\mathbf{k}\cdot\mathbf{R}_n} \int \phi_i(\mathbf{r}) \bar{H} \phi_j(\mathbf{r}-\mathbf{R}_n) d\mathbf{r}$$

$$S_{ij}(\mathbf{k}) = \boxed{\sum_{\mathbf{R}_n}} e^{i\mathbf{k}\cdot\mathbf{R}_n} \int \phi_i(\mathbf{r}) \phi_j(\mathbf{r}-\mathbf{R}_n) d\mathbf{r}$$

K-Loop (4); Solving Eigenvalue Problem (Outline)

[1] $$\mathbf{HC} = \boldsymbol{\varepsilon}\mathbf{SC}$$: generalized eigenvalue problem

[2] $$\mathbf{U}^{\dagger}\mathbf{SU} = \mathbf{s}$$: diagonalization of $\mathbf{S}$ (solving eigenvalue problem for $\mathbf{S}$)

[3] $$\begin{cases} \mathbf{X} = \mathbf{U}\mathbf{s}^{-1/2} \\ \mathbf{X}^{\dagger} = \left(\mathbf{s}^{-1/2}\right)^{\dagger}\mathbf{U}^{\dagger} \\ \left(\mathbf{X}^{\dagger}\mathbf{SX} = \mathbf{I}\right) \end{cases}$$: canonical orthogonalization of $\mathbf{S}$

[4] $$\mathbf{X}^{\dagger}\mathbf{HXX}^{\dagger}\mathbf{C} = \boldsymbol{\varepsilon}\mathbf{X}^{\dagger}\mathbf{SXX}^{\dagger}\mathbf{C}$$: introducing $\mathbf{X}$ into $\mathbf{HC} = \boldsymbol{\varepsilon}\mathbf{SC}$

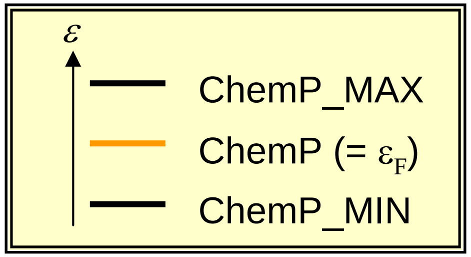
[5] $$\mathbf{H}'\mathbf{C}' = \boldsymbol{\varepsilon}\mathbf{C}'$$: standard eigenvalue problem

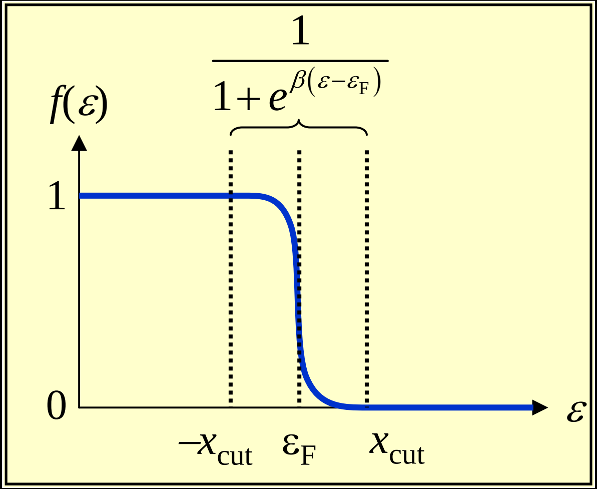
The detail can be checked in the supplemental documentation "*Eigenvalue_Problem_(in_Band_DFT_Col.c)*".

K-Loop (5); Finding Chemical Potential

```
  po = 0;
  loop_num = 0;
  ChemP_MAX = 10.0;
  ChemP_MIN =-10.0;

  do {

    loop_num++;

    ChemP = 0.50*(ChemP_MAX + ChemP_MIN);
    Num_State = 0.0;

    for (kloop=0; kloop<T_knum; kloop++){
      for (spin=0; spin<=SpinP_switch; spin++){
        for (l=1; l<=MaxN; l++){

          x = (EIGEN[spin][kloop][l] - ChemP)*Beta;

          if (x<=-x_cut)        FermiF = 1.0;
          else if (x>=x_cut)    FermiF = 0.0;
          else                  FermiF = 1.0/(1.0 + exp(x));

          Num_State += FermiF*(double)T_k_op[kloop];
        }
      }
    }

    if (SpinP_switch==0)
      Num_State = 2.0*Num_State/sum_weights;
    else
      Num_State = Num_State/sum_weights;

    Dnum = TZ - Num_State - system_charge;

    if (0.0<=Dnum)  ChemP_MIN = ChemP;
    else            ChemP_MAX = ChemP;
    if (fabs(Dnum)<10e-14)  po = 1;
  }
  while (po==0 && loop_num<2000);
```

from *l*.1462 in "Band_DFT_Col.c"





Num_state = $f(\varepsilon)w(\mathbf{k})\sum_{\mathbf{k},l} n_{\mathbf{k},l}$

$w(\mathbf{k})$: T_k_op

$n_{\mathbf{k},l}$: occupancy (= 0 or 1)

TZ : total valence charge

- Dnum > 0 : running short of electrons

  ⇒ put Fermi level upper
- Dnum < 0 : having excess electrons

  ⇒ put Fermi level lower

K-Loop (6); Density Matrix $\rho_{ij}(\mathbf{R}_n)$

$$\rho_{ij}\left(\mathbf{R}_n\right)=\frac{1}{V_{\mathrm{B}}}\int_{\mathrm{B}}d\mathbf{k}\sum_{l}^{\mathrm{occ.}}\mathrm{e}^{i\mathbf{R}_n\cdot\mathbf{k}}c_{l,i}^{*}\left(\mathbf{k}\right)c_{l,j}\left(\mathbf{k}\right)$$

from *l*. 2498 in "Band_DFT_Col.c"

```
for (AN=1+OMPID; AN<=atomnum; AN+=Nthrds){

        GA_AN = order_GA[AN];
        wanA = WhatSpecies[GA_AN];
        tnoA = Spe_Total_CNO[wanA];
        Anum = MP[GA_AN];

        k = (int)H[0][AN].r;

        for (LB_AN=0; LB_AN<=FNAN[GA_AN]; LB_AN++){
          GB_AN = natn[GA_AN][LB_AN];
          Rn = ncn[GA_AN][LB_AN];
          wanB = WhatSpecies[GB_AN];
          tnoB = Spe_Total_CNO[wanB];
          Bnum = MP[GB_AN];

          l1 = atv_ijk[Rn][1];
          l2 = atv_ijk[Rn][2];
          l3 = atv_ijk[Rn][3];
          kRn = k1*(double)l1 + k2*(double)l2 + k3*(double)l3;

          si = sin(2.0*PI*kRn);
          co = cos(2.0*PI*kRn);
```

$$\mathbf{k}=2\pi\frac{\mathbf{b}\times\mathbf{c}}{\Delta V}k_1+2\pi\frac{\mathbf{c}\times\mathbf{a}}{\Delta V}k_2+2\pi\frac{\mathbf{a}\times\mathbf{b}}{\Delta V}k_3$$

$$\left(\Delta V=\mathbf{a}\cdot\left(\mathbf{b}\times\mathbf{c}\right)\right)$$

$$\mathbf{R}_n=k\mathbf{a}+l\mathbf{b}+m\mathbf{c}$$

$$\therefore\mathbf{k}\cdot\mathbf{R}_n=2\pi\left(k_1k+k_2l+k_3m\right)$$

$$e^{i\mathbf{k}\cdot\mathbf{R}_n}$$

```
        for (i=0; i<tnoA; i++){

               ia = Anum + i;

               for (j=0; j<tnoB; j++){

                 jb = Bnum + j;

          d1 = 0.0;
          d2 = 0.0;

          for (l=1; l<=lmax; l++){

           tmp = co*(H[ia][l].r*H[jb][l].r + H[ia][l].i*H[jb][l].i)
                -si*(H[ia][l].r*H[jb][l].i - H[ia][l].i*H[jb][l].r);

            d1 += VecFkw[l]*tmp;
            d2 += VecFkwE[l]*tmp;;

          }

          CDM1[k] += d1;
          EDM1[k] += d2;

          /* increment of k */

          k++;

        }
      }
    }
  }
```

Summation and integration with regard to **k** and *l*

$$\sum_{l}^{\text{occ.}} \int d\mathbf{k} \cdots$$

summation of conjugated terms

$$e^{i\mathbf{R}_n\cdot\mathbf{k}} c_{l,i}^{*}(\mathbf{k}) c_{l,j}(\mathbf{k}) + e^{-i\mathbf{R}_n\cdot\mathbf{k}} c_{l,i}(\mathbf{k}) c_{l,j}^{*}(\mathbf{k})$$

1-D index for (i, j) and $\mathbf{R}_n$

Density matrix is used in checking SCF-converge