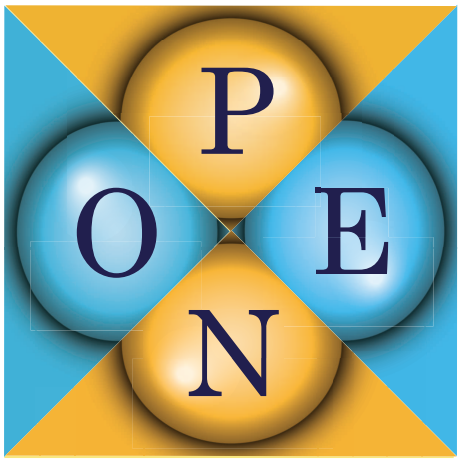
User's manual of OpenMX Ver. 3.5



Contributors

T. Ozaki (JAIST),
H. Kino (NIMS),
J. Yu (SNU),
M. J. Han (SNU),
N. Kobayashi (Tsukuba Univ.),
M. Ohfuchi (Fujitsu Labs.)
F. Ishii (Kanazawa Univ.)
T. Ohwaki (Nissan)
H. Weng (JAIST)
M. Toyoda (JAIST)
K. Terakura (JAIST)

July 25, 2011

Contents

1 About OpenMX

OpenMX (Open source package for Material eXplorer) is a software package for nano-scale material simulations based on density functional theories (DFT) [1], norm-conserving pseudopotentials [2, 20, 21], and pseudo-atomic localized basis functions [23]. Since the code is designed for the realization of large-scale *ab initio* calculations on parallel computers, it is anticipated that OpenMX can be a useful and powerful tool for nano-scale material sciences in a wide variety of systems such as bio-materials, carbon nanotubes, magnetic materials, and nanoscale conductors. The distribution of the program package and the source codes follow the practice of the GNU General Public License (GPL) [47], and they are downloadable from http:http://www.openmx-square.org/

Features and capabilities of OpenMX Ver. 3.5 are as follows:

- Total energy and forces by cluster, band, and O(N) methods
- Local density approximation (LDA, LSDA) [2, 3, 4] and generalized gradient approximation (GGA) [5] to the exchange-correlation potential
- Norm-conserving pseudopotentials [2, 20, 21]
- Variationally optimized pseudo-atomic basis functions [23]
- Fully and scalar relativistic treatment within pseudopotential scheme [10, 19, 13]
- Non-collinear DFT [6, 7, 8, 9]
- Constraint DFT for non-collinear spin and orbital orientation [11]
- Collinear LDA+U and non-collinear LDA+U methods [16]
- Macroscopic polarization by Berry's phase [12]
- Divide-conquer (DC) method [28], generalized DC method, and Krylov subspace method for O(N) eigenvalue solver
- Simple, RMM-DIIS [31], GR-Pulay [30], Kerker [32], and RMM-DIIS with Kerker's metric [31] charge mixing schemes
- Exchange coupling parameter [14, 15]
- Optical conductivity
- Charge doping
- Uniform electric field
- Full and constrained geometry optimization
- Electric transport calculation by a non-equilibrium Green's function method
- Construction of maximally localized wannier functions
- NVE ensemble molecular dynamics

- NVT ensemble molecular dynamics by a velocity scaling [17] and the Nose-Hoover methods [18]
- Mulliken, Voronoi, and ESP fitting analysis of charge and spin densities
- Analysis of wave functions and electron (spin) densities
- Dispersion analysis by the band calculation
- Density of states (DOS) and projected DOS
- Flexible data format for the input
- Completely dynamic memory allocation
- Parallel execution by Message Passing Interface (MPI)
- Parallel execution by OpenMP
- Useful user interface for developers
- Evaluation of two-center integrals using Fourier transformation [27]
- Evaluation of three-center integrals by a projector expansion method [24]
- Solution of Poisson's equation using FFT [26]

Considerable functionalities are available for calculations of physical properties such as magnetic, dielectric, electric transport properties as listed above. Not only conventional diagonalization schemes are provided for clusters, molecules, slab, and solids, but also linear scaling methods are supported as the eigenvalue solver. Three calculation parts in OpenMX are mainly time-consuming:

- Evaluation of Hamiltonian matrix elements
- Solution of Poisson's equation
- Diagonalization of the generalized secular equation

For the first and second parts, the computational time always scales as O(N) and O(Nlog(N)) for any eigenvalue solver, where N is the number of atoms, basis functions, or grid points. When the conventional diagonalization scheme (cluster and band methods) is used, the computational time for the third part scales as O(N^3). On the other hand, the O(N) methods can solve the eigenvalue problem in O(N) operation in exchange for accuracy. For large scale calculations parallel execution by MPI or OpenMX is supported for parallel machines. The hybrid parallelization by OpenMP/MPI is also supported which is suitable for PC cluster consisting of multicore processors. All work arrays in the program codes are dynamically allocated with the minimum memory size required by an input file. The execution environment is unix and linux. For the execution of OpenMX, you are required to possess pseudo-atomic basis orbitals and pseudopotentials. These input data can be calculated using ADPACK which is a program package for atomic density functional calculations. Conveniently, the data for several elements and ADPACK are available from a web site (http://www.openmx-square.org/). We are continuously working toward development. Motivated contributors who want to develop the open source codes are welcome. If so, the contact information is available in the above website.

2 Installation

2.1 Including libraries

OpenMX uses two or three library packages. The following two libraries are indispensable.

- LAPACK (and BLAS) (http://www.netlib.org/)
- FFTW (http://www.fftw.org/)

If you try to perform the MPI execution, in addition to above two libraries, you need to install a MPI package such as

- MPICH (http://www-unix.mcs.anl.gov/mpi)
- LAM (http://www.lam-mpi.org/)

If these library packages are not installed on your machine, before the installation of OpenMX, you are required to install them. If these libraries packages are available on your machine, you can proceed the following procedure for the installation. Then, after downloading openmx3.5.tar.gz, decompress it as follows:

```
% tar zxvf openmx3.5.tar.gz
```

When it is completed, you can find four directories (source, work, DFT_DATA, DFT_DATA06) under the directory, openmx3.5. The directories, 'source', 'work', 'DFT_DATA' and 'DFT_DATA06', contain source files, input files, and data files for the pseudo-atomic basis functions and the pseudopotentials of Ver. 2004 and 2006, respectively.

2.2 Serial version

To proceed the installation of the serial version, move to the directory, 'source', and modify the *makefile* in 'source' to specify the compiler and libraries by **CC** and **LIB**. The default for the specification of **CC** and **LIB** in *makefile* is as follows:

```
CC      = gcc -O3 -Dnompi -Dnoomp -I/usr/local/include -I/home/ozaki/include
LIB     = -L/usr/local/lib -lfftw3 -llapack -lblas -lg2c -static
```

where '-Dnompi' means that MPI is not used, and '-Dnoomp' means that OpenMP is not used. These options must be added if you want to generate the serial version in all cases when you change a compiler. You have to set the **CC** and **LIB** appropriately on your computational environment so that the compilation and linking can be correctly performed and the executable file can be well optimized. After specifying the **CC** and **LIB**, install as follows:

```
% make install
```

When the compilation is completed normally, then you can find the executable file, openmx, in the directory, 'work'. You may change the compiler to make the executable file efficient, and if the intel compiler, icc, is used, the specification may be like this:

```
CC    = icc -O3 -Dnompi -Dnoomp
```

2.3 MPI version

As well as the case of the serial version, to generate the MPI version only thing you have to do is to specify **CC** and **LIB** in the *makefile* in 'source'. To proceed the installation of the MPI version, move to the directory, 'source', and specify **CC** and **LIB** in the *makefile* as follows:

```
CC    = mpicc -Dnoomp -O3 -I/usr/local/include
LIB   = -L/usr/local/lib -lfftw3 -llapack -lblas -lg2c -static
```

Of course, the specification depends on your computer environment. After specifying CC appropriately, then install as follows:

```
% make install
```

When the compilation is completed normally, then you can find the executable file, openmx, in the directory, 'work'. To make the execution of OpenMX efficient, you can change a compiler and compile options appropriate for your computational environment, which can generate an optimized executable file. Several examples for **CC** and **LIB** can be found in *makefile* in the 'source' directory for your convenience.

2.4 OpenMP/MPI version

To generate the OpenMP/MPI hybrid parallelized version, only thing you have to do is to specify **CC** and **LIB** in the *makefile* in 'source'. To proceed the installation of the OpenMP/MPI version, move to the directory, 'source', and specify **CC** and **LIB** in the *makefile*, for example, as follows:

For icc

```
CC    = mpicc -openmp -O3 -I/usr/local/include
LIB   = -L/usr/local/lib -lfftw3 -llapack -lblas -lg2c -static
```

For pgcc

```
CC    = mpicc -mp -O3 -I/usr/local/include
LIB   = -L/usr/local/lib -lfftw3 -llapack -lblas -lg2c -static
```

Of course, the specification depends on your computer environment. Also, it is noted that older versions of icc and pgcc do not support the compiler option for OpenMP. After specifying CC appropriately, then install as follows:

```
% make install
```

When the compilation is completed normally, then you can find the executable file, openmx, in the directory, 'work'. To make the execution of OpenMX efficient, you can change a compiler and compile options appropriate for your computational environment, which can generate an optimized executable file. It should be mentioned that the compilation of only OpenMP without MPI is also possible.

2.5 FFTW2 or FFTW3

OpenMX Ver. 3.5 supports both FFTW2 and FFTW3. In OpenMX Ver. 3.5, we assume FFTW3 as defaul. Then, you may link FFTW3 in your makefile as follows:

```
LIB      = -L/usr/local/lib -fftw3 -llapack -lblas -lg2c -static
```

If you want to use FFTW2, you need to add '-Dfftw2' for the compile option as follows:

```
CC       = gcc -O3 -Dfftw2 -Dnompi -Dnoomp
LIB      = -L/usr/local/lib -fftw -llapack -lblas -lg2c -static
```

Since the computational time for FFT employed in OpenMX is a small fraction in the total computational time, you can use either FFTW2 or FFTW3 without loosing significant efficiency.

2.6 Other options, -Dblaswrap and -lI77

In some environment, adding two options *-Dblaswrap* and *-lI77* is required, while we do not fully understand why such a dependency exists. In such a case, add two options for **CC** and LIB as follows:

```
CC       = gcc -O3 -Dnompi -Dnoomp -Dblaswrap
LIB      = -L/usr/local/lib -fftw3 -llapack -lblas -lg2c -lI77 -static
```

Other options, -Df77, -Df77_, -Df77__, -DF77, -DF77_, -DF77__

When lapack and blas routines are linked, the specification of the routine name could depend on the machine environment. The variation could be capital or small letter, or with or without of the underscore. To choose a proper name of lapack and blas routines on your computational environment, you can specify an option by -Df77, -Df77_, -Df77__, -DF77, -DF77_, or -DF77__. If the capital letter is needed in calling the lapack routines, then choose 'F', and choose a type of the underscore by none, '_', or '__'. The default set is '-Df77_'.

2.7 Platforms

So far, we have confirmed that OpenMX Ver. 3.5 runs normally on the following machines:

- Pentium4
- Xeon
- Opteron
- Itanium2
- Cray XT5
- Sun Fire V890

2.8 Tips for installation

Since most problems in installation of OpenMX come from compilation of LAPACK and BLAS and its linking, tips for installation on several platforms are given below.

- Intel Pentium 4 and Xeon (32 bit)

  A simple way is to use highly optimized ATLAS library (libatlas_p4.a) provided by Dr. Axel Kohlmeyer (Thanks to Dr. Axel Kohlmeyer) on the following website.

  http://www.theochem.ruhr-uni-bochum.de/ axel.kohlmeyer/cpmd-linux.html#atlas

  Then, you may link these libraries in makefile of OpenMX as follows:

  CC = mpicc -openmp -O3 -I/usr/local/include
  LIB = -L/usr/local/lib -lfftw3 -latlas_p4 -static

- Intel Pentium D and Xeon (EM64T)

  A simple way is to use highly optimized ATLAS library (libatlas_x86_64.a) provided by Dr. Axel Kohlmeyer (Thanks to him) on the following website.

  http://www.theochem.ruhr-uni-bochum.de/ axel.kohlmeyer/cpmd-linux.html#atlas

  Then, you may link these libraries in makefile of OpenMX as follows:

  CC = mpicc -openmp -O3 -I$(HOME)/include -I$(HOME)/include
  LIB = -L$(HOME)/lib -lfftw3 -latlas_x86_64 -static

- AMD Opteron

  A simple way is to use ACML library (acml-3-6-0-gnu-64bit.tgz) provided by AMD on the following website.

  http://developer.amd.com/acml.aspx

  Then, you may link these libraries in makefile of OpenMX as follows:

  CC = mpicc -mp -fast -I$(HOME)/include
  LIB = -L$(HOME)/lib -lfftw3 -lacml -L/opt/gcc33/lib64 -lg2c

- Intel Itanium

  A simple way is to use MKL library. Then, you may link these libraries in makefile of OpenMX as follows:

  CC = mpicc -openmp -O3 -I$(HOME)/include -I/opt/intel/mkl/include -Wl,–allow-multiple-definition
  LIB = -L$(HOME)/lib -lfftw3 -L/opt/intel/mkl/lib/64 -lmkl_lapack -lmkl_ip -lguide -L/usr/lib -lpthread -static

3 Test calculation

If the installation is completed normally, please move to the directory 'work' and perform the program, openmx, using an input file, *Methane.dat*, which can be found in the directory 'work' as follows:

```
% ./openmx Methane.dat > met.std &
```

If you use the MPI version:

```
% mpirun -np 1 openmx Methane.dat > met.std &
```

Or if you use the MPI/OpenMP version:

```
% mpirun -np 1 openmx Methane.dat -nt 1 > met.std &
```

The test input file, Methane.dat, is for performing the SCF calculation of a methane molecule with a fixed structure (No MD). The calculation is performed in only about 19 seconds by using a 2.8 GHz Xeon machine, although it is dependent on a computer. When the calculation is completed normally, 11 files and one directory

```
met.std              standard output of the SCF calculation
met.out              input file and standard output
met.xyz              final geometrical structure
met.ene              values computed at every MD step
met.memory0          analysis for used memory
met.md               geometrical structures at every MD step
met.md2              geometrical structure of the final MD step
met.cif              cif file of the initial structure for Material Studio
met.tden.cube        total electron density in the Gaussian cube format
met.v0.cube          Kohn-Sham potential in the Gaussian cube format
met.vhart.cube       Hartree potential in the Gaussian cube format
met_rst/             directory storing restart files
```

are output to the directory, 'work'. The output data to a standard output is stored to the file, met.std which is helpful to know the calculation flow of SCF procedure. The file, met.out, includes computed results such as the total energy, forces, the Kohn-Sham eigenvalues, Mulliken charges, the convergence history for the SCF calculation, and analyzed computational time. A part of the file, met.out, is shown below. It is found that the eigenvalues energy converges by ten iterations within 1.0e-8 Hartree of the eigenvalues energy.

```
*******************************************************
*******************************************************
                SCF history at MD= 1
*******************************************************
*******************************************************
```

```
 SCF=   1  NormRD=  1.000000000000  Uele= -3.799184452246
 SCF=   2  NormRD=  0.294505017736  Uele= -3.180922853695
 SCF=   3  NormRD=  0.088735677892  Uele= -3.371991788328
 SCF=   4  NormRD=  0.021096020042  Uele= -3.435330322070
 SCF=   5  NormRD=  0.006019683784  Uele= -3.449516147408
 SCF=   6  NormRD=  0.000784960310  Uele= -3.452522027174
 SCF=   7  NormRD=  0.000002401488  Uele= -3.453266301971
 SCF=   8  NormRD=  0.000000599833  Uele= -3.453266643608
 SCF=   9  NormRD=  0.000000184742  Uele= -3.453266654138
 SCF=  10  NormRD=  0.000000562332  Uele= -3.453266655628
```

Also, the total energy, chemical potential, Kohn-Sham eigenvalues, the Mulliken charges, dipole moment, forces, fractional coordinate, and analysis of computational time are output in 'met.out' as follows:

```
*****************************************************
        Total energy (Hartree) at MD = 1
*****************************************************

  Uele.         -3.453266655628

  Ukin.          5.824571448666
  UH0.         -14.517598384684
  UH1.           0.012112580595
  Una.          -6.365977496421
  Unl.           0.681047544610
  Uxc0.         -1.609135574068
  Uxc1.         -1.609135574068
  Ucore.         9.551521413583
  Uhub.          0.000000000000
  Ucs.           0.000000000000
  Uzs.           0.000000000000
  Uzo.           0.000000000000
  Uef.           0.000000000000
  Utot.         -8.032594041787

  Note:

  Utot = Ukin+UH0+UH1+Una+Unl+Uxc0+Uxc1+Ucore+Uhub+Ucs+Uzs+Uzo+Uef

  Uene:   band energy
  Ukin:   kinetic energy
  UH0:    electric part of screened Coulomb energy
  UH1:    difference electron-electron Coulomb energy
```

```
  Una:    neutral atom potential energy
  Unl:    non-local potential energy
  Uxc0:   exchange-correlation energy for alpha spin
  Uxc1:   exchange-correlation energy for beta spin
  Ucore:  core-core Coulomb energy
  Uhub:   LDA+U energy
  Ucs:    constraint energy for spin orientation
  Uzs:    Zeeman term for spin magnetic moment
  Uzo:    Zeeman term for orbital magnetic moment
  Uef:    electric energy by electric field

  (see also PRB 72, 045121(2005) for the energy contributions)



  Chemical potential (Hartree)       0.000000000000

************************************************************
************************************************************
           Eigenvalues (Hartree) for SCF KS-eq.
************************************************************
************************************************************

   Chemical Potential (Hartree) =   0.00000000000000
   Number of States             =   8.00000000000000
   HOMO =  4
   Eigenvalues
                Up-spin            Down-spin
          1  -0.64275532805563  -0.64275532805563
          2  -0.36132252595285  -0.36132252595285
          3  -0.36127775831387  -0.36127775831387
          4  -0.36127771549143  -0.36127771549143
          5   0.26426269019400   0.26426269019400
          6   0.26445588063823   0.26445588063823
          7   0.26445588290286   0.26445588290286
          8   0.31938640324811   0.31938640324811

************************************************************
************************************************************
                  Mulliken populations
************************************************************
************************************************************

  Total spin S =  0.000000000000
```

		Up spin	Down spin	Sum	Diff
1	C	2.363735209	2.363735209	4.727470417	0.000000000
2	H	0.409066202	0.409066202	0.818132405	0.000000000
3	H	0.409066194	0.409066194	0.818132388	0.000000000
4	H	0.409066200	0.409066200	0.818132400	0.000000000
5	H	0.409066195	0.409066195	0.818132389	0.000000000

Decomposed Mulliken populations

1 C		Up spin	Down spin	Sum	Diff
	multiple				
s	0	0.598003833	0.598003833	1.196007665	0.000000000
sum over m		0.598003833	0.598003833	1.196007665	0.000000000
sum over m+mul		0.598003833	0.598003833	1.196007665	0.000000000
px	0	0.588514078	0.588514078	1.177028156	0.000000000
py	0	0.588703212	0.588703212	1.177406425	0.000000000
pz	0	0.588514085	0.588514085	1.177028171	0.000000000
sum over m		1.765731376	1.765731376	3.531462752	0.000000000
sum over m+mul		1.765731376	1.765731376	3.531462752	0.000000000

2 H		Up spin	Down spin	Sum	Diff
	multiple				
s	0	0.409066202	0.409066202	0.818132405	0.000000000
sum over m		0.409066202	0.409066202	0.818132405	0.000000000
sum over m+mul		0.409066202	0.409066202	0.818132405	0.000000000

3 H		Up spin	Down spin	Sum	Diff
	multiple				
s	0	0.409066194	0.409066194	0.818132388	0.000000000
sum over m		0.409066194	0.409066194	0.818132388	0.000000000
sum over m+mul		0.409066194	0.409066194	0.818132388	0.000000000

4 H		Up spin	Down spin	Sum	Diff
	multiple				
s	0	0.409066200	0.409066200	0.818132400	0.000000000
sum over m		0.409066200	0.409066200	0.818132400	0.000000000
sum over m+mul		0.409066200	0.409066200	0.818132400	0.000000000

5 H		Up spin	Down spin	Sum	Diff
	multiple				
s	0	0.409066195	0.409066195	0.818132389	0.000000000
sum over m		0.409066195	0.409066195	0.818132389	0.000000000
sum over m+mul		0.409066195	0.409066195	0.818132389	0.000000000

```
************************************************************
************************************************************
                    Dipole moment (Debye)
************************************************************
************************************************************

 Absolute D        0.00000009

                      Dx               Dy               Dz
 Total              0.00000004       0.00000005      -0.00000007
 Core               0.00000000       0.00000000       0.00000000
 Electron           0.00000004       0.00000005      -0.00000007
 Back ground       -0.00000000      -0.00000000       0.00000000

************************************************************
************************************************************
       xyz-coordinates (Ang) and forces (Hartree/Bohr)
************************************************************
************************************************************

<coordinates.forces
  5
    1     C     0.00000   0.00000   0.00000  -0.000000037541  0.000...
    2     H    -0.88998  -0.62931   0.00000  -0.048431334064 -0.034...
    3     H     0.00000   0.62931  -0.88998   0.000000053600  0.034...
    4     H     0.00000   0.62931   0.88998  -0.000000012054  0.034...
    5     H     0.88998  -0.62931   0.00000   0.048431331537 -0.034...
coordinates.forces>

************************************************************
************************************************************
       Fractional coordinates of the final structure
************************************************************
************************************************************

     1      C     0.00000000000000   0.00000000000000   0.00000000000000
     2      H     0.86968043640398   0.89633135611159   0.00000000000000
     3      H     0.00000000000000   0.10366864388841   0.86968043640398
     4      H     0.00000000000000   0.10366864388841   0.13031956359602
     5      H     0.13031956359602   0.89633135611159   0.00000000000000

************************************************************
************************************************************
```

```
               Computational Time (second)
************************************************************
************************************************************

   Elapsed.Time.        18.554

                            Min_ID    Min_Time    Max_ID    Max_Time
   Total Computational Time =  0       18.554       0       18.554
   readfile                 =  0       16.096       0       16.096
   truncation               =  0        0.728       0        0.728
   MD_pac                   =  0        0.010       0        0.010
   DFT                      =  0        1.376       0        1.376

*** In DFT ***

   Set_OLP_Kin              =  0        0.119       0        0.119
   Set_Nonlocal             =  0        0.198       0        0.198
   Set_Hamiltonian          =  0        0.087       0        0.087
   Poisson                  =  0        0.118       0        0.118
   Diagonalization          =  0        0.023       0        0.023
   Mixing_DM                =  0        0.002       0        0.002
   Force                    =  0        0.288       0        0.288
   Total_Energy             =  0        0.111       0        0.111
   Set_Aden_Grid            =  0        0.027       0        0.027
   Set_Orbitals_Grid        =  0        0.111       0        0.111
   Set_Density_Grid         =  0        0.053       0        0.053
   Others                   =  0        0.237       0        0.237
```

The files, met.tden.cube, met.v0.cube, met.vhart.cube, are the total electron density, the Kohn-Sham potential, and the Hartree potential, respectively, which are output in the Gaussian cube format. Since the Gaussian cube format is one of well used grid formats, you can visualize the files using free molecular modeling software such as gOpenMol [48], Molekel [49], and XCrysDen [50]. The visualization will be illustrated in the latter section.

4 Automatic running test

In addition to a running test of the Section 'Test calculation', if you want to check whether most functionalities of OpenMX have been successfully installed on your computer or not, we recommend for you to perform an automatic running test. To do this, you can run OpenMX as follows:

For the serial running

```
% ./openmx -runtest
```

For the MPI parallel running

```
% mpirun -np 4 openmx -runtest
```

For the OpenMP/MPI parallel running

```
% mpirun -np 4 openmx -runtest -nt 1
```

In this parallel running, you can specify other options for mpirun. Then, OpenMX will run with 14 test files, and compare calculated results with the reference results which are stored in 'work/input_example'. The comparison (absolute difference in the total energy and force) is stored in a file 'runtest.result' in the directory 'work'. The reference results were calculated using single processor of a 2.6 GHz Opteron machine. If the difference is within last seven digits, we may consider that the installation is successful. As an example, 'runtest.result' generated by the automatic running test is shown below:

1	input_example/Benzene.dat	Elapsed time(s)= 46.32	diff Utot= 0.000000000001	diff Force= 0.000000000000
2	input_example/C60.dat	Elapsed time(s)= 103.76	diff Utot= 0.000000000002	diff Force= 0.000000000001
3	input_example/CO.dat	Elapsed time(s)= 109.15	diff Utot= 0.000000000000	diff Force= 0.000000000000
4	input_example/Cr2.dat	Elapsed time(s)= 69.06	diff Utot= 0.000000000000	diff Force= 0.000000000000
5	input_example/Crys-MnO.dat	Elapsed time(s)= 135.83	diff Utot= 0.000000000004	diff Force= 0.000000000000
6	input_example/GaAs.dat	Elapsed time(s)= 137.91	diff Utot= 0.000000000000	diff Force= 0.000000000000
7	input_example/Glycine.dat	Elapsed time(s)= 40.18	diff Utot= 0.000000000000	diff Force= 0.000000000000
8	input_example/Graphite4.dat	Elapsed time(s)= 21.02	diff Utot= 0.000000000000	diff Force= 0.000000000000
9	input_example/H2O-EF.dat	Elapsed time(s)= 35.39	diff Utot= 0.000000000000	diff Force= 0.000000000000
10	input_example/H2O.dat	Elapsed time(s)= 41.26	diff Utot= 0.000000000000	diff Force= 0.000000000000
11	input_example/HYb.dat	Elapsed time(s)= 107.79	diff Utot= 0.000000000002	diff Force= 0.000000000000
12	input_example/Methane.dat	Elapsed time(s)= 25.67	diff Utot= 0.000000000003	diff Force= 0.000000002246
13	input_example/Mol_MnO.dat	Elapsed time(s)= 110.44	diff Utot= 0.000000000000	diff Force= 0.000000000000
14	input_example/Ndia2.dat	Elapsed time(s)= 19.52	diff Utot= 0.000000000000	diff Force= 0.000000000000

Total elapsed time (s) 1003.29

The comparison was made using 6 processors on the same machine. As you may know, the floating point operation depends on not only computational environment, but also the number of processors used in parallel execution. So we see in above example that there is a small difference even using the same machine. In addition, since two work arrays in OpenMX are allocated as single-precision floating point numbers for saving the size of working memory, the difference between 32 bits and 64 bits machines can be large in this 'runtest'. The elapsed time of each job is also output, so it is helpful for comparison of the computational speed depending on computational environment. In the directory 'work/input_example' you can find 'runtest.result' files generated on several platforms.

If you want to make reference files by yourself, please execute OpenMX as follows:

```
% ./openmx -maketest
```

Then, for *.dat files in 'work/input_example', OpenMX will generate *.out files in 'work/input_example'. So, you can add a new dat file which is used in the next running test. But, please make sure that the previous out files in 'work/input_example' will be overwritten by this procedure. For advanced testers for checking the reliability of code, see also the Sections 'Automatic force tester' and 'Automatic memory leak tester'.

5 Automatic running test with large-scale systems

In some cases, one may want to know machine performance for more time consuming calculations. For this purpose, an automatic running test with relatively large-scale systems can be performed by

For the serial running

```
% ./openmx -runtestL
```

For the MPI parallel running

```
% mpirun -np 4 openmx -runtestL
```

For the OpenMP/MPI parallel running

```
% mpirun -np 4 openmx -runtestL -nt 1
```

Then, OpenMX will run with 20 test files, and compare calculated results with the reference results which are stored in 'work/large_example'. The comparison (absolute difference in the total energy and force) is stored in a file 'runtestL.result' in the directory 'work'. The reference results were calculated using 40 MPI processes of a 2.6GHz Opteron cluster machine. If the difference is within last seven digits, we may consider that the installation is successful. As an example, 'runtestL.result' generated by the automatic running test is shown below:

1	large_example/5_5_13COb2.dat	Elapsed time(s)= 1740.27	diff Utot= 0.000000000038	diff Force= 0.000000000002
2	large_example/B2C62_Band.dat	Elapsed time(s)= 5035.37	diff Utot= 0.000000015973	diff Force= 0.000000006675
3	large_example/CG15c-Kry.dat	Elapsed time(s)= 1298.85	diff Utot= 0.000000001480	diff Force= 0.000000002069
4	large_example/DIA512-1.dat	Elapsed time(s)= 615.16	diff Utot= 0.000000033780	diff Force= 0.000000009994
5	large_example/FM.dat	Elapsed time(s)= 4533.59	diff Utot= 0.000000000018	diff Force= 0.000000000000
6	large_example/FeBCC.dat	Elapsed time(s)= 4128.44	diff Utot= 0.000000005436	diff Force= 0.000000000002
7	large_example/GEL.dat	Elapsed time(s)= 764.81	diff Utot= 0.000000000006	diff Force= 0.000000000000
8	large_example/GFRAG.dat	Elapsed time(s)= 801.09	diff Utot= 0.000000000002	diff Force= 0.000000000001
9	large_example/GGFF.dat	Elapsed time(s)=19139.17	diff Utot= 0.000000000026	diff Force= 0.000000000004
10	large_example/MCCN.dat	Elapsed time(s)= 2201.42	diff Utot= 0.000000000104	diff Force= 0.000000000065
11	large_example/Mn12_148_F.dat	Elapsed time(s)= 1538.40	diff Utot= 0.000000000014	diff Force= 0.000000000000
12	large_example/N1C999.dat	Elapsed time(s)= 3765.41	diff Utot= 0.000000013076	diff Force= 0.000000017373
13	large_example/Ni63-O64.dat	Elapsed time(s)= 2540.13	diff Utot= 0.000000000220	diff Force= 0.000000000126
14	large_example/Pt63.dat	Elapsed time(s)= 1718.62	diff Utot= 0.000000016353	diff Force= 0.000000000012
15	large_example/SialicAcid.dat	Elapsed time(s)= 204.23	diff Utot= 0.000000000278	diff Force= 0.000000000139
16	large_example/aAFM.dat	Elapsed time(s)= 5934.01	diff Utot= 0.000000000021	diff Force= 0.000000000001
17	large_example/cAFM.dat	Elapsed time(s)= 3994.10	diff Utot= 0.000000000022	diff Force= 0.000000000001
18	large_example/gAFM.dat	Elapsed time(s)= 5326.87	diff Utot= 0.000000000017	diff Force= 0.000000000001
19	large_example/nsV4Bz5.dat	Elapsed time(s)= 2675.28	diff Utot= 0.000000000394	diff Force= 0.000000000147
20	large_example/opt_4T2L_n.dat	Elapsed time(s)=27734.05	diff Utot= 0.000000000006	diff Force= 0.000000000005

Total elapsed time (s) 95689.26

The comparison was made using 20 processes by MPI and 2 threads by OpenMP (totally 40 cores) on the same machine. Since the automatic running test requires considerable memory size, you may encounter a segmentation fault on computational environment with small memory. Also the above example implies that the total elapsed time is more than 1 day even using 40 cores.

6 Input file

6.1 An example: methane molecule

An input file *Methane.dat* in the directory 'work' is shown below. This input file has a flexible data format, in which a parameter is given behind a keyword, the order of keywords is arbitrary, and a blank and a comment can be also described freely. For the keywords and options, both capital, small letters, and the mixture are acceptable, although these options in below example are written in a specific form.

```
#
#  SCF calculation of a methane molecule by the LDA
#  and the cluster method
#

#
# File Name
#

System.CurrrentDirectory         ./    # default=./
System.Name                     met
level.of.stdout                   1    # default=1 (1-3)
level.of.fileout                  1    # default=1 (0-2)

#
# Definition of Atomic Species
#

Species.Number       2
<Definition.of.Atomic.Species
 H   H4.0-s1          H_TM
 C   C4.5-s1p1        C_TM_PCC
Definition.of.Atomic.Species>

#
# Atoms
#

Atoms.Number         5
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU|FRAC
<Atoms.SpeciesAndCoordinates
 1  C      0.000000    0.000000    0.000000     2.0  2.0
 2  H     -0.889981   -0.629312    0.000000     0.5  0.5
 3  H      0.000000    0.629312   -0.889981     0.5  0.5
 4  H      0.000000    0.629312    0.889981     0.5  0.5
 5  H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit             Ang # Ang|AU
#<Atoms.UnitVectors
#  10.0   0.0   0.0
#   0.0  10.0   0.0
```

```
#   0.0   0.0  10.0
#Atoms.UnitVectors>

#
# SCF or Electronic System
#

scf.XcType                   LDA        # LDA|LSDA-CA|LSDA-PW|GGA-PBE
scf.SpinPolarization         off        # On|Off|NC
scf.ElectronicTemperature  100.0        # default=300 (K)
scf.energycutoff           120.0        # default=150 (Ry)
scf.maxIter                 60          # default=40
scf.EigenvalueSolver        cluster     # DC|GDC|Cluster|Band
scf.Kgrid                  1 1 1        # means n1 x n2 x n3
scf.Mixing.Type           rmm-diis      # Simple|Rmm-Diis|Gr-Pulay|Kerker|Rmm-Diisk
scf.Init.Mixing.Weight     0.30         # default=0.30
scf.Min.Mixing.Weight      0.001        # default=0.001
scf.Max.Mixing.Weight      0.400        # default=0.40
scf.Mixing.History          7           # default=5
scf.Mixing.StartPulay       4           # default=6
scf.criterion             1.0e-8        # default=1.0e-6 (Hartree)
scf.lapack.dste            dstevx       # dstevx|dstedc|dstegr,default=dstevx

#
# 1D FFT
#

1DFFT.NumGridK             900          # default=900
1DFFT.NumGridR             900          # default=900
1DFFT.EnergyCutoff        2500.0        # default=3600 (Ry)

#
# Orbital Optimization
#

orbitalOpt.Method           off         # Off|Unrestricted|Restricted
orbitalOpt.InitCoes     Symmetrical     # Symmetrical|Free
orbitalOpt.initPrefactor   0.1          # default=0.1
orbitalOpt.scf.maxIter      15          # default=12
orbitalOpt.MD.maxIter        7          # default=5
orbitalOpt.per.MDIter       20          # default=1000000
orbitalOpt.criterion      1.0e-4        # default=1.0e-4 (Hartree/borh)

#
# output of contracted orbitals
#

CntOrb.fileout               off        # on|off, default=off
Num.CntOrb.Atoms             1          # default=1
<Atoms.Cont.Orbitals
```

```
 1
Atoms.Cont.Orbitals>

#
# SCF Order-N
#

orderN.HoppingRanges        6.0        # default=5.0 (Ang)
orderN.NumHoppings           2         # default=2

#
# MD or Geometry Optimization
#

MD.Type                     nomd        # Nomd|Opt|NVE|NVT_VS|NVT_NH
                                      # Constraint_Opt|DIIS
MD.Opt.DIIS.History          4
MD.Opt.StartDIIS             5         # default=5

MD.maxIter                   1         # default=1
MD.TimeStep                 1.0        # default=0.5 (fs)
MD.Opt.criterion         1.0e-5        # default=1.0e-4 (Hartree/bohr)

#
# restart using a restart file, *.rst
#

scf.restart                 off        # on|off,default=off

#
# MO output
#

MO.fileout                   off       # on|off
num.HOMOs                     1        # default=1
num.LUMOs                     1        # default=1
MO.Nkpoint                    1        # default=1
<MO.kpoint
  0.0  0.0  0.0
MO.kpoint>

#
# DOS and PDOS
#

Dos.fileout                  off       # on|off, default=off
Dos.Erange               -10.0  10.0   # default = -20 20
Dos.Kgrid                  1  1  1     # default = Kgrid1 Kgrid2 Kgrid3

#
```

```
# output Hamiltonian and overlap
#

HS.fileout                    off       # on|off, default=off
```

6.2 Keywords

The specification of each keyword is given below. The list does not include all the keywords in OpenMX. Some of keywords will be explaned in each corresponding section.

File name

System.CurrrentDir
The output directory of output files is specified by this keyword. The default is './'.

System.Name
The file name of output files is specified by this keyword.

DATA.PATH
The path to the VPS and PAO directories can be specified in your input file by the following keyword:

```
DATA.PATH    ../DFT_DATA06/    # default=../DFT_DATA/
```

Both the absolute and relative specifications are available.

level.of.stdout
The amount of informations output to the standard output information in the middle of calculation is controlled by the keyword, level.of.stdout. In case of 'level.of.stdout=1', minimum informations. In case of 'level.of.stdout=2', additional informations together with the minimum output information. 'level.of.stdout=3' is for developers. The default is 1.

level.of.fileout
The amount of informations output to the files in the middle of calculation is controlled by the keyword 'level.of.fileout'. In case of 'level.of.fileout=0', minimum informations (no Gaussian cube and grid files). In case of 'level.of.fileout=1', standard output. In case of 'level.of.fileout=2', additional informations together with the standard output. The default is 1.

Definition of Atomic Species

Species.Number
The number of atomic species including the system is specified by the keyword 'Species.Number'.

Definition.of.Atomic.Species
Please specify atomic species by giving both the file name of pseudo-atomic basis orbitals and pseu-

dopotentials which must be existing in the directories 'DFT_DATA/PAO' and 'DFT_DATA/VPS', respectively. For example, they are specified as follows:

```
<Definition.of.Atomic.Species
 H   H4.0-s11p11        H_TM
 C   C4.5-s11p11        C_TM
Definition.of.Atomic.Species>
```

The beginning of the description must be '<Definition.of.Atomic.Species', and the last of the description must be 'Definition.of.Atomic.Species>'. In the first column, you can give any name to specify the atomic species. The name is used in the specification of atomic coordinates by 'Atoms.SpeciesAndCoordinates'. In the second column, the file name of the pseudo-atomic basis orbitals without the file extension and the number of primitive orbitals and contracted orbitals are given. Here we introduce an abbreviation of the basis orbital we used as H4.0-s11p11, where H4.0 indicates the file name of the pseudo-atomic basis orbitals without the file extension which must exist in the directory, 'DFT_DATA/PAO', s11 means that one optimized orbitals are constructed from one primitive orbitals for the s-orbital, which means no contraction. Also, in case of s11, corresponding to no contraction, you can use a simple notation 's1' instead of 's11'. Thus, 'H4.0-s1p1' is equivalent to 'H4.0-s11p11'. In the third column, the file name for the pseudopotentials without the file extension is given. Also the file must exist in the directory, 'DFT_DATA/VPS'. It can be possible to assign as the different atomic species for the same atomic element by specifying the different basis orbitals and pseudopotentials. For example, you can define the atomic species as follows:

```
<Definition.of.Atomic.Species
 H1   H4.0-s1p1         H_TM
 H2   H4.0-s2p2d1       H_TM_PCC
 C1   C4.5-s2p2         C_TM
 C2   C4.5-s2p2d2       C_TM_PCC
Definition.of.Atomic.Species>
```

The flexible definition may be useful for the decrease of computational efforts, in which only high level basis functions are used for atoms belonging to the essential part which determines the electric properties in the system, and lower level basis functions are used for atoms in the other inert parts.

Atoms

Atoms.Number
The total number of atoms in the system is specified by the keyword 'Atoms.Number'.

Atoms.SpeciesAndCoordinates.Unit
The unit of the atomic coordinates is specified by the keyword 'Atoms.SpeciesAndCoordinates.Unit'. Please specify 'Ang' when you use the unit of Angstrom, and 'AU' when the unit of atomic unit. The fractional coordinate is also available by 'FRAC'. Then, please specify the coordinates spanned by **a**, **b**, and **c**-axes given in 'Atoms.UnitVectors'. In the fractional coordinates, the coordinates can range from -0.5 to 0.5, and the coordinates beyond its range will be automatically adjusted after reading

the input file

Atoms.SpeciesAndCoordinates
The atomic coordinates and the number of spin charge are given by the keyword 'Atoms.SpeciesAndCoordinates' as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.000000    0.000000    0.000000     2.0  2.0
  2   H     -0.889981   -0.629312    0.000000     0.5  0.5
  3   H      0.000000    0.629312   -0.889981     0.5  0.5
  4   H      0.000000    0.629312    0.889981     0.5  0.5
  5   H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
```

The beginning of the description must be '<Atoms.SpeciesAndCoordinates', and the last of the description must be 'Atoms.SpeciesAndCoordinates>'. The first column is a sequential serial number for identifying atoms. The second column is given to specify the atomic species which must be given in the first column of the specification of the keyword 'Definition.of.Atomic.Species' in advance. In the third, fourth, and fifth columns, x-, y-, z-coordinates are given. The sixth and seventh columns give the number of up and down initial spin charges for each atom, respectively. The sum of up and down charges must be the number of valence electrons for the atomic element. When you calculate spin-polarized systems using 'LSDA-CA' or 'LSDA-PW', you can give the initial spin charges for each atom, which might be those of the ground state, to accelerate the SCF convergence.

Atoms.UnitVectors.Unit
The unit of the vectors for the unit cell is specified by the keyword 'Atoms.UnitVectors.Unit'. Please specify Ang when you use the unit of Angstrom, and AU when the unit of atomic unit.

Atoms.UnitVectors
The vectors, **a**, **b**, and **c** of the unit cell are given by the keyword 'Atoms.UnitVectors' as follows:

```
<Atoms.UnitVectors
  10.0   0.0   0.0
   0.0  10.0   0.0
   0.0   0.0  10.0
Atoms.UnitVectors>
```

The beginning of the description must be '<Atoms.UnitVectors', and the last of the description must be 'Atoms.UnitVectors>'. The first, second, and third rows correspond to the vectors, **a**, **b**, and **c** of the unit cell, respectively. If the keyword is absent in the cluster calculation, a unit cell is automatically determined so that the isolated system can not overlap with the image systems in the repeated cells. See also the Section 'Automatic determination of the cell size'.

SCF or Electronic System

scf.XcType

The keyword 'scf.XcType' specifies the exchange-correlation potential. Currently, 'LDA', 'LSDA-CA', 'LSDA-PW', and 'GGA-PBE' are available, where 'LSDA-CA' is the local spin density functional of Ceperley-Alder [2], 'LSDA-PW' is the local spin density functional of Perdew-Wang, in which the gradient of density is set to zero in their GGA formalism [4]. Note: 'LSDA-CA' is faster than 'LSDA-PW'. 'GGA-PBE' is a GGA functional proposed by Perdew et al [5].

scf.SpinPolarization
The keyword 'scf.SpinPolarization' specifies the non-spin polarization or the spin polarization for the electronic structure. If the calculation for the spin polarization is performed, then specify 'ON'. If the calculation for the non-spin polarization is performed, then specify 'OFF'. When you use LDA for the keyword 'scf.XcType' the keyword 'scf.SpinPolarization' must be off. In addition to these options, 'NC' is supported for the non-collinear DFT calculation. For this calculation, see also the Section 'Non-collinear DFT'.

scf.partialCoreCorrection
The keyword 'scf.partialCoreCorrection' is a flag for a partial core correction (PCC) in calculations of exchange-correlation energy and potential. 'ON' means that the PCC is made, and 'OFF' is none. In any cases, the flag should be 'ON', since pseudopotentials generated with PCC should be used with the PCC, and also the PCC does not affect the result for pseudopotentials without the PCC because of zero PCC charge in this case.

scf.Hubbard.U
In case of LDA+U calculations, the keyword 'scf.Hubbard.U' should be switched on (on|off). The default is off.

scf.Hubbard.Occupation
In the LDA+U method, three occupation number operators 'onsite', 'full', and 'dual', are available which can be specified by the keyword 'scf.Hubbard.Occupation'.

Hubbard.U.values
An effective U-value on each orbital of species is defined by the keyword as follows:

```
<Hubbard.U.values                  #  eV
 Ni  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 4.0 2d 0.0
 O   1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hubbard.U.values>
```

The beginning of the description must be '<Hubbard.U.values', and the last of the description must be 'Hubbard.U.values>'. For all the basis orbitals specified by the 'Definition.of.Atomic.Species', you have to give an effective U-value in above format. The '1s' and '2s' mean the first and second s-orbital, and the number behind '1s' is the effective U-value for the first s-orbital. The same rule is applied to p- and d-orbitals.

scf.Constraint.NC.Spin
The keyword 'scf.Constraint.NC.Spin' should be switched on (on|off) when the constraint DFT for the non-collinear spin orientation is performed.

scf.Constraint.NC.Spin.v
The keyword 'scf.Constraint.NC.Spin.v' gives a prefactor (eV) of the penalty functional in the con-

straint DFT for the non-collinear spin orientation.

scf.ElectronicTemperature
The electronic temperature (K) is given by the keyword 'scf.ElectronicTemperature'. The default is 300 (K).

scf.energycutoff
The keyword 'scf.energycutoff' specifies the cutoff energy which is used in the calculation of matrix elements associated with difference charge Coulomb potential and exchange-correlation potential and the solution of Poisson's equation using fast Fourier transform (FFT). The default is 150 (Ryd).

scf.Ngrid
The keyword 'scf.Ngrid' gives the number of grids to discretize the a-, b-, and c-axes. Although 'scf.energycutoff' is usually used for the discretization, if you specify the number of grids by 'scf.Ngrid', they are used for the discretization instead of those by 'scf.energycutoff'.

scf.maxIter
The maximum number of SCF iterations is specified by the keyword 'scf.maxIter'. The SCF loop is terminated at the number specified by 'scf.maxIter' even when a convergence criterion is not satisfied. The default is 40.

scf.EigenvalueSolver
The solution method for the eigenvalue problem is specified by the keyword 'scf.EigenvalueSolver'. An O(N) divide-conquer method 'DC', an O(N) generalized divide-conquer method 'GDC', O(N) Krylov subspace method 'Krylov', the cluster calculation 'Cluster', and the band calculation 'Band' are available.

scf.lapack.dste
The keyword specifies a lapack routine which is used to evaluate eigenvalues and eigenvectors of the tridiagonalized matrix in the cluster, band, and O(N) calculations. Three lapack routines, dstegr, dstedc, and dstevx are available. For further details, see the Section 'Selection of lapack routine'. The default is 'dstevx', and we strongly recommend for to use dstevx for both the stability and efficiency, since it is possible to calculate only eigenvectors of occupied and unoccupied but low energy exited states instead of calculating all the eigenvectors for saving the computational time.

scf.Kgrid
When you specify the band calculation 'Band' for the keyword 'scf.EigenvalueSolver', then you need to give a set of numbers (n1,n2,n3) of grids to discretize the first Brillouin zone in the k-space by the keyword 'scf.Kgrid'. For the reciprocal vectors $\tilde{\mathbf{a}}$, $\tilde{\mathbf{b}}$, and $\tilde{\mathbf{c}}$ in the k-space, please provide a set of numbers (n1,n2,n3) of grids as n1 n2 n3.

scf.Mixing.Type
A mixing method of the electron density (or the density matrix) to generate an input electron density at the next SCF step is specified by keyword, scf.Mixing.Type. A simple mixing method ('Simple'), 'GR-Pulay' method (Guaranteed-Reduction Pulay method) [30], 'RMM-DIIS' method [31], 'Kerker' method [32], and 'RMM-DIISK' method [31] are available. The simple mixing method used here is modified to accelerate the convergence, referring to a convergence history. When 'GR-Pulay', 'RMM-DIIS', 'Kerker', or 'RMM-DIISK' is used, the following recipes are helpful to obtain faster convergence

of SCF calculations:

- Use a rather larger value for 'scf.Mixing.StartPulay'. Before starting the Pulay like mixing, achieve a convergence at some level. An appropriate value may be 10 to 30 for 'scf.Mixing.StartPulay'.
- Use a rather larger value for 'scf.ElectronicTemperature' in case of metallic systems. When 'scf.ElectronicTemperature' is small, numerical instabilities appear often.
- Look for an appropriate value for scf.Mixing.History. In most cases, 'scf.Mixing.History=7' can be a good value.

Among these mixing schemes, the robustest one might be 'RMM-DIISK'.

scf.Init.Mixing.Weight
The keyword, scf.Init.Mixing.Weight, gives a mixing weight first used by the simple mixing method, the GR-Pulay method, the RMM-DIIS, the Kerker, and the RMM-DIISK.
The valid range is 0 <scf.Init.Mixing.Weight< 1. The default is 0.3.

scf.Min.Mixing.Weight
The keyword 'scf.Min.Mixing.Weight' gives the lower limit of a mixing weight in the simple and Kerker mixing methods. The default is 0.001.

scf.Max.Mixing.Weight
The keyword 'scf.Max.Mixing.Weight' gives the upper limit of a mixing weight in the simple and Kerker mixing methods. The default is 0.4.

scf.Kerker.factor
The keyword gives a Kerker factor which is used in the Kerker and RMM-DIISK mixing schemes. The default is 1.0. For further details, See the Section 'SCF convergence'.

scf.Mixing.History
In the GR-Pulay method [30], the RMM-DIIS method [31], the Kerker [32], and the RMM-DIISK [31], the input electron density at the next SCF step is estimated based on the output electron densities in the several previous SCF steps. The keyword 'scf.Mixing.History' specifies the number of previous SCF steps which are utilized in the estimation. For example, if 'scf.Mixing.History' is specified to be 3, and when the SCF step is 6th, the electron densities of 5, 4, and 3 SCF steps are taken into account. The default is 6.

scf.Mixing.StartPulay
The SCF step which starts the GR-Pulay, the RMM-DIIS, the Kerker, and the RMM-DIISK methods is specified by the keyword 'scf.Mixing.StartPulay'. The SCF steps before starting these Pulay-type methods are then performed by the simple or Kerker mixing methods. The default is 6.

scf.Mixing.EveryPulay
The residual vectors in the Pulay-type mixing schemes tend to become linearly dependent each other as the mixing steps accumulate, and the linear dependence among the residual vectors makes the convergence difficult. A way of avoiding the linear dependence is to do the Pulay-type mixing occasionally during the Kerker mixing. With this prescription, you can specify the frequency using the keyword 'scf.Mixing.EveryPulay'. For example, in case of 'scf.Mixing.EveryPulay=5', the Pulay-mixing is made at every five SCF iteration, while Kerker-type mixing is used at the other steps.

'scf.Mixing.EveryPulay=1' corresponds to the conventional Pulay-type mixing. It is noted that the keyword 'scf.Mixing.EveryPulay' is supported for only 'RMM-DIISK', and the default value is 5.

scf.criterion
The keyword 'scf.criterion' specifies a convergence criterion (Hartree) for the SCF calculation. The SCF iteration is ended when a condition, dUele<scf.criterion, is satisfied, where dUele is defined as the absolute deviation between the eigenvalue energy at the current and previous SCF steps. The default is 1.0e-6 (Hartree).

scf.Electric.Field
The keyword 'scf.Electric.Field' gives a uniform external electric field given by a sawtooth waveform. For example, when an electric field of 1.0 GV/m (10^9 V/m) is applied along the a-axis, specify in your input file as follows:

```
scf.Electric.Field   1.0 0.0 0.0   # default=0.0 0.0 0.0 (GV/m)
```

The sign of electric field is taken as that applied to electrons. The default is 0.0 0.0 0.0.

scf.system.charge
The keyword 'scf.system.charge' gives the amount of the electron and hole dopings. The plus and minus signs correspond to hole and electron dopings, respectively. The default is 0.

scf.SpinOrbit.Coupling
When the spin-orbit coupling is included, the keyword should be 'ON', otherwise please set in 'OFF'. In case of the inclusion of the spin-orbit coupling, you have to use j-dependent pseudopotentials. See also the Section 'Relativistic effects' as for the j-dependent pseudopotentials.

1D FFT

1DFFT.EnergyCutoff
The keyword '1DFFT.EnergyCutoff' gives the energy range to tabulate the Fourier transformed radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials. The default is 3600 (Ryd).

1DFFT.NumGridK
The keyword '1DFFT.NumGridK' gives the the number of radial grids in the k-space. The values of the Fourier transformation for radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials are tabulated on the grids, ranging from zero to 1DFFT.EnergyCutoff, as a function of radial axis in the k-space. The default is 900.

1DFFT.NumGridR
The keyword '1DFFT.NumGridR' gives the the number of radial grids in the real space which is used in the numerical grid integrations of the Fourier transformation for radial functions of pseudo-atomic orbitals and of the projectors for non-local potentials. The default is 900.

Orbital Optimization

orbitalOpt.Method

The keyword 'orbitalOpt.Method' specifies a method for the orbital optimization. When the orbital optimization is not performed, then choose 'OFF'. When the orbital optimization is performed, the following three options are available, the unrestricted optimization 'Unrestricted', the restricted optimization 'Restricted', and orbital optimization restricted to species 'Species'. In the 'Unrestricted', the radial functions of basis orbitals are optimized without any constraint. Thus, all the radial functions could differ from each other, which could depend on the following indices, atomic number, angular moment quantum number, magnetic quantum number, and orbital multiplicity. In the 'Restricted', the radial functions of basis orbitals are optimized with a constraint that the radial wave function R is independent on the magnetic quantum number. We prefer 'Restricted' to 'Unrestricted', since the restricted optimization guarantees the rotational invariance of the total energy. In the 'Species', basis orbitals in atoms with the same species name, that you define in 'Definition.of.Atomic.Species', are optimized as the same orbitals. If you want to assign the same orbitals to atoms with almost the same chemical environment, and optimize these orbitals, this scheme is useful.

orbitalOpt.InitCoes

The keyword 'orbitalOpt.InitCoes' specifies a way for setting the initial contraction coefficients. If 'Symmetrical' is chosen, then the initial contraction coefficients is symmetrically given so that the total energy can be invariant for the rotation of system. If 'Free' is chosen, then the initial contraction coefficients could be unsymmetrical.

orbitalOpt.scf.maxIter

The maximum number of SCF iterations in the orbital optimization is specified by the keyword 'orbitalOpt.scf.maxIter'.

orbitalOpt.MD.maxIter

The maximum number of iterations for the orbital optimization is specified by the keyword 'orbitalOpt.MD.maxIter'. The iteration loop for the orbital optimization is terminated at the number specified by 'orbitalOpt.MD.maxIter' even when a convergence criterion is not satisfied.

orbitalOpt.criterion

The keyword 'orbitalOpt.criterion' specifies a convergence criterion ((Hartree/borh)^2) for the orbital optimization. The iterations loop is finished when a condition, Norm of derivatives<orbitalOpt.criterion, is satisfied.

CntOrb.fileout

If you want to output the optimized radial orbitals to files, then the keyword 'CntOrb.fileout' must be ON.

Num.CntOrb.Atoms

The keyword 'Num.CntOrb.Atoms' gives the number of atoms whose optimized radial orbitals are output to files.

Atoms.Cont.Orbitals

The keyword 'Atoms.Cont.Orbitals' specifies the atom number, which was given by the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates' for the output of optimized orbitals as follows:

```
<Atoms.Cont.Orbitals
 1
 2
Atoms.Cont.Orbitals>
```

The beginning of the description must be '<Atoms.Cont.Orbitals', and the last of the description must be 'Atoms.Cont.Orbitals>'. The number of lines should be consistent with the number specified in the keyword 'Atoms.Cont.Orbitals'. For example, the name of files are C_1.pao and H_2.pao, where the symbol corresponds to that given by the first column in the specification of the keyword 'Definition.of.Atomic.Species' and the number after the symbol means that of the first column in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. These outout files, C_1.pao and H_2.pao, can be an input data for pseudo-atomic orbitals as it is.

SCF Order-N

orderN.HoppingRanges
The keyword 'orderN.HoppingRanges' defines the radius of a sphere which is centered on each atom. The logically truncated cluster for each atom is constructed for the atom inside the sphere in the DC, GDC, and Krylov subspace methods.

orderN.NumHoppings
The keyword 'orderN.NumHoppings' gives the number, n, of hopping which is required to construct the logically truncated cluster in the DC, GDC, and Krylov subspace methods. The cluster of size, n, is defined by all the neighbors that can be reached by n hops, where the cutoff distance is given by the sum of the cutoff distances r_1 and r_2 of basis orbitals belonging to atoms 1 and 2.

orderN.KrylovH.order
The dimension of Krylov subspace of Hamiltonian in each truncated cluster is given by the 'orderN.KrylovH.order'.

orderN.KrylovS.order
In case of 'orderN.Exact.Inverse.S=off', the inverse is approximated by a Krylov subspace method for the inverse, where the dimension of the Krylov subspace of overlap matrix in each truncated cluster is given by the keyword 'orderN.KrylovS.order'. The default value is orderN.KrylovH.order×4.

orderN.Exact.Inverse.S
In case of 'orderN.Exact.Inverse.S=on', the inverse of overlap matrix for each truncated cluster is exactly evaluated. Otherwise, see the keyword 'orderN.KrylovS.order'. The default is 'on' (on|off).

orderN.Recalc.Buffer
In case of 'orderN.Recalc.Buffer=on', the buffer matrix is recalculated at every SCF step. Otherwise, the buffer matrix is calculated at the first SCF step, and fixed at subsequent SCF steps. The default is 'off' (on|off).

orderN.Expand.Core
In case of 'orderN.Expand.Core=on', the core region is defined by atoms within a sphere with radius

of $1.2 \times r_{\min}$, where $r_{\min}$ is the distance between the central atom and the nearest atom. In case of 'orderN.Expand.Core=off', the central atom is considered as the core region. The default is 'on' (on|off).

MD or Geometry Optimization

MD.Type
Please specify the type of the molecular dynamics calculation or the geometry optimization. Currently, NO MD (Nomd), MD with the NVE ensemble (NVE), MD with the NVT ensemble by a velocity scaling scheme (NVT_VS)[17], MD with the NVT ensemble by a Nose-Hoover scheme (NVT_NH) [18], the geometry optimization by the steepest decent (SD) method (Opt), and DIIS optimization method (DIIS) are available.

MD.Fixed.XYZ
In the geometry optimization and the molecular dynamics simulations, it is possible to separately fix the x-, y-, and z-coordinates of the atomic position to the initial position in your input file by the following keyword:

```
<MD.Fixed.XYZ
  1  1 1 1
  2  1 0 0
MD.Fixed.XYZ>
```

The example is for a system consisting of two atoms. If you have N atoms, then you have to provide N-th rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, 4th columns are flags for the x-, y-, z-coordinates. '1' means that the coordinate is fixed, and '0' relaxed. It should be noted that the definition of the switch is opposite compared to the previous constraint schemes. In above example, the x-, y-, z-coordinates of the atom '1' are fixed, only the x-coordinate of the atom '2' is fixed. The default setting is that all the coordinates are relaxed. The fixing of atomic positions are valid all the geometry optimizers and molecular dynamics schemes.

MD.maxIter
The keyword 'MD.maxIter' gives the number of MD iterations.

MD.TimeStep
The keyword 'MD.maxIter' gives the time step (fs).

MD.Opt.criterion
When 'Opt' is chosen for the keyword 'MD.Type', then the keyword 'MD.Opt.criterion' specifies a convergence criterion (Hartree/bohr). The geometry optimization is finished when a condition, the maximum force is smaller than 'MD.Opt.criterion', is satisfied.

MD.Opt.DIIS.History
The keyword 'MD.Opt.DIIS.History' gives the number of previous steps to estimate the optimized structure used in the geometry optimization by 'DIIS'. The default value is 4.

MD.Opt.StartDIIS
The geometry optimization step which starts 'DIIS' is specified by the keyword 'MD.Opt.StartDIIS'. The geometry optimization steps before starting DIIS type methods is performed by the steepest decent method. The default value is 5.

MD.TempControl
The keyword specifies temperature for atomic motion in MD of the NVT ensembles. In 'NVT_VS', the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  3
  100   2  1000.0  0.0
  400  10   700.0  0.4
  700  40   500.0  0.7
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '3' gives the number of the following lines to control the temperature. In this case you can see that there are three lines. Following the number '3', in the consecutive lines the first column means the number of MD steps and the second column gives interval of MD steps which determine ranges of MD steps and intervals at which the velocity scaling is made. For above example, a velocity scaling is performed at every two MD steps until 100 MD steps, at every 10 MD steps from 100 to 400 MD steps, and at every 40 MD steps from 400 to 700 MD steps. The third and fourth columns give a given temperature (K) and a scaling parameter α in the interval. For further details see the Section 'Molecular dynamics'. On the other hand, in NVT_NH, the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  4
  1    1000.0
  100  1000.0
  400   700.0
  700   600.0
MD.TempControl>
```

The beginning of the description must be '<MD.TempControl', and the last of the description must be 'MD.TempControl>'. The first number '4' gives the number of the following lines to control the temperature. In this case you can see that there are four lines. Following the number '4', in the consecutive lines the first and second columns give the number of MD steps and a given temperature for nuclear motion. The temperature between the interval is given by a linear interpolation.

NH.Mass.HeatBath
In 'NVT_NH', a mass of heat bath is given by this keyword. The default mass is 20, where we use a unit that the weight of a proton is 1.0.

MD.Init.Velocity

For molecular dynamics simulations, it is possible to provide the initial velocity of each atom by the following keyword:

```
<MD.Init.Velocity
 1    3000.000  0.0  0.0
 2   -3000.000  0.0  0.0
MD.Init.Velocity>
```

The example is for a system consisting of two atoms. If you have N atoms, then you have to provide N-th rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are x-, y-, and z-components of the velocity of each atom. The unit of the velocity is m/s. The keyword 'MD.Init.Velocity' is compatible with the keyword 'MD.Fixed.XYZ'.

Band dispersion

Band.dispersion
When you evaluate the band dispersion, please specify the keyword 'Band.dispersion' ON.

Band.KPath.UnitCell
The keyword 'Band.KPath.UnitCell' gives unit vectors, which are used in the calculation of the band dispersion, as follows:

```
<Band.KPath.UnitCell
 3.56 0.0 0.0
 0.0 3.56 0.0
 0.0 0.0 3.56
Band.KPath.UnitCell>
```

The beginning of the description must be '<Band.KPath.UnitCell', and the last of the description must be 'Band.KPath.UnitCell>'. If 'Band.KPath.UnitCell' exist, the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If 'Band.KPath.UnitCell' does not exist, the reciprocal lattice vectors, which are calculated by the unit vectors specified in 'Atoms.UnitVectors', is employed for the calculation of the band dispersion. In case of fcc, bcc, base centered cubic, and trigonal cells, the reciprocal lattice vectors for the calculation of the band dispersion should be specified using the keyword 'Band.KPath.UnitCell' based on the consuetude in the band calculations.

Band.Nkpath
The keyword 'Band.Nkpath' gives the number of paths for band dispersion.

Band.kpath
The keyword 'Band.kpath' specifies the paths of band dispersion as follows:

```
<Band.kpath
```

```
  15  0.0 0.0 0.0   1.0 0.0 0.0   g X
  15  1.0 0.0 0.0   1.0 0.5 0.0   X W
  15  1.0 0.5 0.0   0.5 0.5 0.5   W L
  15  0.5 0.5 0.5   0.0 0.0 0.0   L g
  15  0.0 0.0 0.0   1.0 1.0 0.0   g X
Band.kpath>
```

The beginning of the description must be '<Band.kpath', and the last of the description must be 'Band.kpath>'. The number of lines should be consistent with 'Band.Nkpath'. The first column is the number of grids at which eigenvalues are evaluated on the path. The following (n1, n2, n3) and (n1', n2', n3'), spanned by the reciprocal lattice vectors, specifies the start and end points of the path in the first Brillouin zone. If 'Band.KPath.UnitCell' exists, the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If 'Band.KPath.UnitCell' does not exist, the reciprocal lattice vectors, which are calculated by the unit vectors specified in 'Atoms.UnitVectors' is employed for the calculation of the band dispersion. The final two alphabets give the name of the start and end points of the path.

Restarting

scf.restart
If you want to restart the SCF calculation using a previous file '*_rst/*' which should be generated in advance, then set the keyword 'scf.restart' to 'ON'.

Outout of molecular orbitals (MOs)

MO.fileout
If you want to output molecular orbitals (MOs) to files, then set the keyword 'MO.fileout' to 'ON'.

num.HOMOs
The keyword 'num.HOMOs' gives the number of the highest occupied molecular orbitals (HOMOs) that you want to output to files

num.LUMOs
The keyword 'num.LUMOs' gives the number of the highest occupied molecular orbitals (LUMOs) that you want to output to files

MO.Nkpoint
When you have specified 'MO.fileout=ON' and scf.EigenvalueSolver=Band, the keyword 'MO.Nkpoint' gives the number of the k-points at which you want to output MOs to files

MO.kpoint
The keyword 'MO.kpoint' specifies the k-point, at which MOs are evaluated for the output to files, as follows:

```
<MO.kpoint
```

```
  0.0  0.0  0.0
MO.kpoint>
```

The beginning of the description must be '<MO.kpoint', and the last of the description must be 'MO.kpoint>'. The k-points are specified by (n1, n2, n3) which is spanned by the reciprocal lattice vectors, where the the reciprocal lattice vectors are determined in the same way as Band.kpath

DOS and PDOS

Dos.fileout
If you want to evaluate density of states (DOS) and projected local density of states (LDOS), please set in 'Dos.fileout=ON'.

Dos.Erange
The keyword, Dos.Erange, determines the energy range for the DOS calculation as

```
Dos.Erange                -10.0  10.0
```

The first and second values are the lower and upper bounds of the energy range (eV) for the DOS calculation, respectively.

Dos.Kgrid
The keyword, Dos.Kgrid, gives a set of numbers (n1,n2,n3) of grids to descretize the first Brillouin zone in the k-space, which is used in the DOS calculation.

Interface for developers

HS.fileout
If you want to use Kohn-Sham Hamiltonian, overlap, and density matrices, please set in 'HS.fileout=ON'. Then, these data are stored to *.scfout in a binary form. The utilization of these data is illustrated in the Section, Interface for developers.

Voronoi charge

Voronoi.charge
If you want to calculate Voronoi charges, then set the keyword 'Voronoi.charge' in 'ON'. The result is found in '*.out'.

7 Output files

In case of 'level.of.fileout=0', the following files are generated, where * means System.Name.

- *.out

  The history of SCF calculations, the history of geometry optimization, Mulliken charges, the total energy, and the dipole moment.

- *.memory0, *.memory1,...,*.memory#,..

  Analysis of the size of memory used in processor #.

- *.xyz

  The final geometrical structure obtained by MD or the geometry optimization, which can be read in gOpenMol or xmakemol.

- *.bulk.xyz

  If scf.EigenvalueSolver=Band, atomic coordinates including atoms in copied cells are output, which can be read in gOpenMol or xmakemol.

- *_rst/

  The directory storing restart files.

- *.md

  Geometrical coordinates at every MD step, which can be read in gOpenMol or xmakemol.

- *.md2

  Geometrical coordinates at the final MD step with the species names that you specified .

- *.cif

  Initial geometrical coordinates in the cif format suited for Material Studio.

- *.ene

  Values computed at every MD step. The values are found in the routine, 'iterout.c'

In case of 'level.of.fileout=1', the following Gaussian cube files are generated, in addition to files generated in 'level.of.fileout=0', where * means System.Name.

- *.tden.cube

  Total electron density in a form of the Gaussian cube format

- *.sden.cube

  If spin-polarized calculations using LSDA-CA, LSDA-PW, or GGA-PBE are performed, then spin electron density is output in a Gaussian cube format.

- *.v0.cube

  The Kohn-Sham potential excluding the non-local potential for up-spin in a Gaussian cube format.

- *.v1.cube

  The Kohn-Sham potential excluding the non-local potential for down-spin in a Gaussian cube format in the spin-polarized calculation.

- *.vhart.cube

  The Hartree potential in a Gaussian cube format.

In case of level.of.fileout=2, the following files are generated in addition to files generated in level.of.fileout=1, where * means System.Name.

- *.vxc0.cube

  The exchange-correlation potential for up-spin in a Gaussian cube format.

- *.vxc1.cube

  The exchange-correlation potential for down-spin in a Gaussian cube format.

- *.grid

  The real space grids which are used numerical integrations and the solution of Poisson's equation.

If 'MO.fileout=ON' and 'scf.EigenvalueSolver=Cluster', the following files are also generated:

- *.homo0_0.cube, *.homo0_1.cube, ...

  The HOMOs are output in a Gaussian cube format. The first number below homo means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond HOMO, HOMO-1, and HOMO-2, respectively.

- *.lumo0_0.cube, *.lumo0_1.cube, ...

  The LUMOs are output in a Gaussian cube format. The first number below lumo means a spin state (up=0, down=1). The second number specifies the eigenstates, i.e., 0, 1, and 2 correspond LUMO, LUMO+1, and LUMO+2, respectively.

If MO.fileout=ON and scf.EigenvalueSolver=Band, the following files are also generated:

- *.homo0_0_0_r.cube, *.homo1_0_1_r.cube, ... *.homo0_0_0_i.cube, *.homo1_0_1_i.cube, ...

  The HOMOs are output in a Gaussian cube format. The first number below homo means the k-point number, which is specified by the keyword 'MO.kpoint'. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond HOMO, HOMO-1, and HOMO-2, respectively. The 'r' and 'i' mean the real and imaginary parts of the wave function.

- *.lumo0_0_0_r.cube, *.lumo1_0_1_r.cube, ... *.lumo0_0_0_i.cube, *.lumo1_0_1_i.cube, ...

  The LUMOs are output in a Gaussian cube format. The first number below lumo means the k-point number, which is specified in the keyword, MO.kpoint. The second number is a spin state (up=0, down=1). The third number specifies the eigenstates, i.e., 0, 1, and 2 correspond LUMO, LUMO+1, and LUMO+2, respectively. The 'r' and 'i' mean the real and imaginary parts of the wave function.

If 'Band.Nkpath' is not 0 and 'scf.EigenvalueSolver=Band', the following file is also generated:

- *.Band

  A data file for making the band dispersion.

If 'Dos.fileout=ON', the following files are also generated:

- *.Dos.val

  A data file of eigenvalues for calculating the density of states.

- *.Dos.vec

  A data file of eigenvectors for calculating the density of states.

If scf.SpinPolarization=NC and level.of.fileout=1 or 2, the following files are also generated:

- *.nco.txt

  A vector file which stores a non-collinear orbital moment projected on each atom by means of Mulliken analysis, which can be visualized using 'Plot Vector File' in gOpenMol.

- *.nc.txt

  A vector file which stores a non-collinear spin moment projected on each atom by means of Mulliken analysis, which can be visualized using 'Plot Vector File' in gOpenMol.

- *.nc.txt

  A vector file which stores a non-collinear spin moment on real space grids, which can be visualized using 'Plot Vector File' in gOpenMol.

8 Functional

In OpenMX, local density approximations (LDA, LSDA) [2, 3, 4] and a generalized gradient approximation (GGA) [5] to the exchange-correlation potential are used. Using a keyword 'scf.XcType', you can choose the exchange-correlation type.

```
scf.XcType                  LDA         # LDA|LSDA-CA|LSDA-PW|GGA-PBE
```

Currently, 'LDA', 'LSDA-CA', 'LSDA-PW', and 'GGA-PBE' are available, where 'LSDA-CA' is the local spin density functional of Ceperley-Alder [2], 'LSDA-PW' is the local spin density functional of Perdew-Wang, in which the gradient of density is set in zero in their GGA formalism [4]. Note: 'LSDA-CA' is faster than 'LSDA-PW'. 'GGA-PBE' is GGA proposed by Perdew et al [5]. The GGA is implemented by the first order finite difference in real space. The relevant keyword to specify the spin (un)polarized and non-collinear calculations is 'scf.SpinPolarization'.

```
scf.SpinPolarization        off         # On|Off|NC
```

If the calculation for the spin polarization is performed, then specify 'ON'. If the calculation for the non-spin polarization is performed, then specify 'OFF'. When you use LDA for the keyword 'scf.XcType' the keyword 'scf.SpinPolarization' must be off. In addition to these options, 'NC' is supported for the non-collinear DFT calculation. For this calculation, see also the Section 'Non-collinear DFT'.

LDA+U functionals are also available. For the details, see the Section 'LDA+U'.

9 Basis sets

9.1 Primitive basis function

The primitive basis functions used in OpenMX are the ground and exited states of a pseudo-atom with a confinement pseudopotential [23] shown in Fig. 1. The functions are numerical table function stored in a file of which file extension is 'pao'. You will see that the ground state is nodeless and the first exited state has one node, and the number of nodes increases in the further excited states. The one-particle Kohn-Sham functions are expressed by the linear combination of the atomic type basis functions where each basis function is the product of the radial function and a real spherical harmonics function. The selection of the basis sets is one of important issues to perform reliable calculations. However, the use of a large number of basis orbitals requires an extensive computational resource such as memory size and computational time. So, users are required to use small but accurate basis sets at some level which depends on calculated properties and computer power. As a criterion of the selection, we offer databases (http://www.openmx-square.org/) of basis sets, and convergence properties of the total energy and the equilibrium bond length of dimer molecules. As an example, the convergence properties of a carbon dimer are shown in Fig. 2. You might find that the convergence properties are determined by two simple parameter: a cutoff radius of basis orbitals and the number of basis orbitals, which suggests a large cutoff radius and number of basis orbitals provide more accurate results with high computational demands. The database suggests that basis orbitals with a higher angular momentum are needed to achieve the sufficient convergence for elements, such as F and Cl, in the right side of the periodic table, and that a large cutoff radius of basis orbitals should be used for elements, such as Li and Na, in the left side of the periodic table. If you want to use basis orbitals

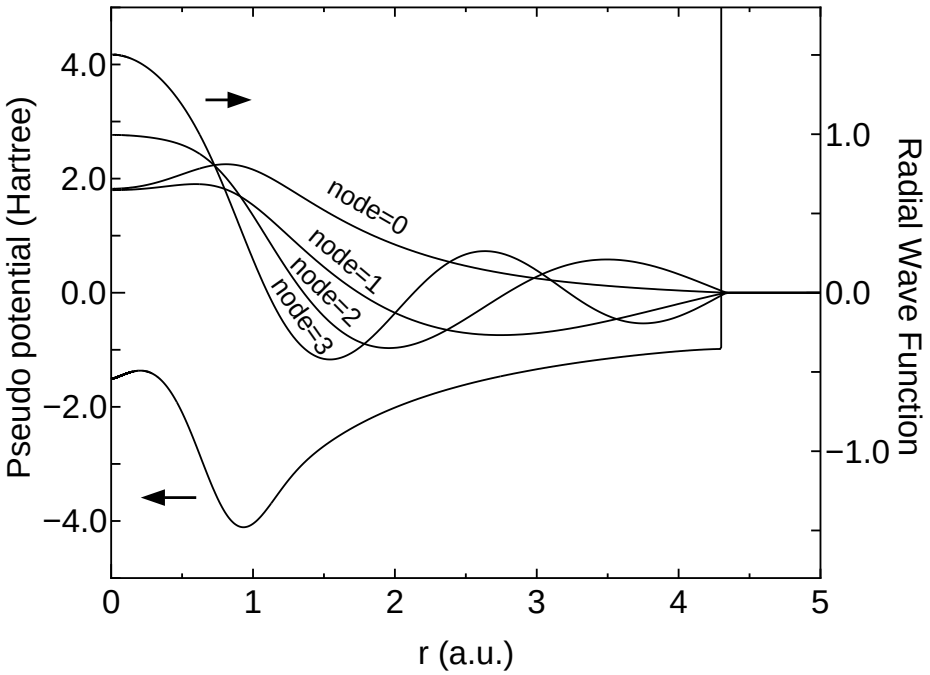


Figure 1: Primitive basis functions for s-orbitals of a carbon pseudo-atom with a confinement pseudopotential.

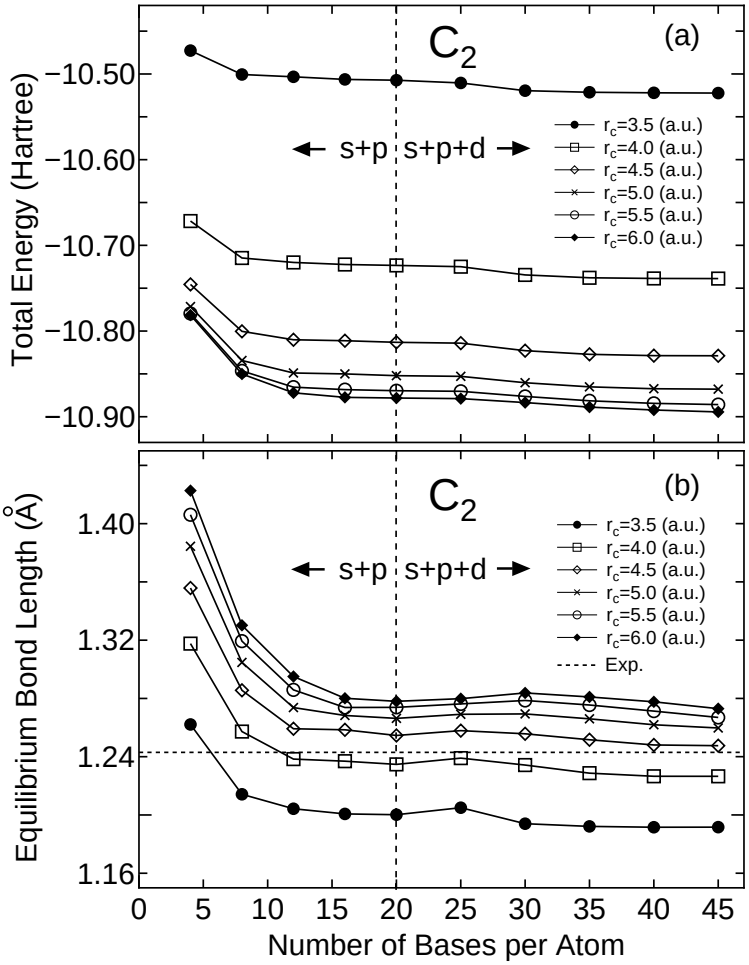


Figure 2: Convergence properties of (a) the total energy and (b) the equilibrium bond length for a carbon dimer with respect to basis set

stored in the database, then copy them to the directory, 'openmx3.5/DFT_DATA/PAO/'. You can freely utilize these data in terms of GNU-GPL, but we cannot offer any warranty on these data. Also it is possible to generate pseudo-atomic basis functions using ADPACK by yourself. The basis set is specified by a keyword 'Definition.of.Atomic.Species' as follows:

```
<Definition.of.Atomic.Species
  H   H4.0-s2p1        H_LDA
  C   C4.5-s2p2        C_LDA
Definition.of.Atomic.Species>
```

where an abbreviation 'H4.0-s2p1' of the basis function is introduced, where H4.0 indicates the file name of the pseudo-atomic basis orbitals without the file extension which must exist in the directory 'DFT_DATA/PAO', and 's2p1' means that two s-orbitals and one p-orbital in the file are used. In this case, totally eight basis functions (2x1+1x3=5) are assigned for 'H'. See also the Section 'Input file' for the details.

9.2 Optimized basis function

Starting from the primitive basis functions, you can optimize the radial shape variationally so that the accuracy can be increased. See the details in the Section 'Orbital optimization'.

9.3 Empty atom scheme

The primitive basis and optimized basis functions are usually assigned to atoms. Moreover, it is possible to assign basis functions in any vacant region using an 'empty' atom. You will find the empty atom 'E' in the database (http://www.openmx-square.org/). Using the basis functions and pseudopotential, though the pseudopotential is a flat zero potential, you can put the basis functions at any place independently of atomic position. The empty atom scheme enables us to treat a vacancy state and a nearly free electron state on metal surfaces within the LCAO method. As an example, a calculation of a F-center in NaCl with a Cl vacancy is shown in Fig. 3. We see that the highest occupied state at Γ point is the F-center state. You can follow the calculation using *NaCl_FC.dat* in the directory 'work'. The geometry optimization and molecular dynamics simulations are also supported for the empty atom. So, the position of empty atoms can be optimized variationally.

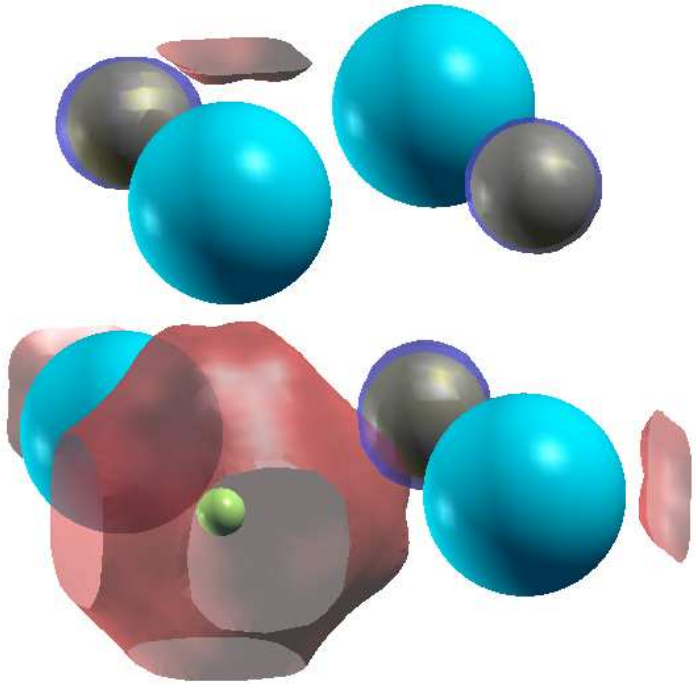


Figure 3: The isosurface map of the highest occupied state at Γ point for NaCl with a Cl-site vacancy, which shows a F-center in NaCl with a Cl vacancy. The isosurface map was drawn using XCrysDen with the isovalue of 0.042 [50]. The calculation was done with the system charge of -1 using a keyword 'scf.system.charge'. The watery and silver colors correspond sodium and chlorine atoms, and the yellow small ball shows the position of empty atom.

9.4 Specification of a directory storing PAO and VPS files

The path to the VPS and PAO directories can be specified in your input file by the following keyword:

```
DATA.PATH    ../DFT_DATA06/    # default=../DFT_DATA/
```

Both the absolute and relative specifications are available. PAO files in the the database (2004) should not be used for the VPS in the database (2006) unless you understand well the content, since semicore states included in several elements are different from each other. So, the consistency in the version of PAO and VPS must be kept. For that reason, it would be better to store PAO and VPS files of each version in different directories. In this case, the keyword is useful.

10 Pseudopotentials

The core Coulomb potential in OpenMX is replaced by a tractable norm-conserving pseudopotential proposed by Troullier and Martine [20]. Although the pseudopotentials can be generated using ADPACK which is a program package for atomic density functional calculations and available from a web site (http://www.openmx-square.org/), for your convenience we offer databases (http://www.openmx-square.org/) of the pseudopotentials. They are database (2004) and database (2006). If you want to use pseudopotentials stored in the database, then copy them to the directory, 'openmx3.5/DFT_DATA/VPS/' or 'openmx3.5/DFT_DATA06/VPS/'. You can freely utilize these data in terms of GNU-GPL, but we cannot offer any warranty on these data. The assignation of pseudopotentials can be made using a keyword 'Definition.of.Atomic.Species' as in the case of specification of basis functions as follows:

```
<Definition.of.Atomic.Species
  H   H4.0-s2p1        H_CA
  C   C4.5-s2p2        C_CA
Definition.of.Atomic.Species>
```

The pseudopotential file can be specified in the third column, and the file must be existing in the directory 'DFT_DATA/VPS' or 'DFT_DATA06/VPS/'. In the specification of atomic coordinates, it is required to give the number of electrons for up- and down-spins at each atom as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.000000    0.000000    0.000000     2.0  2.0
  2   H     -0.889981   -0.629312    0.000000     0.5  0.5
  3   H      0.000000    0.629312   -0.889981     0.5  0.5
  4   H      0.000000    0.629312    0.889981     0.5  0.5
  5   H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
```

where the sixth and seventh columns give the number of up and down initial spin charges for each atom, respectively. The sum of up and down charges for the atomic element should be the number of electrons which is taken into account in the pseudopotential generation. Then, the proper number for each pseudopotential can be found in the pseudopotential file, *.vps. For example, you will see the following line in the file 'C_CA.vps' for carbon atom in the database (2006).

```
 valence.electron            4.0000
```

The number '4.0' corresponds to the number of electrons which is taken into account in the pseudopotential generation. So, we see in above example that the sum of up (2.0) and down (2.0) spins charges is 4.0 for 'C' in the specification of 'Atoms.SpeciesAndCoordinates'.

When you make pseudopotentials using ADPACK by yourself, you should pay attention to the following points.

- Check whether unphysical calculations have been caused by the ghost states or not. Because of the use of the separable form, the ghost states often appear. You should check whether the

pseudopotentials are appropriate or not by performing calculations of simple systems before you calculate systems that you are interested in. To avoid the generation of the ghost states, the multiple separable form proposed by Blochl could be useful [22].

- Make smooth core densities for the partial core correction. If not so, numerical instabilities appear often, since a high energy cutoff is needed for accurate numerical integrations.

You will find the further details in the manual of the program package 'ADPACK'.

11 Cutoff energy

11.1 Convergence

The computational effort and accuracy depend on the cutoff energy, which is controlled by the keyword 'scf.energycutoff', for the numerical integrations and the solution of Poisson's equation [24]. Figure 4 shows the convergence property of the total energy of a methane molecule with respect to the cutoff energy, where the input file is *Methane.dat* used in the Section 'Input file'. Since the cutoff energy is not for basis set, but for the numerical integrations, the total energy does not have to converge from the upper energy region with respect to the cutoff energy like that of plane wave basis set. In most cases, the cutoff energy of 150-200 Ryd is an optimum choice. However, it should be noted that there is a subtle problem which requires the cutoff energy more than 300-500 Ryd. Calculations of a very flat potential minimum and a small energy difference among different spin orders could be such a subtle problem.

Structural parameters and the dipole moment of a water molecule, calculated with a different cutoff energy, are shown in Table 1, where the input file was *H2O.dat* in the directory 'work'. The fully convergent result is obtained using around 150 Ryd. Although a sufficient cutoff energy depends on elements, 150-200 Ryd might be enough to achieve the convergence for most cases. However, we recommend that you would check the convergence of total energy for your systems. For the other cutoff energy, 1DFFT.EnergyCutoff, we commonly use 3600 (Ryd) which is quite enough for the convergence with no high computational demands.

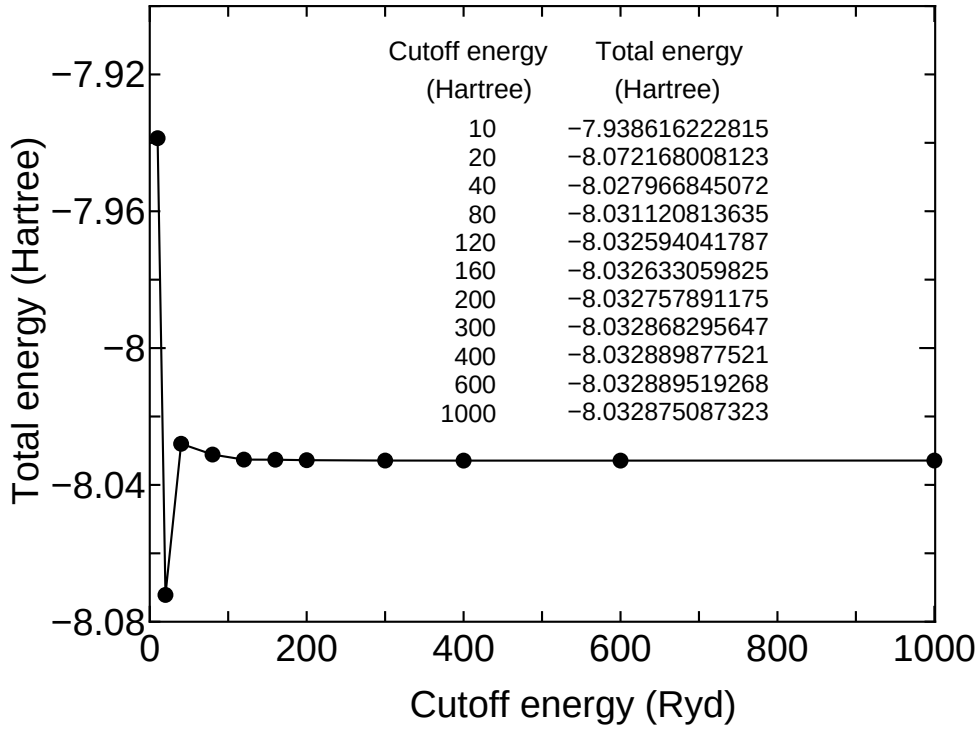


Figure 4: Convergence property of the total energy of a methane molecule with respect to cutoff energy

Table 1: Convergence properties of structural parameters, dipole moment of a water molecule with respect to cutoff energy. The input file is *H2O.dat* in the directory 'work'.

Ecut(Ryd)	r(H-O) (Å)	∠ (H-O-H) (deg)	Dipole moment (Debye)
60	0.971	105.1	1.849
90	0.971	104.7	1.855
120	0.971	104.7	1.856
150	0.971	104.7	1.856
180	0.971	104.7	1.856
Exp.	0.957	104.5	1.85

11.2 A tip for calculating the energy curve for bulks

When the energy curve for bulk system is calculated as a function of the lattice parameter, a sudden change of the number of real space grids is a serious problem which produces an erratic discontinuity on the energy curve. In fact, we see the discontinuity in cases of 200 and 290 (Ryd) in Fig. 5 when the cutoff energy is fixed. The discontinuity occurs at the lattice parameter where the number of grids changes. To avoid the discontinuity on the energy curve, a keyword 'scf.Ngrid' is available.

```
scf.Ngrid      32 32 32     # n1, n2, and n3 for a-, b-, and c-axes
```

When the number of grids is explicitly specified by the keyword, the axis is discretized by the number without depending on the keyword 'scf.energycutoff'. We see in Fig. 5 that the fixed grids with 32x32x32 gives a smooth curve.

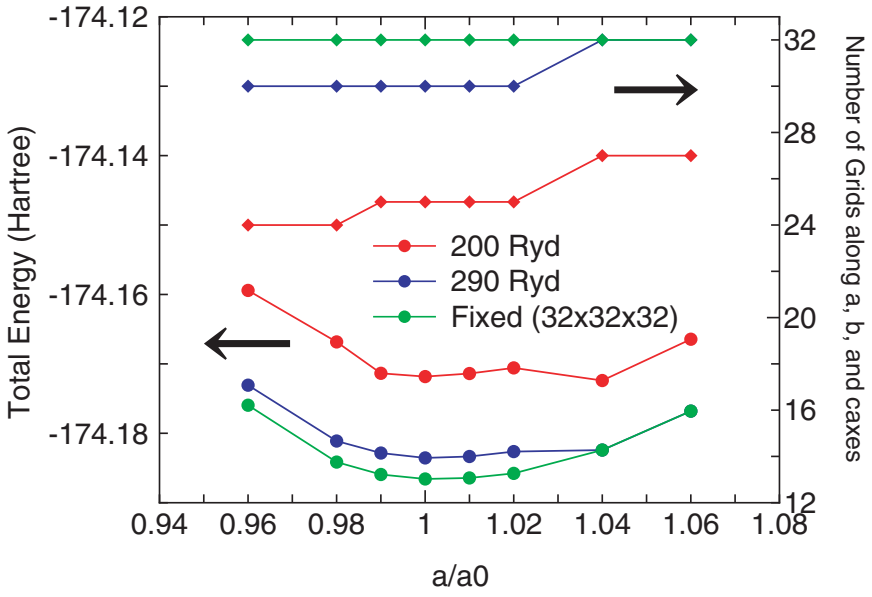


Figure 5: The total energy of bcc iron as a function of lattice parameter, where the equilibrium lattice constant a_0 is 2.87 Å. A cubic unit cell including two atoms was considered. The input file is *Febcc2.dat* in the directory 'work'.

11.3 Fixing the relative position of regular grid

OpenMX Ver. 3.5 uses the regular real space grid for the evaluation of Hamiltonian matrix elements associated with the Hartree potential by the difference charge density and exchange-correlation potential and solution of Poisson's equation. Thus, the total energy depends on the relative position between atomic coordinates and the regular grid. When one calculate interaction energy or energy curve as a function of atomic coordinates, it is quite important to keep the relative position in all the calculations required for the evaluation of the interaction energy. In the calculation for one of the structures, you will find 'Grid_Origin' in the standard output which gives x-, y-, and z-components of the origin of the regular grid as:

```
Grid_Origin  xxx  yyy  zzz
```

Then, in order to keep the relative position, you have to include the following keyword 'scf.fixed.grid' in your input file for all the systems in the calculations required for the evaluation of the interaction energy:

```
scf.fixed.grid  xxx  yyy  zzz
```

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation for one of the structures. The procedure largely suppresses the numerical error involved in the regular grid.

In addition, as discussed in the previous subsection 'A tip for calculating the energy curve for bulks', the number of grids should be fixed by the keyword 'scf.Ngrid' when the lattice parameters are also changed in the evaluation of interaction energy.

12 SCF convergence

Five charge mixing schemes in OpenMX Ver. 3.5 are available by the keyword 'scf.Mixing.Type':

- Simple mixing (Simple)

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight

- Residual minimization method in the direct inversion iterative subspace (RMM-DIIS) [31]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay

- Guaranteed reduction Pulay method (GR-Pulay) [30]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay

- Kerker mixing (Kerker) [32]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Kerker.factor

- RMM-DIIS with Kerker metric (RMM-DIISK) [31]

  Relevant keywords: scf.Init.Mixing.Weight, scf.Min.Mixing.Weight, scf.Max.Mixing.Weight, scf.Mixing.History, scf.Mixing.StartPulay, scf.Mixing.EveryPulay, scf.Kerker.factor

In the first three schemes density matrices (real space) are mixed to generate the input density matrix which can be easily converted into (spin) charge density. On the other hand, the charge mixing is made in Fourier space in the last two schemes. Generally, it is easier to achieve SCF convergence in large gap systems using any mixing scheme. However, it would be difficult to achieve a sufficient SCF convergence in smaller gap and metallic systems, since a charge sloshing problem in the SCF calculations becomes serious often. To handle such difficult systems, two mixing schemes are currently available: Kerker and RMM-DIISK methods. The two mixing schemes could be an effective way for achieving the SCF convergence of metallic systems. When 'Kerker' or 'RMM-DIISK' is used, the following prescriptions are helpful to obtain the convergence of SCF calculations:

- Increase of 'scf.Mixing.History'. A relatively larger vaule 30-50 may lead to the convergence. In addition, 'scf.Mixing.EveryPulay' should be set 1.

- Use a rather larger value for 'scf.Mixing.StartPulay'. Before starting the Pulay type mixing, achieve a convergence at some level. An appropriate value may be 10 to 30 for 'scf.Mixing.StartPulay'.

- Use a rather larger value for 'scf.ElectronicTemperature' in case of metallic systems. When 'scf.ElectronicTemperature' is small, numerical instabilities appear often.

In addition, the charge sloshing, which comes from charge components with long wave length, can be significantly suppressed by tuning Kerker's factor α by the keyword 'scf.Kerker.factor', where Kerker's metric is defined by

$$\langle A|B\rangle = \sum_{\mathbf{q}} \frac{A^*_{\mathbf{q}} B_{\mathbf{q}}}{w_{\mathbf{q}}}$$

$$w_{\mathbf{q}} = \frac{|\mathbf{q}|^2}{|\mathbf{q}|^2 + q_0^2}$$

$$q_0 = \alpha |\mathbf{q}_{\min}|$$

where $\mathbf{q}_{\min}$ is the $\mathbf{q}$ vector with the minimum magnitude except 0-vector. A larger α significantly suppresses the charge sloshing, but leads to slower convergence. Since an optimum value depends on system, you may tune an appropriate value for your system, while the default value is 1.0.

Furthermore, the behavior of 'RMM-DIISK' can be controlled by the following keyword:

```
scf.Mixing.EveryPulay    5   # default = 5
```

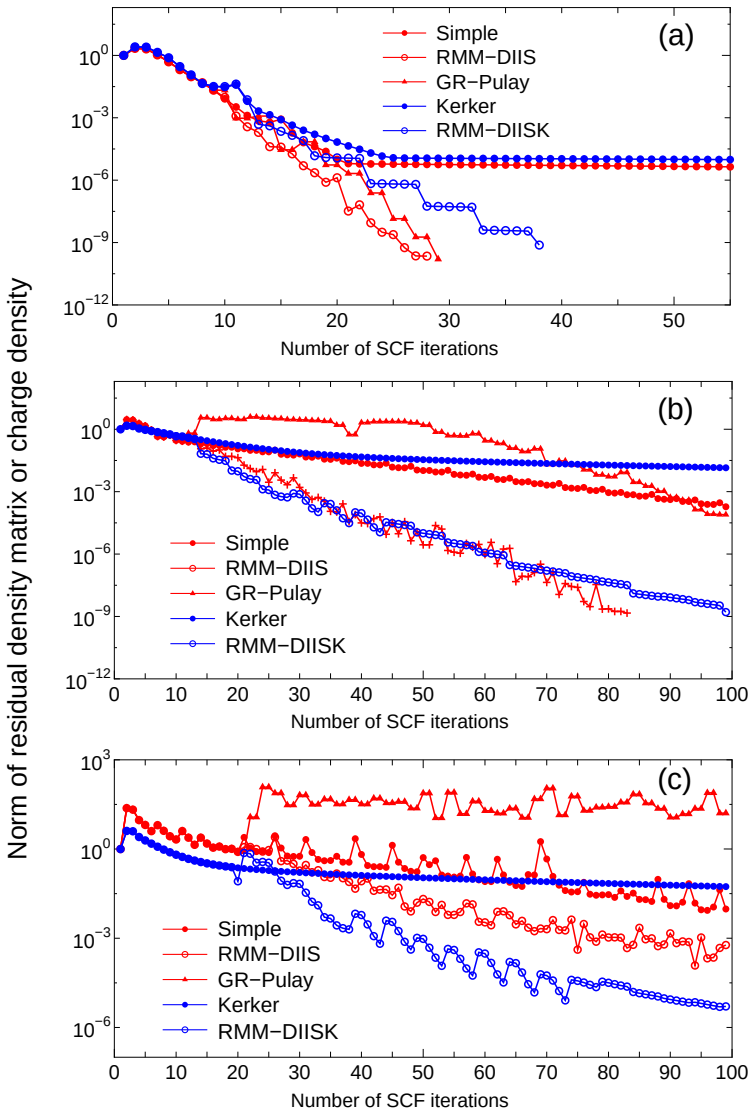


Figure 6: Convergence properties of the norm of residual density matrix or charge density in the SCF calculations using five mixing schemes of (a) a sialic acid molecule, (b) a Pt_{13} cluster, and (c) a Pt_{63} cluster. The input files are *SialicAcid.dat*, *Pt13.dat*, and *Pt63.dat* in the directory 'work'.

The residual vectors in the Pulay-type mixing schemes tend to become linearly dependent each other as the mixing steps accumulate, and the linear dependence among the residual vectors makes the convergence difficult. A way of avoiding the linear dependence is to do the Pulay-type mixing occasionally during the Kerker mixing. With this prescription, you can specify the frequency using the keyword 'scf.Mixing.EveryPulay'. For example, in case of 'scf.Mixing.EveryPulay=5', the Pulay-mixing is made at every five SCF iteration, while Kerker-type mixing is used at the other steps. 'scf.Mixing.EveryPulay=1' corresponds to the conventional Pulay-type mixing. It is noted that the keyword 'scf.Mixing.EveryPulay' is supported for only 'RMM-DIISK', and the default value is five.

The above prescription works in some cases. But the most recommended prescription to accelerate the convergence is the following:

- Increase of 'scf.Mixing.History'. A relatively larger vaule 30-50 may lead to the convergence. In addition, 'scf.Mixing.EveryPulay' should be set 1.

Since the Pulay type mixing such as RMM-DIIS and RMM-DIISK is based on a quasi Newton method, the convergence speed is governed by how a good Hessian matrix can be found. As 'scf.Mixing.History' increases, the calculated Hessian may become more accurate.

In Fig. 6 a comparison of five mixing schemes is shown in the SCF convergence for (a) a sialic acid molecule, (b) a Pt_{13} cluster, and (c) a Pt_{63} cluster, where the norm of residual density matrix or charge density can be found as NormRD in the file *.out and the input files are *SialicAcid.dat*, *Pt13.dat*, and *Pt63.dat* in the directory 'work'. We see that 'RMM-DIISK' works with robustness for all the systems shown in Fig. 6. In most cases, 'RMM-DIISK' will be the best choice, while the use of 'Kerker' is required with a large 'scf.Kerker.factor' and a small 'scf.Max.Mixing.Weight' in quite difficult cases in which the convergence is hardly obtained.

13 Restarting

After finishing your first calculation or achieving the self consistency (SC), you may want to continue the calculation or to calculate density of states, band dispersion, molecular orbitals, and etc. using the SC charge in order to save the computational time. To do this, a keyword 'scf.restart' is available.

```
scf.restart            on        # on|off,default=off
```

When the keyword 'scf.restart' is switched on, restart files generated by your first calculation will be used as the input Hamiltonian or charge density in the second calculation, while 'System.Name' in the second calculation should be the same as in the first calculation. The restart files are stored in a directory '*_rst' below the 'work' directory, where * means System.Name. The restart files in the '*_rst' contain all the information for both the density matrix mixing schemes and k-space mixing schemes. So, it is possible to use another mixing scheme in the second calculation. In the geometry optimization and molecular dynamics simulations, the restart files generated at the previous steps are automatically utilized at the next step to accelerate the convergence using an extrapolation scheme [33, 34]. As an example, we illustrate the restarting procedure using a input file *C60.dat* which can be found in the directory 'work'. In Fig. 7, we see that the second calculation is accelerated due to the use of the restart file.

Input file for the restart calculation

An input file, *.dat#, is generated at every MD step for the restart calculation with the final structure and the same 'Grid_Origin' explained in the Sec. 'Fixing the relative position of regular grid'. Using the file, *.dat#, it can be possible to continue MD calculations and geometry optimization from the last step.

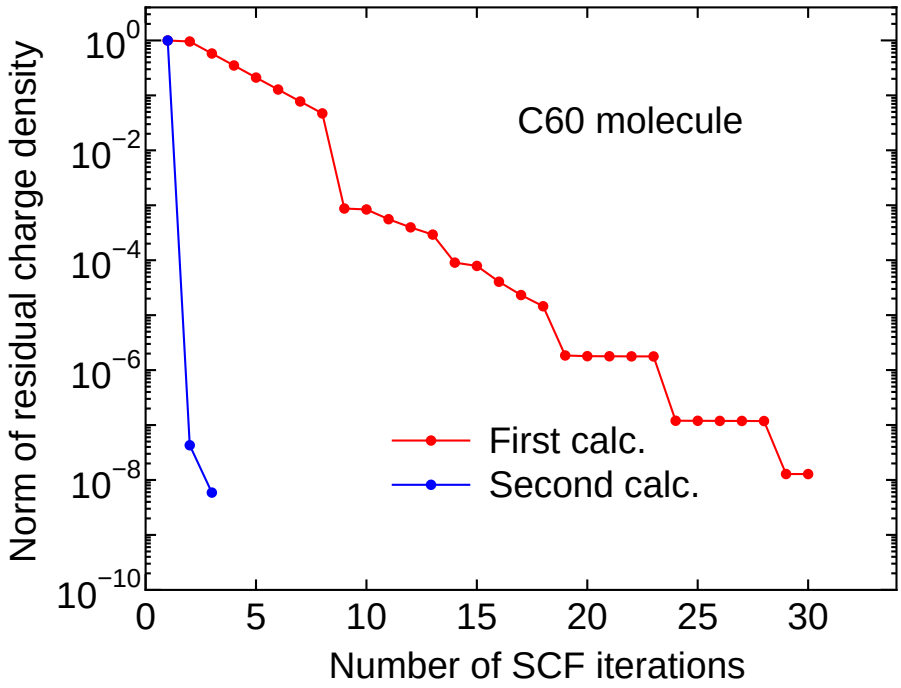


Figure 7: SCF convergence of a C_{60} molecule. In the second calculation, the restart files generated by the first calculation were used. The input file is *C60.dat* in the directory 'work'.

14 Geometry optimization

14.1 Steepest decent optimization

An example of the geometry optimization is illustrated in this Section. As the initial structure, we considered the methane molecule given in the Section 'Input file', but the x-coordinate of the carbon atom of a methane molecule was moved to 0.3 Å as follows:

```
<Atoms.SpeciesAndCoordinates
  1   C      0.300000    0.000000    0.000000     2.0  2.0
  2   H     -0.889981   -0.629312    0.000000     0.5  0.5
  3   H      0.000000    0.629312   -0.889981     0.5  0.5
  4   H      0.000000    0.629312    0.889981     0.5  0.5
  5   H      0.889981   -0.629312    0.000000     0.5  0.5
Atoms.SpeciesAndCoordinates>
```

Then, a keyword 'MD.type' was specified as 'Opt', and set to 200 for a keyword 'MD.maxIter'. The 'Opt' is based on a simple steepest decent method with a variable prefactor. Figure 8 (a) shows the convergence history of the norm of the maximum force on atom as a function of the number of the optimization steps. We see that the norm of the maximum force on atom smoothly converges. Using *Methane2.dat* in the directory 'work', you can trace the calculation.

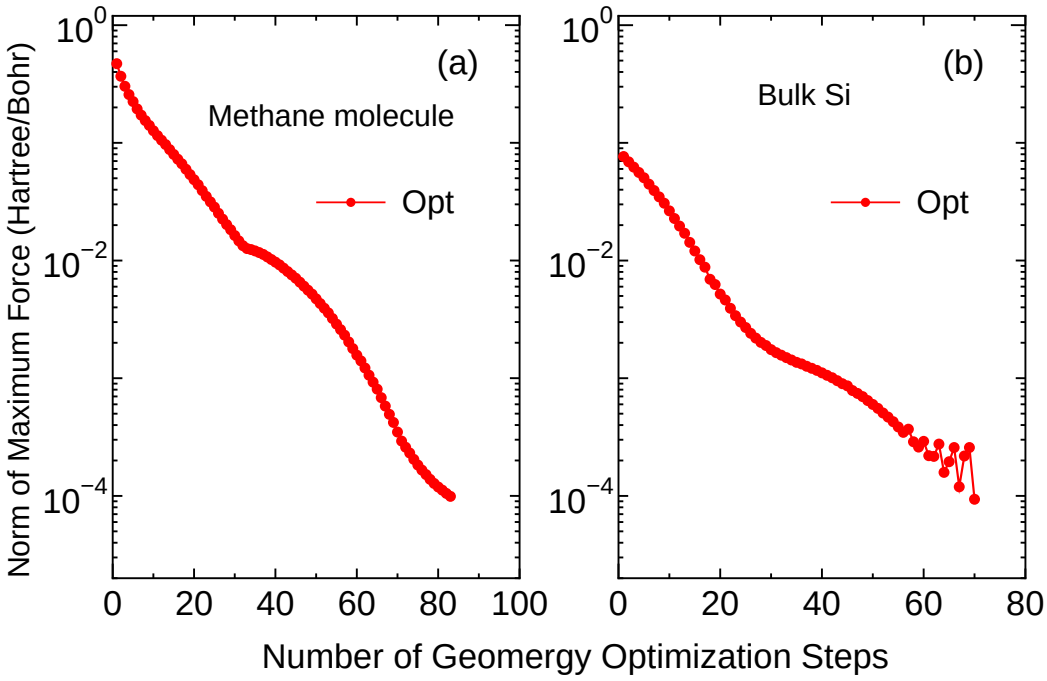


Figure 8: The norm of the maximum force on atom of (a) a methane molecule (b) silicon in the diamond structure as a function of the number of MD steps. The initial structures were ones distorted from the the equilibrium structures. The input files are *Methane2.dat* and *Si8.dat* in the directory 'work', respectively.

14.2 EF, BFGS, RF, and DIIS optimizations

Although 'Opt' is a robust scheme, the convergence speed is very slow in general. Much faster schemes based on a quasi Newton method are available for the geometry optimization. They are the eigenvector following (EF) method [36], the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [38], the rational function (RF) method [37], and a direct inversion iterative sub-space (DIIS) method [35], implemented in the Cartesian coordinate. In the EF and RF methods, the approximate Hessian is updated by the BFGS method. Thus, five geometry optimizers, Opt, EF, BFGS, RF and DIIS, are available in OpenMX Ver. 3.5, which can be specified by 'MD.Type'. The relevant keywords are listed below:

```
MD.Type                    EF        # Opt|DIIS|BFGS|RF|EF
MD.Opt.DIIS.History         3        # default=3
MD.Opt.StartDIIS            5        # default=5
MD.Opt.EveryDIIS          200        # default=200
MD.maxIter                100        # default=1
MD.Opt.criterion        1.0e-4       # default=0.0003 (Hartree/bohr)
```

Then, you can control these schemes by two keywords:

```
MD.Opt.DIIS.History     7       # default=7
MD.Opt.StartDIIS        5       # default=5
```

The keyword 'MD.Opt.DIIS.History' specifies the number of the previous steps to update an optimum Hessian matrix. The default value is 7. Also, the geometry optimization step at which 'EF', 'BFGS', 'RF', or 'DIIS', starts is specified by the keyword 'MD.Opt.StartDIIS'. The geometry optimization steps before starting the these methods is performed by the steepest decent method as in 'Opt'. The default value is 5.

The initial step in the optimization is automatically tuned by monitoring the maximum force in the initial structure, while it was specified by the keyword "MD.Initial.MaxStep" in the version 3.2 (the keyword 'MD.Initial.MaxStep' is not available in OpenMX Ver. 3.5). As shown in the Fig. 9 which shows the number of geometry steps to achieve the maximum force of below 0.0001 hartree/bohr in molecules and bulks, in most cases the EF method seems to be the most robust and efficient scheme.

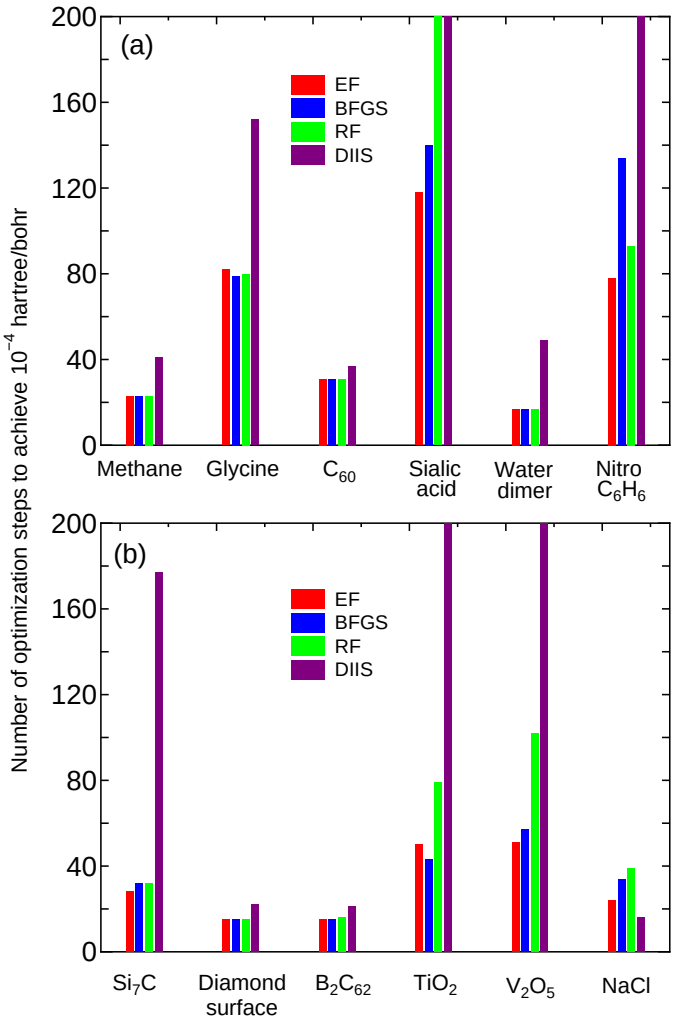


Figure 9: The number of optimization steps to achieve 10^{-4} hartree/bohr for (a) molecular systems and (b) bulk systems using four kinds of optimization methods.

14.3 Constrained relaxation

It is possible to optimize geometrical structures with a constraint in which atoms can be fixed in the initial position. The constraint can be applied separately to the x-, y-, and z-coordinates of the atomic position to the initial position in your input file by the following keyword 'MD.Fixed.XYZ':

```
<MD.Fixed.XYZ
   1  1 1 1
   2  1 0 0
 MD.Fixed.XYZ>
```

The example is for a system consisting of two atoms. If you have N atoms, then you have to provide N-th rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, 4th columns are flags

for the x-, y-, z-coordinates. '1' means that the coordinate is fixed, and '0' relaxed. It should be noted that the definition of the switch is opposite compared to the previous constraint schemes supported in OpenMX3.0. In above example, the x-, y-, z-coordinates of the atom '1' are fixed, only the x-coordinate of the atom '2' is fixed. The default setting is that all the coordinates are relaxed. The fixing of atomic positions are valid for all the geometry optimizers and molecular dynamics schemes. The constrained relaxation may be useful for a refinement of the local structure in large-scale systems.

15 Molecular dynamics

You can perform three molecular dynamics simulations, constant energy molecular dynamics (NVE), constant temperature molecular dynamics by a velocity scaling (NVT_VS), and constant temperature molecular dynamics by the Nose-Hoover method (NVT_NH).

15.1 NVE molecular dynamics

A constant energy molecular dynamics is performed by the following keyword 'MD.Type':

```
MD.Type           NVE       # NOMD|Opt|NVE|NVT_VS|NVT_NH
```

Calculated quantities at every MD step are stored in an output file '*.ene', where * means System.Name. Although you can find the details in 'iterout.c', several quantities are summarized for your convenience as follows:

```
 1:    MD step
 2:    MD time
14:    kinetic energy of nuclear motion, Ukc (Hartree)
15:    DFT total energy, Utot (Hartree)
16:    Utot + Ukc (Hartree)
17:    Fermi energy (Hartree)
```

which means that the first and second columns correspond to MD step and MD time, and so on.

15.2 NVT molecular dynamics by a velocity scaling

A velocity scaling scheme [17] is supported to perform NVT ensemble molecular dynamics by the following keyword:

```
MD.Type           NVT_VS    # NOMD|Opt|NVE|NVT_VS|NVT_NH
```

Then, in this NVT molecular dynamics the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  3
  100   2  1000.0  0.0
  400  10   700.0  0.4
  700  40   500.0  0.7
MD.TempControl>
```

The beginning of the description must be <MD.TempControl, and the last of the description must be MD.TempControl>. The first number '3' gives the number of the following lines to control the temperature. In this case you can see that there are three lines. Following the number '3', in the consecutive lines the first column means the number of MD steps and the second column gives interval of MD steps which determine ranges of MD steps and intervals at which the velocity scaling is made. For above example, a velocity scaling is performed at every two MD steps until 100 MD steps, at every 10 MD steps from 100 to 400 MD steps, and at every 40 MD steps from 400 to 700 MD steps. The third and fourth columns give a given temperature T_{give} (K) and a scaling parameter α in the interval,

while the temperature in the interval is given by a linear interpolation. In this velocity scaling, velocity is scaled by

$$s = \sqrt{\frac{T_{\rm given} + (T_{\rm calc} - T_{\rm given}) * \alpha}{T_{\rm calc}}}$$

$$\mathbf{v}_i' = \mathbf{v}_i \times s$$

where $T_{\rm given}$ and $T_{\rm calc}$ are a given and calculated temperatures, respectively. After the final MD step given in the specification 'MD.TempControl', the NVT ensemble is switched to a NVE ensemble. Calculated quantities at every MD step are stored in an output file '*.ene', where * means System.Name. Although you can find the details in 'iterout.c', several quantities are summarized for your convenience as follows:

```
 1:    MD step
 2:    MD time
14:    kinetic energy of nuclear motion, Ukc (Hartree)
15:    DFT total energy, Utot (Hartree)
16:    Utot + Ukc (Hartree)
17:    Fermi energy (Hartree)
18:    Given temperature for nuclear motion (K)
19:    Calculated temperature for nuclear motion (K)
22:    Nose-Hoover Hamiltonian (Hartree)
```

n which means that the first and second columns correspond to MD step and MD time, and so on. As an example, we show a result for the velocity scaling MD of a glycine molecule in Fig. 10 (a). We see that the temperature in a molecule oscillates around the given temperature. Also for visualization of molecular dynamics an output file '*.md' can be easily animated using free software xmakemol [69].

15.3 NVT molecular dynamics by the Nose-Hoover method

Nose-Hoover molecular dynamics [18] is supported to perform NVT ensemble molecular dynamics by the following keyword:

```
MD.Type          NVT_NH      # NOMD|Opt|NVE|NVT_VS|NVT_NH
```

Then, in this NVT molecular dynamics the temperature for nuclear motion can be controlled by

```
<MD.TempControl
  4
  1    1000.0
  100  1000.0
  400   700.0
  700   600.0
MD.TempControl>
```

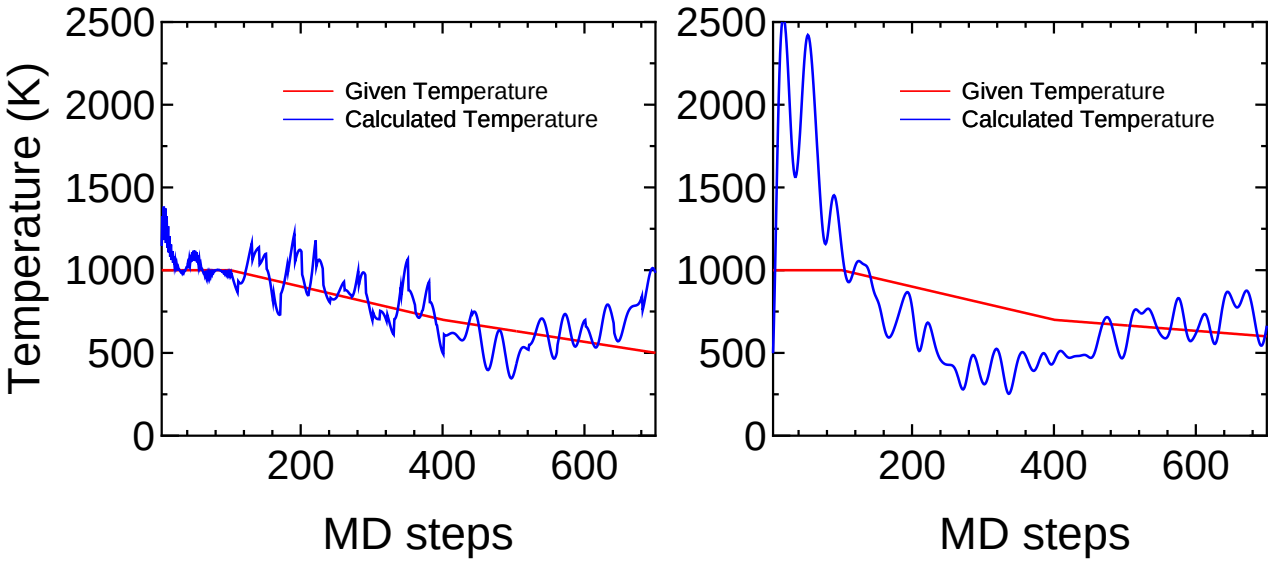


Figure 10: (a) Given and calculated temperatures of a glycine molecule as a function of MD steps in a velocity scaling NVT molecular dynamics. (b) Given and calculated temperatures a glycine molecule as a function of MD steps in the Nose-Hoover NVT molecular dynamics. The input files are *Gly_VS.dat* and *Gly_NH.dat* in the directory 'work', respectively.

The beginning of the description must be <MD.TempControl, and the last of the description must be MD.TempControl>. The first number '4' gives the number of the following lines to control the temperature. In this case you can see that there are four lines. Following the number '4', in the consecutive lines the first and second columns give the number of MD steps and a given temperature for nuclear motion. The temperature between the interval is given by a linear interpolation. Although the same keyword 'MD.TempControl' as used in the velocity scaling MD is utilized in this specification, it is noted that the format is different from each other. In addition to the specification of 'MD.TempControl', you must specify a mass of heat bath by the following keyword:

```
NH.Mass.HeatBath          30.0      # default = 20.0
```

In this specification, we use a unit that the weight of a proton is 1.0. Calculated quantities at every MD step are stored in an output file '*.ene' as explained in 'NVT molecular dynamics by a velocity scaling'. As an example, we show a result for Nose-Hoover MD of a glycine molecule in Fig. 10 (b). We see that the temperature in a molecule oscillates around the given temperature. Also for visualization of molecular dynamics an output file '*.md' can be easily animated using free software xmakemol [69] as well as NVT_VS.

15.4 Constraint molecular dynamics

A constraint scheme is available in the molecular dynamics simulations in which atoms can be fixed in the initial position. The specification is same as in the subsection 'Constrained relaxation'. See the subsection for the specification.

15.5 Initial velocity

For molecular dynamics simulations, it is possible to provide the initial velocity of each atom by the following keyword:

```
<MD.Init.Velocity
 1    3000.000  0.0  0.0
 2   -3000.000  0.0  0.0
MD.Init.Velocity>
```

The example is for a system consisting of two atoms. If you have N atoms, then you have to provide N-th rows in this specification. The 1st column is the same sequential number to specify atom as in the specification of the keyword 'Atoms.SpeciesAndCoordinates'. The 2nd, 3rd, and 4th columns are x-, y-, and z-components of the velocity of each atom. The unit of the velocity is m/s. The keyword 'MD.Init.Velocity' is compatible with the keyword 'MD.Fixed.XYZ'.

16 Visualization

The electron densities, molecular orbitals, and potentials are output to files in a Gaussian cube format. Figure 11 shows examples of isosurface maps visualized by using gOpenMol [48]. These data are output in a form of the Gaussian cube. So, many softwares, such as gOpenMol [48], Molekel [49], and XCrysDen [50], can be used for the visualization. You can find the details of files output in the cube format in the Section 'Output files'. It should be noted that current gOpenMol does not support a cube file of orthorhombic cell, while XCrysDen supports any cube file of both orthogonal and orthorhombic cells.

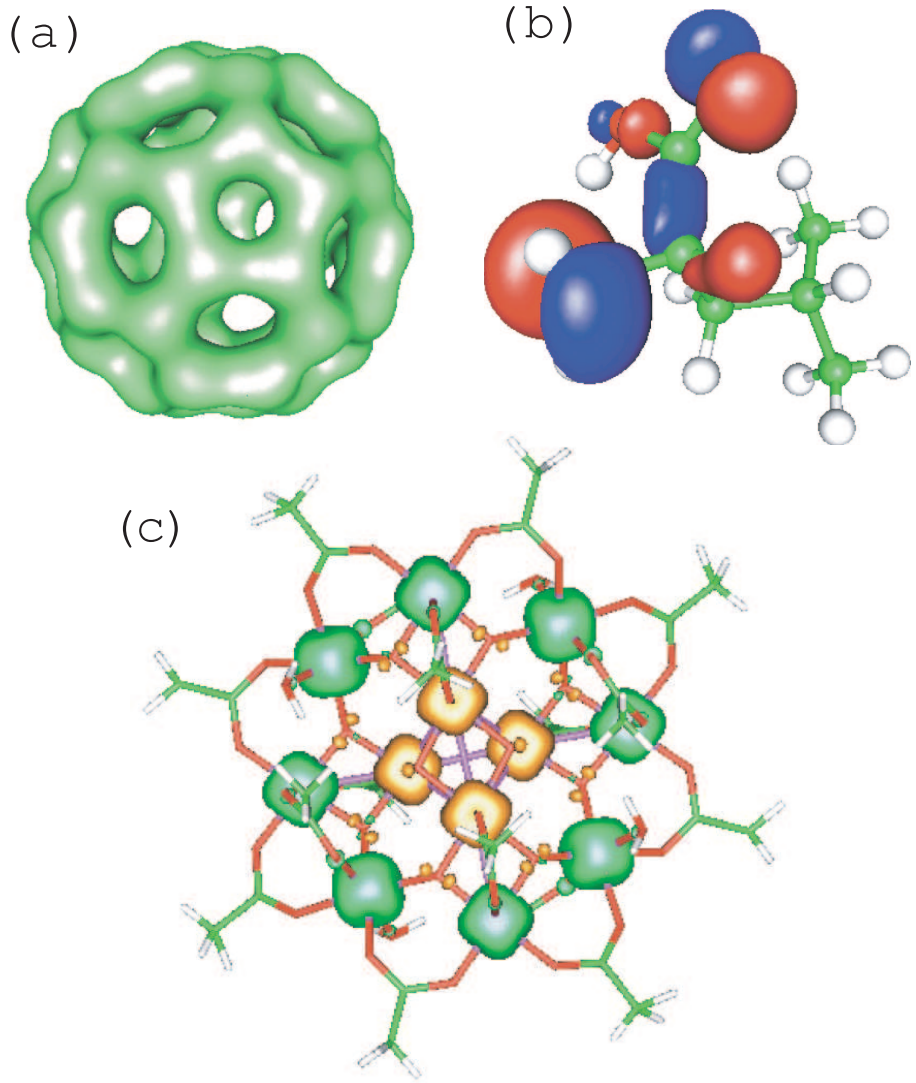


Figure 11: (a) Isosurface map of the total electron density of a C_{60} molecule where 0.13 was used as an isovalue of total electron density. (b) Isosurface map of the highest occupied molecular orbital (HOMO) of a L-leucine molecule where |0.05| was used as an isovalue of the molecular orbital. (b) Isosurface map of the spin electron density of a molecular magnet ($Mn_{12}O_{12}(CH_3COO)_{16}(H_2O)_4$ [51]) where |0.02| was used as an isovalue of the spin electron density.

17 Band dispersion

The band dispersion is calculated by the following two steps:

(1) SCF calculation

Let us illustrate the calculation of band dispersion using the carbon diamond. In a file *Cdia.dat* in the directory 'work', the atomic coordinates, cell vectors, and scf.Kgrid are given by

```
Atoms.Number         2
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
<Atoms.SpeciesAndCoordinates
 1   C  0.000  0.000  0.000   2.0 2.0
 2   C  0.890  0.890  0.890   2.0 2.0
Atoms.SpeciesAndCoordinates>
Atoms.UnitVectors.Unit             Ang # Ang|AU
<Atoms.UnitVectors
  1.7800  1.7800  0.0000
  1.7800  0.0000  1.7800
  0.0000  1.7800  1.7800
Atoms.UnitVectors>

scf.Kgrid                  7 7 7       # means n1 x n2 x n3
```

The unit cell for the band dispersion and k-paths are given by

```
Band.dispersion             on        # on|off, default=off
<Band.KPath.UnitCell
 3.56  0.00  0.00
 0.00  3.56  0.00
 0.00  0.00  3.56
Band.KPath.UnitCell>
Band.Nkpath                  5
<Band.kpath
  15  0.0 0.0 0.0   1.0 0.0 0.0   g X
  15  1.0 0.0 0.0   1.0 0.5 0.0   X W
  15  1.0 0.5 0.0   0.5 0.5 0.5   W L
  15  0.5 0.5 0.5   0.0 0.0 0.0   L g
  15  0.0 0.0 0.0   1.0 1.0 0.0   g X
Band.kpath>
```

Then, we execute OpenMX by:

```
% ./openmx Cdia.dat
```

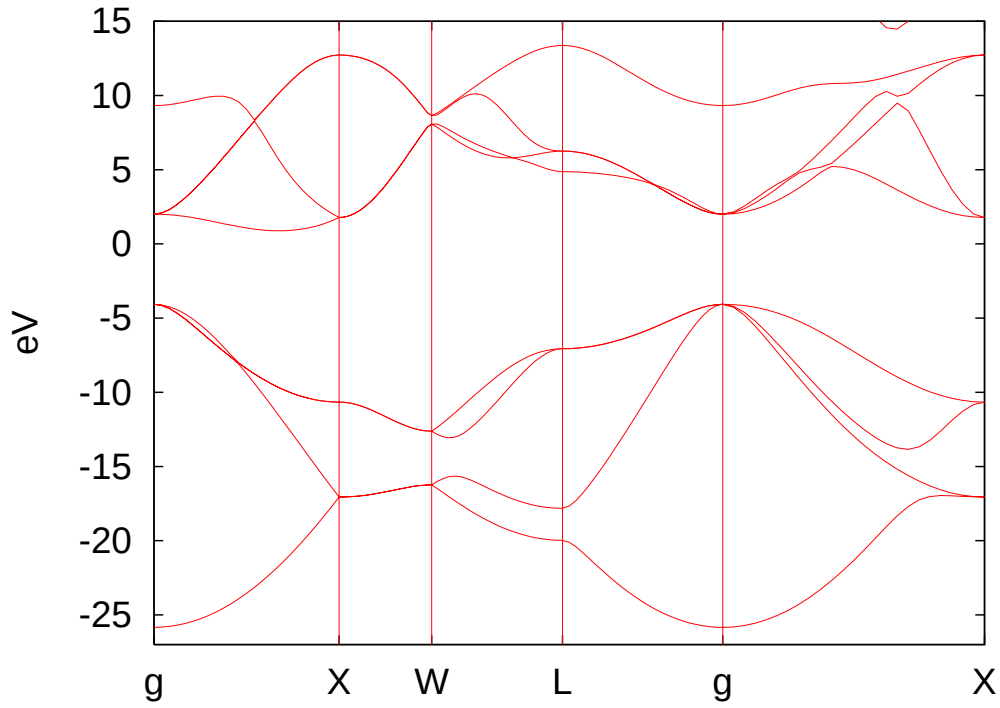


Figure 12: Band dispersion of carbon diamond. The input file is *Cdia.dat* in the directory 'work'.

When the execution is completed normally, then you can find a file 'cdia.Band' in the directory 'work'. When 'Band.KPath.UnitCell' does not exist, the unit cell specified by the 'Atoms.UnitVectors' will be used. Then, it is noted that '0.5 0.0 0.0' corresponds to the 'X'-point

(2) Converting of the data to a gnuplot form

There is a file 'bandgnu13.c' in the directory 'source'. Compile the file as follows:

```
% gcc bandgnu13.c -lm -o bandgnu13
```

When the compile is completed normally, then you can find an executable file, bandgnu13, in the directory 'source'. Please copy the executable file to the directory 'work'. Using the executable file 'bandgnu13' a file 'cdia.Band' is converted in a gnuplot format as

```
% ./bandgnu13 cdia.Band
```

Then, three or two files 'cdia.GNUBAND' and 'cdia.BANDDAT1' ('cdia.BANDDAT2') are generated. The file 'cdia.GNUBAND' is a script for gnuplot, and read data files 'cdia.BANDDAT1' and 'cdia.BANDDAT2' for up- and down-spin, respectively. If spin-polarized calculations using 'LSDA-CA' or 'LSDA-PW' is employed in the SCF calculation, '*.BANDDAT2' for down-spin is generated in addition to '*.BANDDAT1'. The file 'cdia.GNUBAND' is plotted using gnuplot as follows:

```
% gnuplot cdia.GNUBAND
```

Figure 12 shows the band dispersion of carbon diamond, generated by the above procedure, while the range of y-axis was changed in the file cdia.GNUBAND. It is also noted that the chemical potential is automatically shifted to the origin of energy.

A problem in drawing of the band dispersion is how to choose a unit cell used in calculating of the band dispersion. Often, the unit cell used in calculating of the band dispersion is different from that used in the definition of the periodic system. In such a case you need to define the unit cell used in calculating of the band dispersion by the keyword 'Band.KPath.UnitCell'. If you define 'Band.KPath.UnitCell', the reciprocal lattice vectors for the calculation of the band dispersion are calculated by the unit vectors specified in 'Band.KPath.UnitCell'. If you do not define 'Band.KPath.UnitCell', the reciprocal lattice vectors, which are calculated by the unit vectors specified in 'Atoms.UnitVectors' is employed for the calculation of the band dispersion. In case of fcc, bcc, base centered cubic, and trigonal cells, the reciprocal lattice vectors for the calculation of the band dispersion should be specified using the keyword 'Band.KPath.UnitCell' based on the consuetude in the band calculations.

18 Density of states

18.1 Conventional scheme

The density of states (DOS) is calculated by the following two steps:

(1) SCF calculation

Let us illustrate the calculation of the DOS using the carbon diamond. In a file 'Cdia.dat' in the directory 'work', the keywords for the DOS are set to

```
Dos.fileout                on
Dos.Erange              -20.0  20.0
Dos.Kgrid                12 12 12
```

In the specification of the keyword 'Dos.Erange', the first and second values are the lower and upper bounds of the energy range (eV) for the DOS calculation, respectively, where the origin (0.0) of energy corresponds to the chemical potential. Also, in the specification of the keyword 'Dos.Kgrid', a set of numbers (n1,n2,n3) is the number of grids to discretize the first Brillouin zone in the k-space, which is used in the DOS calculation. Then, we execute OpenMX by:

```
% ./openmx Cdia.dat
```

When the execution is completed normally, then you can find files, 'cdia.Dos.val' and 'cdia.Dos.vec' in the directory 'work'. The eigenvalues and eigenvectors are stored in the files 'cdia.Dos.val' and 'cdia.Dos.vec' in a text and binary forms, respectively. The DOS calculation is supported even for the O(N) calculation, while Gaussian broadening methods is employed in this case.

(2) Calculation of the DOS

Let us compile a program package for calculating the DOS. Move the directory, 'source', and then compile as follows:

```
% make DosMain
```

When the compile is completed normally, then you can find an executable file 'DosMain' in the directory 'source'. Please copy the file 'DosMain' to the directory 'work', and then move the directory 'work'. You can calculate the DOS and projected DOS (PDOS) using the program, DosMain, from two files 'cdia.Dos.val' and 'cdia.Dos.vec' as:

```
% ./DosMain cdia.Dos.val cdia.Dos.vec
```

Then, you are interactively asked from the program as follow:

```
% ./DosMain cdia.Dos.val cdia.Dos.vec
Max of Spe_Total_CNO = 8
```

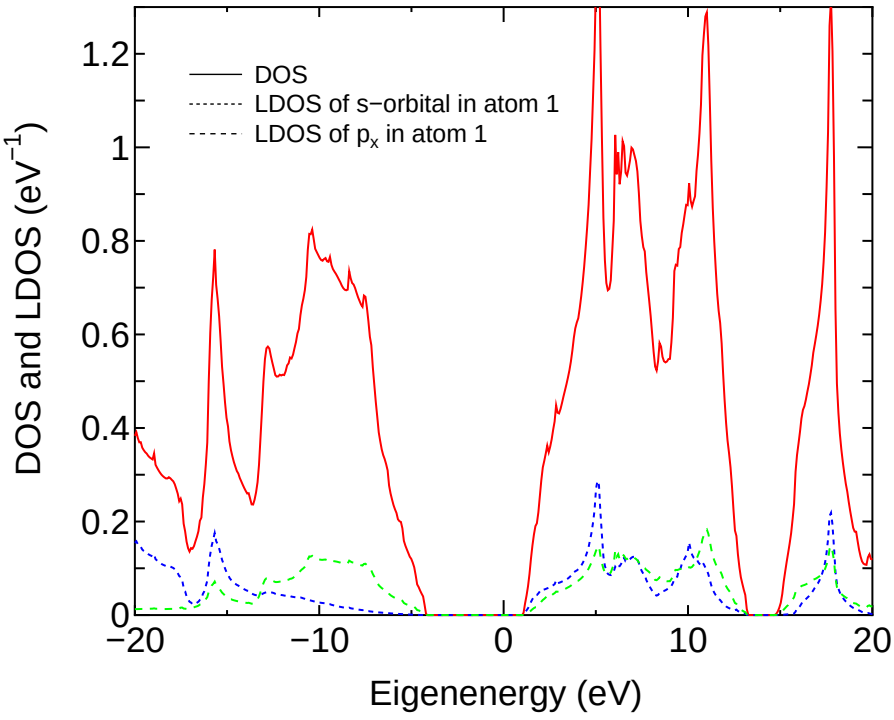


Figure 13: DOS and LDOS of the carbon diamond.

```
1 1 101 102 103 101 102 103
<cdia.Dos.val>
<cdia>
Which method do you use?, Tetrahedron(1), Gaussian Broadeninig(2)
1
Do you want Dos(1) or PDos(2)?
2

Number of atoms=2
Which atoms for PDOS : (1,...,2), ex 1 2
1
pdos_n=1
1
<Spectra_Tetrahedron> start
Spe_Num_Relation 0 0 1
Spe_Num_Relation 0 1 1
Spe_Num_Relation 0 2 101
Spe_Num_Relation 0 3 102
Spe_Num_Relation 0 4 103
Spe_Num_Relation 0 5 101
Spe_Num_Relation 0 6 102
Spe_Num_Relation 0 7 103
```

```
make cdia.PDOS.Tetrahedron.atom1.s1
make cdia.PDOS.Tetrahedron.atom1.p1
make cdia.PDOS.Tetrahedron.atom1.p2
make cdia.PDOS.Tetrahedron.atom1.p3
make cdia.PDOS.Tetrahedron.atom1
```

The tetrahedron [39] and Gaussian broadening methods for evaluating DOS are available. Also, you can select DOS or PDOS. When you select the calculation of PDOS, then please select atoms for evaluating PDOS. In this case, each DOS projected orbitals (s, px (p1), py (p2), pz (p3),..) in selected atoms are output in each files. In these files, the first and second columns are energy in eV and DOS (eV^{-1}) or PDOS (eV^{-1}). If spin-polarized calculations using 'LSDA-CA', 'LSDA-PW', or 'GGA-PBE' is employed in the SCF calculation, the second and third columns in these files correspond to DOS or PDOS for up and down spins. If you select the Gaussian broadening method, you are requested to set a parameter, value of Gaussian, a (eV), which determines the width of Gaussian defined by $\exp(-(E/a)^2)$. Figure 13 shows the DOS and LDOS of carbon diamond.

18.2 For calculations with lots of k-points

Since the calculation of density of states (DOS) of a large-scale system with lots of k-points requires a considerable memory size, the post-processing code 'DosMain' for generating the partial and total DOS tends to suffer from a segmentation fault. For such a case, a Gaussian DOS scheme is available in which the partial DOS is calculated by the Gaussian broadening method in the OpenMX on-the fly calculation and the information of wave functions is not stored in the file '*.Dos.vec'. Since this scheme does not require a large size of memory, it can be used to calculate DOS of large-scale systems. Then, you can specify the following keywords in your input file.

```
DosGauss.fileout      on       # default=off, on|off
DosGauss.Num.Mesh    200       # default=200
DosGauss.Width       0.2       # default=0.2 (eV)
```

When you use the scheme, specify 'on' for the keyword 'DosGauss.fileout'. And the keyword 'DosGauss.Num.Mesh' gives the number of partitioning for the energy range specified by the keyword 'Dos.Erange'. The keyword 'DosGauss.Width' gives the width, a, of the Gaussian $\exp(-(E/a)^2)$. The keyword 'DosGauss.fileout' and the keyword 'Dos.fileout' are mutually exclusive. Therefore, when you use the scheme the keyword, 'Dos.fileout' must be 'off' as follows:

```
Dos.fileout          off       # on|off, default=off
```

Also, the following two keywords are valid for both the keywords 'Dos.fileout' and 'DosGauss.file'.

```
Dos.Erange       -20.0  20.0   # default=-20 20
Dos.Kgrid           5 5 5      # default=Kgrid1 Kgrid2 Kgrid3
```

It should be noted that the keyword 'DosGauss.fileout' generates only the Gaussian broadening DOS, which means that the DOS by the tetrahedron method cannot be calculated by the keyword 'DosGauss.fileout'. After the OpenMX calculations with these keywords, the procedure for DosMain is same as in the conventional scheme.

19 Orbital optimization

The basis orbitals can be variationally optimized using the orbital optimization method [23]. As an illustration of the orbital optimization, let us explain using a methane molecule of which input file is *Methane_OO.dat*. The following keywords in this file are set as follows:

```
<Definition.of.Atomic.Species
 H   H4.0-s41p41        H_TM
 C   C4.5-s41p41        C_TM_PCC
Definition.of.Atomic.Species>

orbitalOpt.Method          species     # Off|Unrestricted|Restricted
orbitalOpt.InitCoes     Symmetrical    # Symmetrical|Free
orbitalOpt.initPrefactor   0.1         # default=0.1
orbitalOpt.scf.maxIter      25         # default=12
orbitalOpt.MD.maxIter       10         # default=5
orbitalOpt.per.MDIter       20         # default=1000000
orbitalOpt.criterion      1.0e-6       # default=1.0e-4 (Hartree/borh)^2
Num.CntOrb.Atoms             2         # default=1
<Atoms.Cont.Orbitals
 1
 2
Atoms.Cont.Orbitals>
```

Then, we execute OpenMX as:

```
% ./openmx Methane.dat
```

When the execution is completed normally, you can find the history of orbital optimization in the file 'met_oo.out' as:

```
*************************************************************
*************************************************************
         History of orbital optimization   MD= 1
*********     Gradient Norm ((Hartree/borh)^2)     ********
             Required criterion=  0.000001000000
*************************************************************

   iter=   1  Gradient Norm=  0.081251614657  Uele= -2.750500719281
   iter=   2  Gradient Norm=  0.018543400953  Uele= -2.933260690003
   iter=   3  Gradient Norm=  0.005918002913  Uele= -2.966113950591
   iter=   4  Gradient Norm=  0.001553729359  Uele= -3.010077558163
   iter=   5  Gradient Norm=  0.000356946294  Uele= -3.012729963043
   iter=   6  Gradient Norm=  0.000119196944  Uele= -3.024577717351
   iter=   7  Gradient Norm=  0.000042934968  Uele= -3.024772396249
```

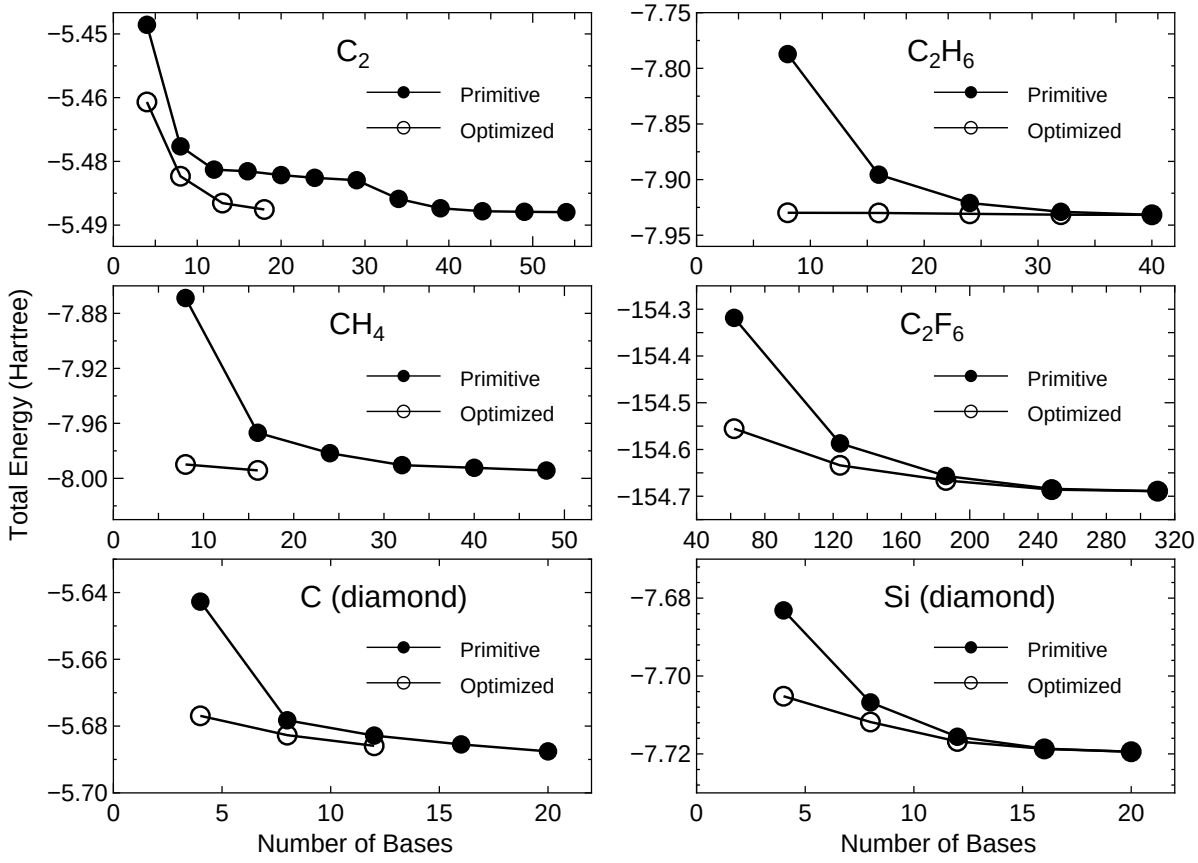


Figure 14: The total energy for a carbon dimer C_2, a methane molecule CH_4, carbon and silicon in the diamond structure, a ethane molecule C_2H_6, and a hexafluoro ethane molecule C_2F_6 as a function of the number of primitive and optimized orbitals. The total energy and the number of orbitals are defined as those per atom for C_2, carbon and silicon in the diamond, and as those per molecule for CH_4, C_2H_6, and C_2F_6.

```
iter=   8  Gradient Norm=  0.000031243105  Uele= -3.026624698820
iter=   9  Gradient Norm=  0.000020515771  Uele= -3.026569330230
iter=  10  Gradient Norm=  0.000015126154  Uele= -3.026833093004
```

In most cases, ten iterative steps are enough to achieve a sufficient convergence. The comparison between the primitive basis orbitals and the optimized orbitals in the total energy is given by

```
Primitive basis orbitals
   Utot  =      -8.032594073571 (Hartree)

Optimized orbitals by the orbital optimization
   Utot  =      -8.150139929748 (Hartree)
```

We see that the small but accurate basis set orbitals can be generated by the orbital optimization. In Fig. 14 we show the convergence properties of total energies for a carbon dimer C_2, a methane

molecule CH_4, and the diamond as a function of the number of unoptimized and optimized orbitals. We see that a remarkable convergent results are obtained using the optimized orbitals for all systems. In this illustration of a methane molecule, the optimized radial orbitals are output to files, C_1.pao and H_2.pao. These output files, C_1.pao and H_2.pao, could be an input data for pseudo-atomic orbitals as it is. This means that it is possible to perform a pre-optimization of basis orbitals for systems you are interested in. The pre-optimization could be performed for smaller but chemically similar systems.

The following three options are available for the keyword 'orbitalOpt.Method', the unrestricted optimization 'Unrestricted', the restricted optimization 'Restricted', and Orbital optimization restricted to species 'Species'.

- Unrestricted

  The radial functions of basis orbitals are optimized without any constraint. Thus, all the radial functions could differ from each other, which could depend on the following indices, atomic number, angular moment quantum number, magnetic quantum number, and orbital multiplicity.

- Restricted

  The radial functions of basis orbitals are optimized with a constraint that the radial wave function R is independent on the magnetic quantum number. We prefer 'Restricted' to 'Unrestricted', since the restricted optimization guarantees the rotational invariance of the total energy.

- Species

  Basis orbitals in atoms with the same species name, that you define in 'Definition.of.Atomic.Species', are optimized as the same orbitals. If you want to assign the same orbitals to atoms with almost the same chemical environment, and optimize these orbitals, this scheme could be quite convenient.

20 Order(N) method

The computational effort of the conventional diagonalization scheme scales as the third power of the number of basis orbitals, which means that the part could be a bottleneck when large-scale systems are calculated. On the other hand, the O(N) methods can solve the eigenvalue problem in O(N) operation in exchange for accuracy. Thus, O(N) methods could be efficient for large-scale systems, while a careful consideration is always required for the accuracy. In OpenMX Ver. 3.5, three O(N) methods are available: a divide-conquer (DC) method [28], a generalized divide-conquer (DC) method [28], and a Krylov subspace method [25]. In the following subsections each O(N) method is illustrated by examples.

20.1 Divide-conquer method

The DC method is a robust scheme and can be applicable to a wide variety of materials with a reasonable degree of accuracy and efficiency, while this scheme is suitable especially for covalent systems. In this subsection, the O(N) calculation using the DC method is illustrated. In an input file 'DIA64_DC.dat' which can be found in the directory 'work', please specify DC for the keyword 'scf.EigenvalueSolver'.

```
scf.EigenvalueSolver    DC
```

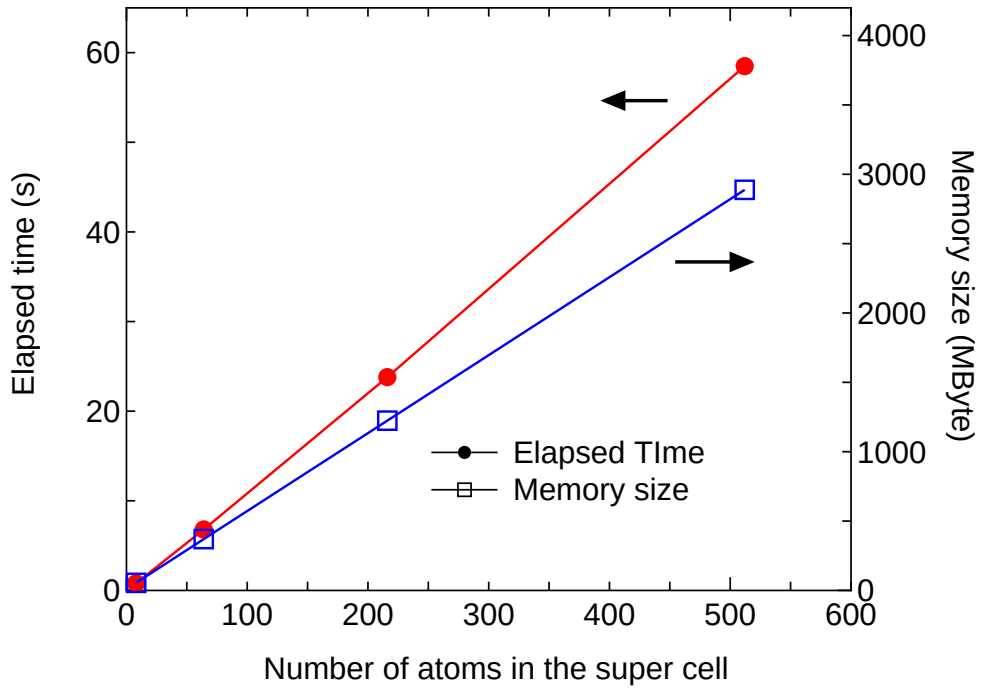


Figure 15: Elapsed time of the diagonalization part per SCF step and computational memory size as a function of carbon atoms in the diamond supercell. C4.0-s1p1 was used as basis orbitals. For the DC method, orderN.HoppingRanges=6.0 (Å) and orderN.NumHoppings=1 are used. An Opteron machine (2.4 GHz) was used to measure the elapsed time. The input files are *DIA8_DC.dat*, *DIA64_DC.dat*, *DIA216_DC.dat*, and *DIA512_DC.dat* in the directory 'work'.

Table 2: Total energy and computational time per MD step of a C_{60} molecule and small peptide molecules (valorphin [52]) and DNA consisting of cytosines and guanines calculated by the conventional diagonalization and the O(N) DC method, where a minimal basis set was used. In this Table, numbers in the parenthesis after DC means orderN.HoppingRanges and orderN.NumHoppings used in the DC calculation. The computational times were measured using an Opteron PC cluster (16 cpus × 2.4 GHz). The input files are *C60_DC.dat*, *Valorphin_DC.dat*, *CG15c_DC.dat* in the directory 'work'.

	Total energy (Hartree)	Computational time (s)
C_{60}		
(60 atoms, 240 orbitals)		
Conventional	-332.25510	21
DC (7.0, 2)	-332.26218	32
Valorphin		
(125 atoms, 317 orbitals)		
Conventional	-559.20738	68
DC (6.5, 2)	-559.20782	88
DNA		
(650 atoms, 1980 orbitals)		
Conventional	-4130.93861	1265
DC (6.3, 2)	-4130.93645	1213

Then, one can execute OpenMX by:

```
% ./openmx DIA64_DC.dat
```

This input file is for an O(N) calculation (1 MD step) of the diamond including 64 carbon atoms. The computational time is 397 seconds using a Xeon machine (2.8 GHz). Figure 15 shows the computational time and memory size to calculate a MD step of the carbon diamond as a function of number of atoms in the supercell. In fact, we see that the computational time and memory size are almost proportional to the number of atoms.

The accuracy and efficiency of the DC method are controlled by two simple parameters: 'orderN.HoppingRanges' and 'orderN.NumHoppings'.

- orderN.HoppingRanges

  The keyword 'orderN.HoppingRanges' defines the radius of a sphere which is centered on each atom. The logically truncated cluster for each atom is constructed for the atoms inside the sphere.

- orderN.NumHoppings

  The keyword 'orderN.NumHoppings' gives the number, n, of hopping which is required to construct the logically truncated cluster. The cluster of size, n, is defined by all neighbors that can be reached by n hops, where the cutoff distance is given by the sum of the cutoff distances r_1 and r_2 of basis orbitals belonging to atoms 1 and 2.

If the number of atoms in the systems is N, N small eigenvalue problems for the N logically truncated clusters are solved, and then the total density of states (DOS) is constructed as the sum of the projected DOS of each logically truncated cluster. Although the appropriate values for 'orderN.HoppingRanges' and 'orderN.NumHoppings' depend on interested systems, for molecular systems the following values are recommended as a trade-off between the computational accuracy and efficiency:

```
orderN.HoppingRanges      6.0 - 7.0
orderN.NumHoppings            2
```

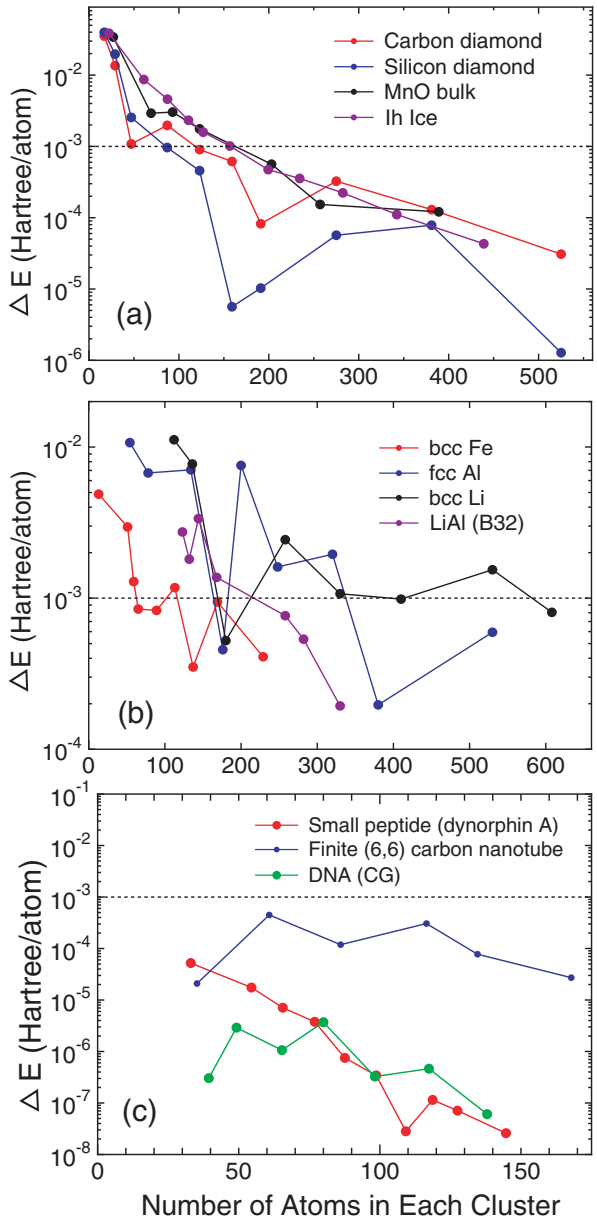


Figure 16: Error in the total energy of (a) bulks with a finite gap, (b) metals, and (c) molecular systems calculated by the divide-conquer (DC) method as a function of the number of atoms in each cluster. The dotted horizontal line indicates 'milli-Hartree' accuracy.

Table 2 shows the comparison in the total energy between the exact diagonalization and the DC method for a C_{60} molecule and small peptide molecules (valorphin [52]), and DNA consisting of cytosines and guanines. We find that errors in the total energy calculated by the DC method are about a few mHartree in this system size. Also, it can be estimated that the DC method is faster than the conventional diagonalization when the number of atoms is larger than 500 atoms, while the crossing point between the conventional diagonalization and the DC method with respect to computational time depends on systems and the number of processors in parallel implementation.

To see an overall tendency in the convergence properties of total energy with respect to the size of truncated cluster, the error in the total energy, compared to the exact diagonalization, is shown as a function of the number of atoms in each cluster for (a) bulks with a finite gap, (b) metals, and (c) molecular systems in Fig. 16. We see that the error decreases almost exponentially for the bulks with a finite gap and molecular systems, while the convergence speed is slower for metals.

20.2 Generalized divide-conquer method

A generalized divide-conquer (GDC) method, in which a cluster is taken into account as the core region instead of a single atom in the DC method, is available by the following option:

```
scf.EigenvalueSolver       GDC
```

The core region is automatically determined so that the computational efficiency can be maximized and this scheme can be more efficient than the conventional DC method for low dimensional systems, while the details are not shown here. The total number of truncated clusters is reduced as a result of the clustered core region which is is the reason for the efficiency. As well as the DC method, the accuracy and efficiency are controlled by the following keywords: 'orderN.HoppingRanges' and 'orderN.NumHoppings'.

20.3 Krylov subspace method

The DC and GDC methods are robust and accurate for a wide variety of systems. However, to obtain an accurate result the size of truncated clusters tends to be large for metallic systems as shown in Fig. 16. A way of reducing the computational efforts is to map the original vector space defined by the truncated cluster into a Krylov subspace of which dimension is smaller than that of the original space [25]. The Krylov subspace method is available by

```
scf.EigenvalueSolver       Krylov
```

Basically, the accuracy and efficiency are controlled by the following three keywords:

```
orderN.HoppingRanges         6.0
orderN.NumHoppings            2
orderN.KrylovH.order        400
```

The keywords 'orderN.HoppingRanges' and 'orderN.NumHoppings' define the radius of a sphere centered on each atom and the number of hopping in the same sense as those in the DC and GDC methods. The dimension of Krylov subspace of Hamiltonian in each truncated cluster is given by the 'orderN.KrylovH.order'. Moreover, the Krylov subspace method can be precisely tuned by the following keywords:

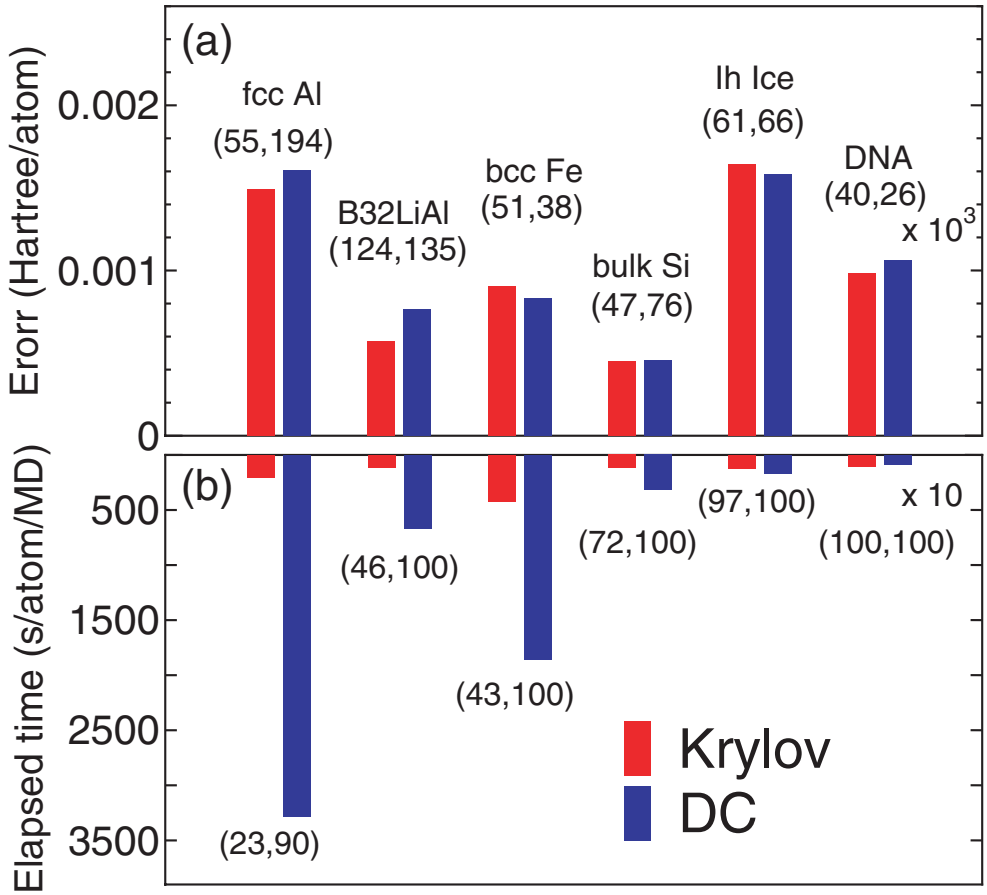


Figure 17: (a) absolute error, with respect to the band calculations, in the total energy (Hartree/atom) calculated by the proposed and DC methods for metals and finite gap systems, (b) computational time (s/atom/MD). For a substantial comparison, the calculations were performed using a single Xeon processor. The set of numbers in the parenthesis of (a) means the average number of atoms in the core and buffer regions. The set of numbers in the parenthesis of (b) means the percentage of the dimension of the subspaces relative to the total number of basis functions in the truncated cluster, respectively.

- orderN.Exact.Inverse.S on| off, default=on

  In case of 'orderN.Exact.Inverse.S=on', the inverse of overlap matrix for each truncated cluster is exactly evaluated. Otherwise, see the next keyword 'orderN.KrylovS.order'.

- orderN.KrylovS.order 1200, default=orderN.KrylovH.order×4

  In case of 'orderN.Exact.Inverse.S=off', the inverse is approximated by a Krylov subspace method for the inverse, where the dimension of the Krylov subspace of overlap matrix in each truncated cluster is given by the keyword 'orderN.KrylovS.order'.

- orderN.Recalc.Buffer on| off, default=off

  In case of 'orderN.Recalc.Buffer=on', the buffer matrix is recalculated at every SCF step. Otherwise, the buffer matrix is calculated at the first SCF step, and fixed at subsequent SCF steps.

- orderN.Expand.Core on| off, default=on

  In case of 'orderN.Expand.Core=on', the core region is defined by atoms within a sphere with radius of $1.2 \times r_{\min}$, where $r_{\min}$ is the distance between the central atom and the nearest atom. In case of 'orderN.Expand.Core=off', the central atom is considered as the core region.

It is better to switch on 'orderN.Exact.Inverse.S' and 'orderN.Expand.Core' as the covalency increases, while the opposite could becomes better in simple metallic systems. In Fig. 17 the absolute error in the total energy calculated by the Krylov and DC methods are shown for a wide variety of materials. It is found that in comparison with the DC method, the Krylov subspace method is more efficient especially for metallic systems, and that the efficiency become comparable as the covalency and ionicity in the electronic structure increase.

21 MPI parallelization

For large scale calculations, parallel execution by MPI is supported for parallel machines with distributed memories.

21.1 O(N) calculation

When the O(N) method is employed, it is expected that one can obtain a good parallel efficiency because of the algorithm. A typical MPI execution is as follows:

```
% mpirun -np 4 openmx DIA512.dat > dia512.std &
```

The input file *DIA512_DC.dat* found in the directory 'work' is for the SCF calculation (1 MD) of the diamond including 512 carbon atoms using the divide-conquer (DC) method. The speed-up ratio in comparison of the elapsed time per MD step is shown in Fig. 18 (a) as a function of the number of processors on a Cray XT3 (2.4 GHz/Optetron processor). We see that the parallel efficiency decreases as the number of processors increase, and the speed-up ratio at 128 CPUs is about 50. The decreasing efficiency is due to the decrease of the number of atoms allocated to one processor. So, the weight of other unparallelized parts such as disk I/O becomes significant. Moreover, it should be noted that the efficiency is significantly reduced in non-uniform systems in terms of atomic species and geometrical structure due to disruption of the road balance, while an algorithm is implemented to avoid the disruption.

21.2 Cluster calculation

In the cluster calculation, a double parallelization is made for two loops: spin multiplicity and eigenstates, where the spin multiplicity means one, two, and one for spin-unpolarized, spin-polarized, and non-collinear calculations, respectively. The priority of parallelization is in order of spin multiplicity and eigenstates. In the eigenvalue solver, the Householder transformation, which tridiagonalizes a Hermitian matrix, the back transformation, and other matrix operations are parallelized. Only eigenvalues and eigenvectors of the tridiagonalized matrix are evaluated using lapack routines, which is a minority part in the computational time of the diagonalization if only eigenvectors of occupied and lower exited states are evaluated. To avoid the calculation of eigenstates in the high energy region, it is highly recommended to use 'dstevx' which is specified by the following keyword:

```
scf.lapack.dste    dstevx    # dstegr|dstedc|dstevx, default=dstevx
```

Since 'dstevx' is default, if you like 'dstevx', you do not need to specify the keyword. In case of 'dstevx', the eigenstates to be calculated is automatically determined by the number of electrons. In the other schemes 'dstegr' and 'dstedc', eigenstates in the higher energy region are also calculated. Figure 18 (b) shows the speed-up ratio as a function of processors in the elapsed time for a spin-polarized calculation of a single molecular magnet consisting of 148 atoms. The input file *Mn12.dat* is found in the directory 'work'. It is found that the speed-up ratio is 19 and 27 using 32 and 64 processors, respectively.

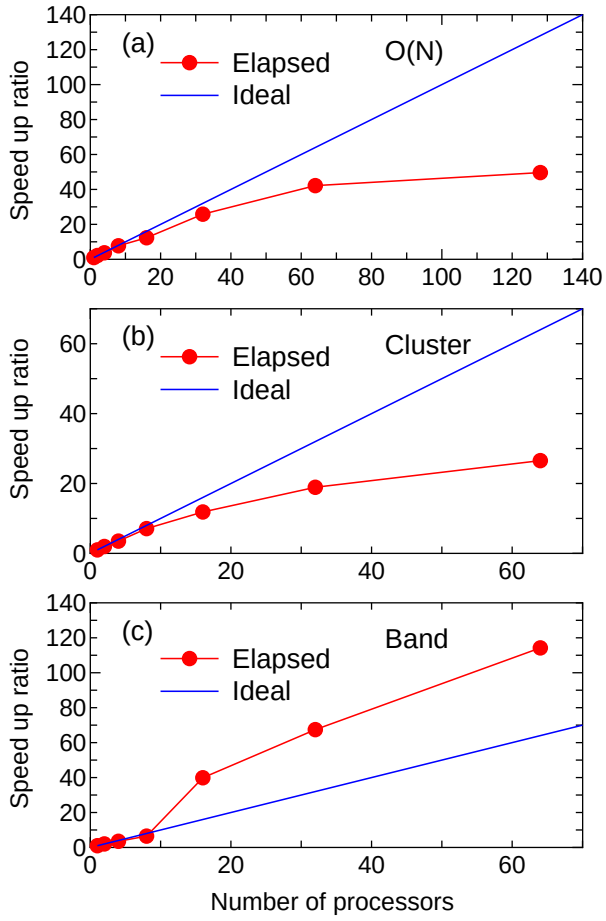


Figure 18: Speed-up ratio of the elapsed time per MD step in parallel calculations using MPI on a Cray XT3 (2.4 GHz Opterons connected with 7.68GB/s networks) (a) for the carbon diamond including 512 atoms in the super cell by the DC method, (b) for a single molecular magnet consisting of 148 atoms by the cluster method, and (c) for the carbon diamond including 64 atoms in the super cell by the band method with 3×3×3 k-points. For comparison, a line which corresponds to the ideal speed-up ratio is also shown.

21.3 Band calculation

In the band calculation, a triple parallelization is made for three loops: spin multiplicity, k-points, and eigenstates, where the spin multiplicity means one, two, and one for spin-unpolarized, spin-polarized, and non-collinear calculations, respectively. The priority of parallelization is in order of spin multiplicity, k-points, and eigenstates. In addition, when the number of processors used in the parallelization exceeds (spin multiplicity)×(the number of k-points), OpenMX uses an efficient way in which finding the Fermi level and calculating the density matrix are performed by just one diagonalization at each k-point. For the other cases, twice diagonalizations are performed at each k-point for saving the size of used memory in which the second diagonalization is performed to calculate the density matrix after finding the Fermi level. In Fig. 18 (c) we see a good speed-up ratio as a function of processors in the elapsed time for a spin-unpolarized calculation of carbon diamond consisting of 64 carbon atoms with 3×3×3 k-points. The input file *DIA64_Band.dat* is found in the

directory 'work'. In this case the spin multiplicity is one, and the number of k-points used for the actual calculation is (3*3*3-1)/2+1=14 since the k-points in the half Brillouin zone is taken into account for the collinear calculation, and the Γ-point is included when all the numbers of k-points for a-, b-, and c-axes are odd. So it is found that the speed-up ratio exceeds the ideal one in the range of processors over 14, which means the algorithm in the parallelization is changed to the efficient scheme. As well as the cluster calculation, to avoid the calculation of eigenstates in the higher energy region, it is highly recommended to use 'dstevx' which is specified by the following keyword 'scf.lapack.dste':

```
scf.lapack.dste    dstevx    # dstegr|dstedc|dstevx, default=dstevx
```

Since 'dstevx' is default, if you like 'dstevx', you do not need to specify the keyword. In case of 'dstevx', the eigenstates to be calculated is automatically evaluated by the number of electrons. In case of 'dstevx', the eigenstates to be calculated is automatically determined by the number of electrons. In the other schemes 'dstegr' and 'dstedc', eigenstates in the higher energy region are also calculated.

21.4 a-axis should be the longest axis

Our parallel execution is made by a simple one-dimensional domain decomposition for a-axis of the unit cell, while other parameters are also used for the parallelization in each subroutine case by case. Therefore, it is better to choose the a-axis as the longest axis for a good road balancing, although, of course, the parallel execution is made for any unit cell.

21.5 Maximum number of processors

In OpenMX Ver. 3.5, for all the calculations of O(N), cluster, and band calculations, the number of processors that you can use for the parallel calculations is limited up to the number of atoms in your system.

22 OpenMP/MPI hybrid parallelization

The OpenMP/MPI hybrid parallel execution can be performed by

```
% mpirun -np 32 openmx DIA512-1.dat -nt 4 > dia512-1.std &
```

where '-nt' means the number of threads in each process managed by MPI. If '-nt' is not specified, then the number of threads is set 1, which corresponds to the pure MPI parallelization.

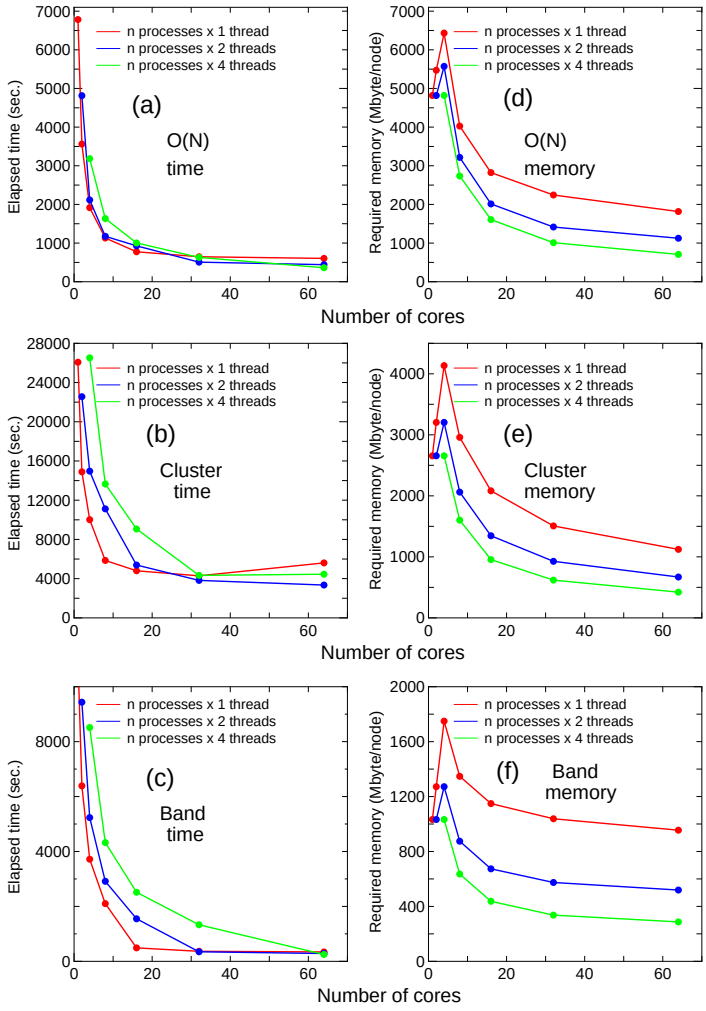


Figure 19: The elapsed time (sec.) and the required memory size (Mbyte) per node in calculations for (a) and (d) the O(N) Krylov subspace, (b) and (e) the cluster, and (c) and (f) the band methods, respectively, where the number of cores is given by the number of processes by MPI times the number of threads by OpenMP. The machine used is an Opteron cluster consisting of two dual core AMD Opteron (tm) processors 2218, 2.6 GHz, with 8 Gbyte memory per node. Those nodes are connected with Gbit ether network. The input files used for those calculations are *DIA512-1.dat*, *Mn12.dat*, and *DIA64_Band.dat* for the O(N) Krylov subspace, the cluster, and the band methods, respectively. They can be found in the directory 'work'.

Figure 19 shows the elapsed time (sec.) and the required memory size (Mbyte) per node in calculations for the O(N) Krylov subspace, the cluster, and the band methods, respectively, where the number of cores is given by the number of processes by MPI times the number of threads by OpenMP. As you can see, the hybrid parallelization using 2 or 4 threads is not fast in the region using the smaller number of processes. However, the hybrid parallelization gives us the shortest elapsed time eventually as the number of processes increases. This behavior may be understood as follows: in the region using the smaller number of processes the required memory size is large enough so that cash miss easily happens. This may lead to considerable communication between processor and memory via bus. So, in the region using the smaller number of processes, the bus becomes a bottle neck in terms of elapsed time. On the other hand, in the region using the large number of processes, the required memory size is small enough that most of data can be stored in the cashes. So, the efficiency in OpenMP parallelization can be recovered. In this case, the hybrid parallelization can obtain both the benefits of MPI and OpenMP. Thus, the hybrid parallelization should be eventually efficient as the number of processes increases. In fact, our benchmark calculation may be the case. Also, it should be emphasized that the required memory size per node can be largely reduced in the hybrid parallelization in OpenMX as shown in the Fig. 19.

23 Large-scale calculation

A simple way of performing large-scale calculations is firstly to employ an O(N) method to obtain a self-consistent charge density, and then is to just once diagonalize using the conventional diagonalization method under the self-consistent charge density to obtain full wave functions. As an illustration of this procedure, we show a large-scale calculation of a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms. First, the SCF calculation of a MCCN was performed using the O(N) DC method and 32 processors of 2.4 GHz Opteron, where C4.5-s2p1 (basis function), 100 Ryd (scf.energycutoff), 1.0e-7 (scf.criterion), 6.5 Å (orderN.HoppingRanges), 2 (orderN.NumHoppings) and RMM-DIISK (mixing scheme) were used. The input file is *MCCN.dat* in the directory 'work'. Figure 20 shows the norm of residual charge density in Fourier space as a function of SCF steps. We see that 68 SCF steps is enough to obtain a convergent charge density for this system, where the computational time was 24 minutes.

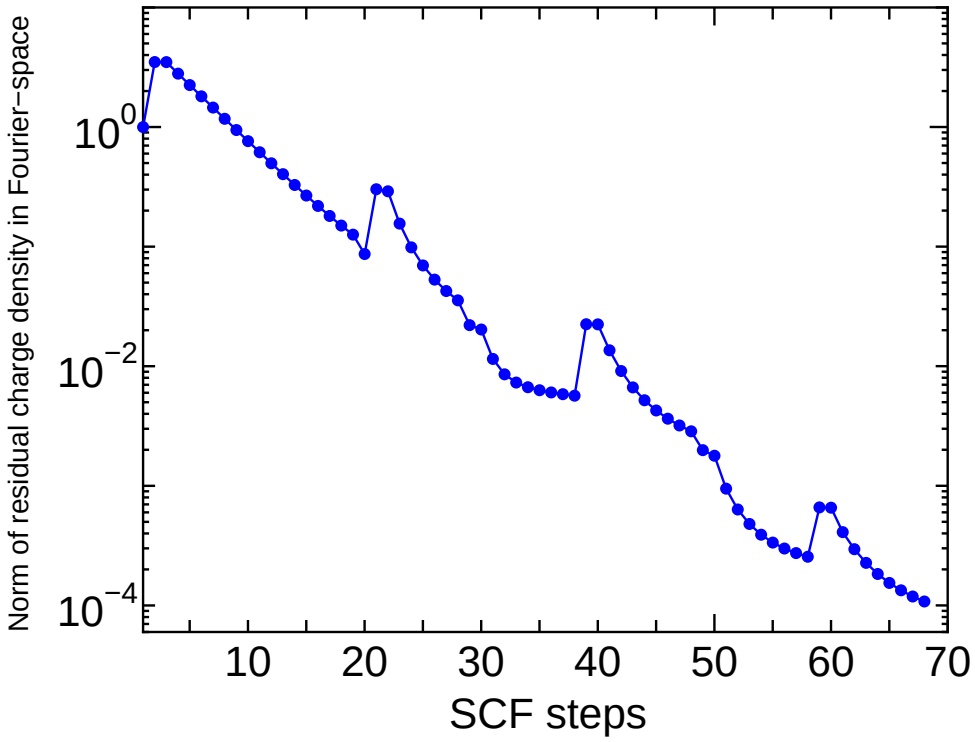


Figure 20: Norm of residual charge density in Fourier space as a function of SCF steps for a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms. The input file is *MCCN.dat* in the directory 'work'.

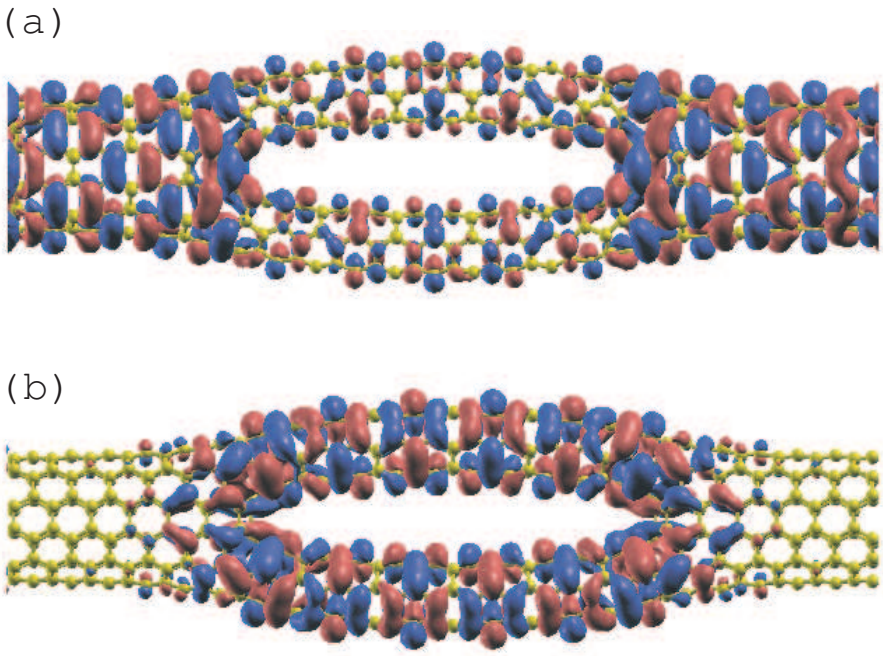


Figure 21: Isosurface map of (a) the highest occupied molecular orbital (HOMO) and (b) the lowest unoccupied molecular orbital (LUMO) of a multiply connected carbon nanotube (MCCN) consisting of 564 carbon atoms, where |0.005| was used as an isovalue of the molecular orbital.

Then, the following keywords were set in

```
scf.maxIter                 1
scf.EigenvalueSolver     Band
scf.Kgrid               1 1 1
scf.restart                on
MO.fileout                 on
num.HOMOs                   2
num.LUMOs                   2
MO.Nkpoint                  1
<MO.kpoint
  0.0  0.0  0.0
MO.kpoint>
```

And we calculated the same system in order to obtain wave functions using 32 processors of 2.4 GHz Opteron, where the computational time was 24 minutes. Figure 20 shows isosurface maps of the HOMO and LUMO (Γ-point) of MCCN calculated by the above procedure. Although the difference between the O(N) method and the conventional diagonalization scheme in the computational time is not significant in this example, the procedure will be useful for larger system including more than a thousand atoms.

24 Electric field

It is possible to apply a uniform external electric field given by a sawtooth waveform during the SCF calculation and the geometry optimization. For example, an electric field of 1.0 GV/m (10^9 V/m) is applied along the a-axis, in your input file specify the keyword 'scf.Electric.Field' as follows:

```
scf.Electric.Field   1.0 0.0 0.0   # default=0.0 0.0 0.0 (GV/m)
```

The sign of electric field is taken as that applied to electrons. If the uniform external electric field is applied to a periodic bulk system without vacuum region, discontinuities of the potential are introduced, which could cause numerical instabilities. On the other hand, for molecular systems, the discontinuities are located in the vacuum region, indicating that numerical instabilities may not be induced.

As an illustration of the electric field, changes of total charge in a nitrobenzene molecule induced by the electric field are shown in Fig. 22. We can see that a large charge transfer takes place among oxygens in -NO_2, para-carbon atom, and para-hydrogen atom. The input file is *Nitro_Benzene.dat* in the directory 'work'. See also Section 'Analysis of difference in two Gaussian cube files' as for the difference charge maps shown in Fig. 22.

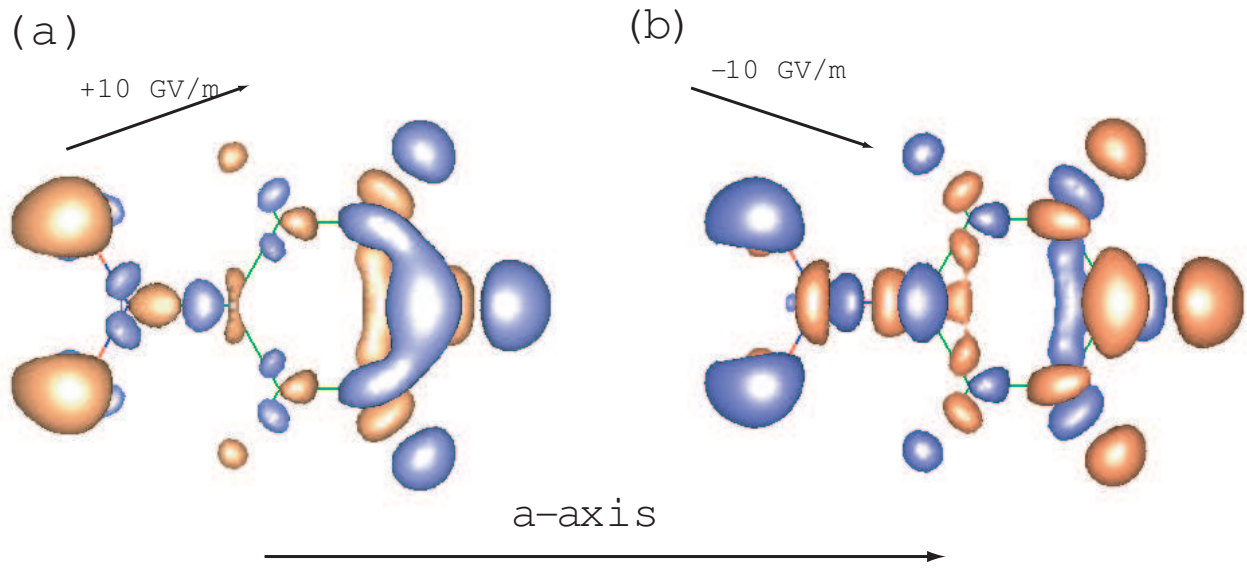


Figure 22: Difference in the total charge density of a nitrobenzene molecule between the zero-bias voltage and (a) 10 GV/m, and (b) -10 GV/m of applied bias along the a-axis, where orange and blue colors mean the increase and decrease of charge density. Tilted arrows depict the slope of applied electric fields. The contour maps were drawn using a software, gOpenMol [48]. The input file is *Nitro_Benzene.dat* in the directory 'work'.

25 Charge doping

The following keyword is available for both the electron and hole dopings.

```
scf.system.charge      1.0      # default=0.0
```

The plus and minus signs correspond to hole and electron dopings, respectively. A partial charge doping is also possible. The excess charge given by the keyword 'scf.system.charge' is compensated by a uniform background opposite charge, since FFT is used to solve Poisson's equation in OpenMX. Therefore, if you compare the total energy between different charged states, a careful treatment is required, because additional electrostatic interactions induced by the background charge are included in the total energy. As an example, we show spin densities of hole doped, neutral, and electron doped (5,5) carbon nanotubes with a finite length of 14 Å in Fig. 23. The neutral and electron doped nanotubes possess the total spin moment of 1.0 and 2.2, while the total spin moment almost disappears in the hole doped nanotube. We can see that the spin polarization takes place at the edges of the neutral and electron doped nanotubes due to dangling bonds of edge regions.

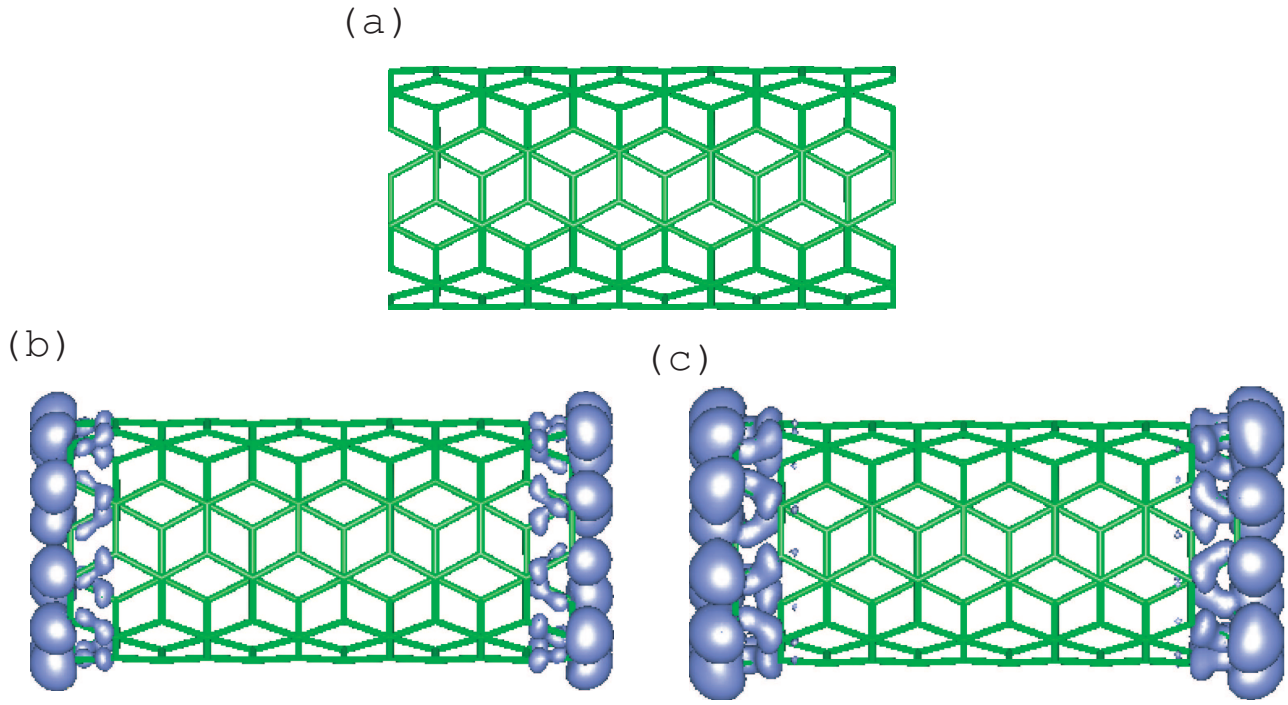


Figure 23: Spin densities of (a) four hole doped, (b) neutral, and (c) four electron doped (5,5) carbon nanotubes with a finite length of 14 Å. The input file is *Doped_NT.dat* in the directory 'work'.

26 Virtual atom with fractional nuclear charge

It is possible to treat a virtual atom with fractional nuclear charge by using a pseudopotential with the corresponding fractional nuclear charge. The pseudopotential for the virtual atom can be generated by ADPACK. The relevant keywords in ADPACK are given by

```
AtomSpecies             6.2
total.electron          6.2
valence.electron        4.2
<ocupied.electrons
 1    2.0
 2    2.0   2.2
ocupied.electrons>
```

The above example is for a virtual atom on the way of carbon and nitrogen atoms. Also, it is noted that basis functions for the pseudopotential of the virtual atom must be generated for the virtual atom with the same fractional nuclear charge, since the atomic charge density stored in *.pao is used to make the neutral atom potential.

As an illustration, the DOS of $C_{7.8}N_{0.2}$ calculated using the method is shown in Fig. 24. The input file is *DIA8-VA.dat* which can be found in the directory, work. In the calculation, one of eight carbon atoms in the unit cell was replaced by a virtual atom with an effective nuclear charge of 4.2, which corresponds to a stoichiometric compound of $C_{7.8}N_{0.2}$.

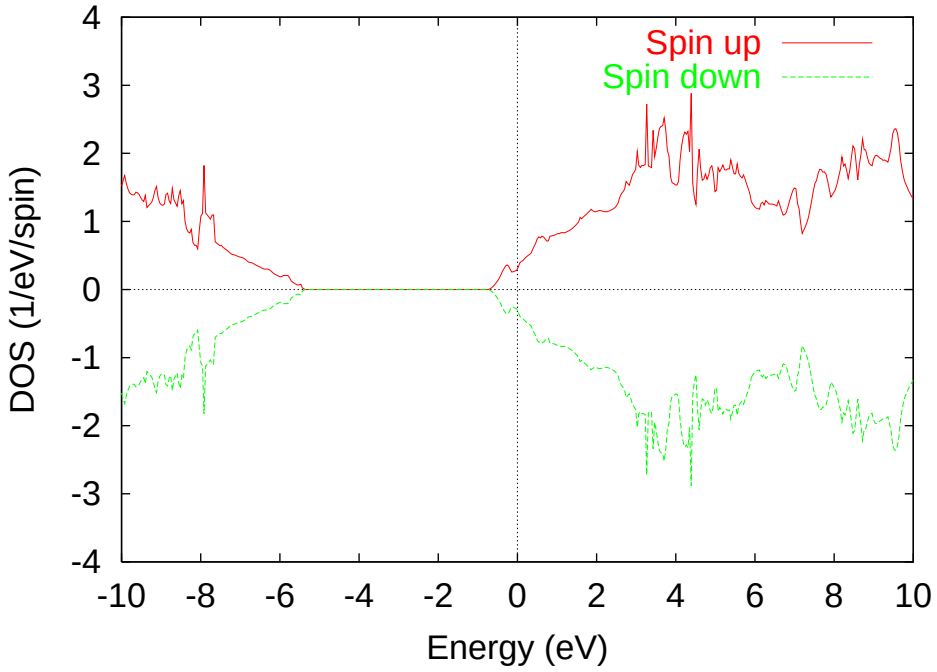


Figure 24: Density of states (DOS) of $C_{7.8}N_{0.2}$ calculated with a pseudopotential of the virtual atom. The input file used for the calculation is *DIA8-VA.dat* which can be found in the directory, work.

27 LCAO coefficients

It is possible to analyze LCAO coefficients in both the cluster and band calculations. In the cluster calculation, if a keyword, 'level.of.fileout', is set in 2, the LCAO coefficients are added into a file, *.out. As an example, LCAO coefficients of *Methane.dat* discussed in the Section 'Test calculation' are shown below:

```
***********************************************************
***********************************************************
  Eigenvalues (Hartree) and Eigenvectors for SCF KS-eq.
***********************************************************
***********************************************************

   Chemical Potential (Hartree) =   0.00000000000000
   HOMO =  4

   LCAO coefficients for up (U) and down (D) spins


                          1 (U)     2 (U)     3 (U)     4 (U)     5 (U)     6 (U)
                        -0.64276  -0.36132  -0.36128  -0.36128   0.26426   0.26446

   1   C 0 s            -0.61131   0.00000   0.00000   0.00000   0.00000  -0.00000
         0 px           -0.00000  -0.00013   0.00405   0.62805  -0.00006  -0.00327
         0 py            0.00000   0.62823  -0.00026   0.00013   1.17816   0.00012
         0 pz            0.00000  -0.00026  -0.62805   0.00405  -0.00012   1.17824
   2   H 0 s            -0.17052  -0.25665  -0.00223  -0.36325   0.68908  -0.00263
   3   H 0 s            -0.17052   0.25688   0.36309  -0.00229  -0.68923   0.97445
   4   H 0 s            -0.17052   0.25658  -0.36331   0.00239  -0.68903  -0.97459
   5   H 0 s            -0.17052  -0.25680   0.00245   0.36315   0.68918   0.00278

                          7 (U)     8 (U)
                         0.26446   0.31939

   1   C 0 s             0.00000   1.93736
         0 px           -1.17824   0.00000
         0 py           -0.00006   0.00000
         0 pz           -0.00327   0.00000
   2   H 0 s            -0.97456  -0.80393
   ......
   ....
```

In bulk calculations, if a keyword 'MO.fileout' is set in ON, LCAO coefficients at k-points which are specified by the keyword 'MO.kpoint' are output into a file *.out. For cluster calculations, 'level.of.fileout' should be 2 to output LCAO coefficients. But, for band calculations, the relevant keyword is 'MO.fileout' rather than 'level.of.fileout'.

28 Charge analysis

Although it is a somewhat ambiguous issue to assign effective charge to each atom, OpenMX provides three schemes, Mulliken charge analysis, Voronoi charge analysis, and electro static potential (ESP) fitting method, to analyze the charge state of each atom.

28.1 Mulliken charge

The Mulliken charges are output in *.out by default as shown in Section 'Test calculation'. In addition to the Mulliken charge projected to each atom, you can also find a decomposed Mulliken charge to each orbital in *.out. The output result stored in *.out for a methane molecule is as follows:

```
Decomposed Mulliken populations

  1    C          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.598003833  0.598003833   1.196007667   0.000000000
 sum over m      0.598003833  0.598003833   1.196007667   0.000000000
 sum over m+mul  0.598003833  0.598003833   1.196007667   0.000000000
px          0    0.588514081  0.588514081   1.177028163   0.000000000
py          0    0.588703212  0.588703212   1.177406424   0.000000000
pz          0    0.588514081  0.588514081   1.177028162   0.000000000
 sum over m      1.765731375  1.765731375   3.531462749   0.000000000
 sum over m+mul  1.765731375  1.765731375   3.531462749   0.000000000

  2    H          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.409066346  0.409066346   0.818132693   0.000000000
 sum over m      0.409066346  0.409066346   0.818132693   0.000000000
 sum over m+mul  0.409066346  0.409066346   0.818132693   0.000000000

  3    H          Up spin      Down spin     Sum           Diff
          multiple
s           0    0.409065912  0.409065912   0.818131824   0.000000000
 sum over m      0.409065912  0.409065912   0.818131824   0.000000000
 sum over m+mul  0.409065912  0.409065912   0.818131824   0.000000000

 .......
 ....
```

As you can see, the Mulliken charges are decomposed for all orbitals. There are two kind of summations in this decomposition. One of summations is 'sum over m' which means a summation over magnetic quantum number for each multiple orbital. The second summation is 'sum over m+mul' which means a summation over both magnetic quantum number and orbital multiplicity, where "multiple" means a number to specify a radial wave function. Therefore, Mulliken charges are decomposed to contributions of all orbitals.

28.2 Voronoi charge

Voronoi charge of each atom is calculated by integrating electron and spin densities in a Voronoi polyhedron. The Voronoi polyhedron is constructed from smeared surfaces which are defined by a Fuzzy cell partitioning method [40]. It should be noted that this Voronoi analysis gives often overestimated or underestimated charge, since Voronoi polyhedron is determined by only the structure without taking account of atomic radius. If you want to calculate Voronoi charge, specify the following keyword 'Voronoi.charge' in your input file:

```
Voronoi.charge      on       # on|off, default = off
```

In case of a methane molecule, the following Voronoi charges are output to *.out.

```
***********************************************************
***********************************************************
                    Voronoi charges
***********************************************************
***********************************************************

  Sum of Voronoi charges for up    =  3.999999031463
  Sum of Voronoi charges for down  =  3.999999031463
  Sum of Voronoi charges for total =  7.999998062926

  Total spin S by Voronoi charges  =  0.000000000000

                    Up spin      Down spin     Sum          Diff
      Atom=   1   1.137912511  1.137912511   2.275825021   0.000000000
      Atom=   2   0.715521700  0.715521700   1.431043399   0.000000000
      Atom=   3   0.715521486  0.715521486   1.431042973   0.000000000
      Atom=   4   0.715521776  0.715521776   1.431043552   0.000000000
      Atom=   5   0.715521559  0.715521559   1.431043118   0.000000000
```

Clearly, we see that carbon atom (Atom=1) and hydrogen atoms (Atom=2-5) tend to possess less charge and much charge, respectively, from a chemical sense. However, the Voronoi analysis could be a useful and complementary information for a bulk system with a closed pack structure.

28.3 Electro-static potential fitting

For small molecular systems, the electro-static potential (ESP) fitting method [54, 55, 56] is useful to determine an effective charge of each atom, while the ESP fitting method can not be applied for large molecules and bulk systems, since there are not enough sampling points for atoms far from surface areas in the ESP fitting method. In the ESP fitting method an effective point net charge on each atom is determined by a least square method with constraints so that the sum of the electro-static potential by effective point charges reproduce electro-static potential calculated by the DFT calculation as much as possible. The ESP fitting charge is calculated by the following two steps:

(1) SCF calculation

After finishing a usual SCF calculation, you have two output files:

```
*.out
*.vhart.cube
```

There is no additional keyword to generate the two files which are default output files by the SCF calculation, while the keyword 'level.of.stdout' should be 1 or 2.

(2) ESP fitting charge

Let us compile a program code for calculating the ESP fitting charge. Move the directory 'source' and then compile as follows:

```
% make esp
```

When the compile is completed normally, then you can find an executable file 'eps' in the directory 'work'. The ESP fitting charge can be calculated from two files *.out and *.vhart.cube using the program 'esp'. For example, you can calculate them for a methane molecule shown in the Section 'Input file' as follows:

```
% ./esp met -c 0 -s 1.4 2.0
```

Then, it is enough to specify the file name without the file extension, however, two files 'met.out' and 'met.vhart.cube' must exist in the directory 'work'. The options '-c' and '-s' are key parameters to specify a constraint and scale factors. You can find the following statement in the header part of a source code 'eps.c':

```
-c      constraint parameter
        '-c 0' means charge conservation
        '-c 1' means charge and dipole moment conservation
-s      scale factors for vdw radius
        '-s 1.4 2.0' means that 1.4 and 2.0 are 1st and 2nd scale factors
```

In this ESP fitting method, we support two constraints, charge conservation and, charge and dipole moment conservation. Although the later can reproduce charge and dipole moment calculated by the DFT calculation, it seems that the introduction of the dipole moment conservation gives often physically unacceptable point charges especially for a relatively large molecule. Thus, we would like to recommend the former constraint. The sampling points are given by the grids in the real space between two shells of the first and second scale factors times van der Waals radii [57]. In the above example, 1.4 and 2.0 correspond to the first and second scale factors. The calculated result appears in the standard output (your display) as follows:

```
 % ./eps met -c 0 -s 1.4 2.0

*********************************************************************
*********************************************************************
 esp: effective charges by a ESP fitting method
 Copyright (C), 2004, Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*********************************************************************
*********************************************************************

Constraint: charge
Scale factors for vdw radius    1.40000    2.00000
Number of grids in a van der Waals shell = 28464
Volume per grid =    0.0235870615 (Bohr^3)
Success

  Atom=   1  Fitting Effective Charge= -0.93558216739
  Atom=   2  Fitting Effective Charge=  0.23389552572
  Atom=   3  Fitting Effective Charge=  0.23389569182
  Atom=   4  Fitting Effective Charge=  0.23389535126
  Atom=   5  Fitting Effective Charge=  0.23389559858

  Magnitude of dipole moment    0.0000015089 (Debye)
  Component x y z     0.0000003114   -0.0000002455   -0.0000014558
RMS between the given ESP and fitting charges (Hartree/Bohr^3)= 0.096515449505
```

29 Non-collinear DFT

A fully unconstrained non-collinear density functional theory (DFT) is supported including the spin-orbit coupling (SOC) [6, 7, 8, 9, 13]. When the non-collinear DFT is performed, the following option for the keyword 'scf.SpinPolarization' is available.

```
scf.SpinPolarization        NC        # On|Off|NC
```

If the option 'NC' is specified, wave functions are expressed by a two components spinor. An initial spin orientation of each site is given by

```
<Atoms.SpeciesAndCoordinates           # Unit=Ang
  1  Mn    0.00000   0.00000   0.00000   8.0  5.0  45.0 0.0 45.0 0.0  1 on
  2  O     1.70000   0.00000   0.00000   3.0  3.0  45.0 0.0 45.0 0.0  1 on
Atoms.SpeciesAndCoordinates>


 1:    sequential serial number
 2:    species name
 3:    x-coordinate
 4:    y-coordinate
 5:    z-coordinate
 6:    initial occupation for up spin
 7:    initial occupation for down spin
 8:    Euler angle, theta, of the magnetic field for spin magnetic moment
 9:    Euler angle, phi, of the magnetic field for spin magnetic moment
       Also, the 8th and 9th are used to generate the initial non-collinear
       spin charge distribution
10:    the Euler angle, theta, of the magnetic field for orbital magnetic moment
11:    the Euler angle, phi, of the magnetic field for orbital magnetic moment
12:    switch for the constraint schemes specified by the keywords
       'scf.Constraint.NC.Spin', 'scf.NC.Zeeman.Orbital' and 'scf.NC.Zeeman.Orbital'.
       '1' means that the constraint is applied, and '0' no constraint.
13:    switch for enhancement of orbital polarization in the LDA+U method,
       'on' means that the enhancement is made, 'off' no enhancement.
```

The initial Euler angles, θ and ϕ, for orientation of the spin and orbital magnetic moment are given by the 8th and 9th columns, and 10th and 11th columns, respectively. The 12th column is a switch for a constraint scheme that a constraint (penalty or Zeeman) functional to the spin and orbital orientation is added on each site, where '1' means that the constraint functional is added, and '0' means no constraint. For the details of the constraint DFT for the spin orientation, see the Section 'Constraint DFT for non-collinear spin orientation'. The final 13th column is a switch for enhancement of orbital polarization in the LDA+U method, 'on' means that the enhancement is made, 'off' no enhancement. Figure 25 shows the spin orientation in a MnO molecule calculated by the non-collinear DFT. You can follow the calculation using an input file *Mol_MnO_NC.dat* in the directory 'work'. To visualize the spin orientation in the real space, two files are generated:

```
*.nc.txt
*.ncsden.txt
```

where * means 'System.Name' you specified. Two files '*.nc.txt' and '*.ncsden.txt' store a projected spin orientation to each atom by Mulliken analysis and the spin orientation on real space grids in a vector file format supplied by gOpenMol. Both the files can be visualized using 'Plot Vector File' in gOpenMol as shown in Fig. 25.

The spin moment and Euler angles of each atom, which are calculated by Mulliken analysis, are found in the *.out file as follows:

```
************************************************************
************************************************************
                  Mulliken populations
************************************************************
************************************************************

  Total spin moment (muB)   4.997792547   Angles (Deg) 45.001196562   0.000000622

                Up       Down       Sum       Diff        theta       phi
 1   Mn    8.63989   3.91894   12.55883   4.72096    44.99801    0.00000
 2    O    3.35900   3.08217    6.44117   0.27684    45.05555    0.00000
```

Also it should be noted that it is difficult to achieve a self consistent field in the non-collinear DFT more than the collinear DFT calculation, since there are many minima, having almost comparable energy, in the spin orientation space, while the constraint DFT is useful for such a case.

In the non-collinear DFT, the inclusion of spin-orbit coupling is supported, while it is not supported for the collinear DFT. See also the Section 'Relativistic effects' for the issue.

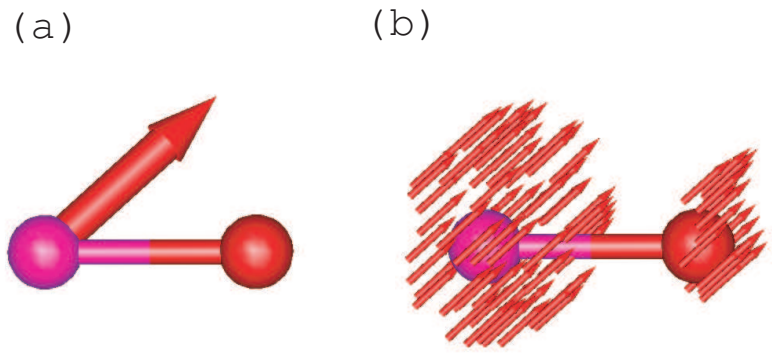


Figure 25: Spin orientation in (a) a projected form on each atom and (b) a real space representation of a MnO molecule calculated by the non-collinear DFT. The figures were visualized by a command 'Plot Vector File' in gOpenMol [48]. The input file is *Mol_MnO_NC.dat* in the directory 'work'.

30 Relativistic effects

Relativistic effects can be incorporated by a fully relativistic and a scalar relativistic pseudopotentials. In the fully relativistic treatment, the spin-orbit coupling is included in addition to kinematic relativistic effects (Darwin and mass velocity terms). On the other hand, the spin-orbit coupling is averaged in the scalar relativistic treatment. Although the scalar relativistic treatment can be incorporated in both the collinear and non-collinear DFT calculations, the fully relativistic treatment is supported for only the non-collinear DFT in the current version of OpenMX.

30.1 Fully relativistic

The fully relativistic effects including the spin-orbit coupling within the pseudopotential scheme can be included in the non-collinear DFT calculations [10, 19, 13], while the inclusion of the spin-orbit coupling is not supported in the collinear DFT calculation. The inclusion of fully relativistic effects is made by the following two steps:

(1) Making of j-dependent pseudopotentials

First, you are requested to generate j-dependent pseudopotentials using ADPACK. For your convenience, the j-dependent pseudopotentials are available for several elements in the database [65]. The details how to make the j-dependent pseudopotential are found in the manual of ADPACK.

(2) SCF calculation

If you specify j-dependent pseudopotentials in the specification of '<Definition.of.Atomic.Species', it is possible to include spin-orbit coupling by the following keyword 'scf.SpinOrbit.Coupling':

```
scf.SpinOrbit.Coupling      on          # On|Off, default=off
```

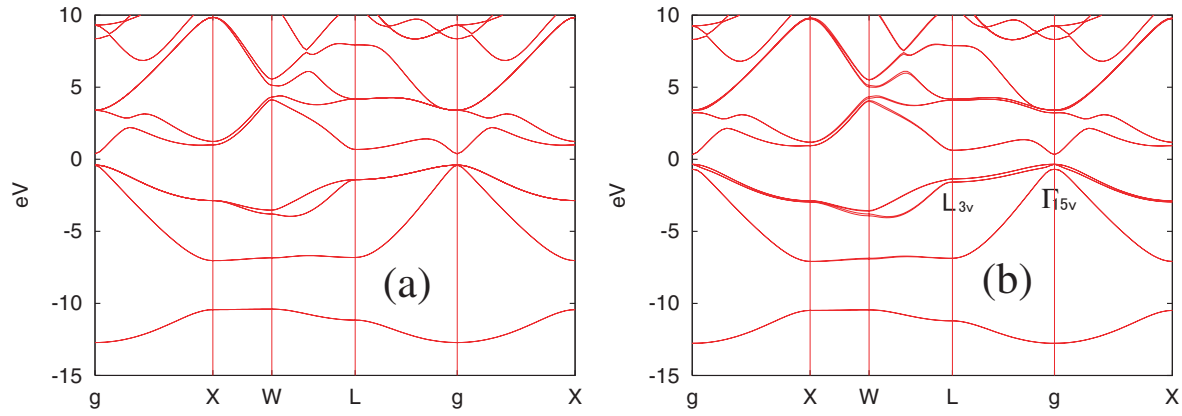


Figure 26: Band structures of a bulk GaAs calculated by the non-collinear DFT (a) without and (b) with the spin-orbit coupling. In these calculations, Ga6.5-s2p2d1 and As6.5-s2p2d1 were used as a basis set, and Ga_LDA.vps and As_LDA.vps were used for pseudopotentials, which are stored in the database. For the exchange-correlation terms, LDA was used. We used 12× 12× 12 and 140 (Ryd) for scf.Kgrid and scf.energycutoff, respectively. Also the experimental value (5.65Å) was used for the lattice constant. The input file is *GaAs.dat* in the directory 'work'.

Table 3: Calculated spin-orbit splittings (eV) at the Γ_{15v} and the L_{3v} of a buld GaAs. The other theoretical values (LMTO: Ref.[58], PP: Ref.[59]) and experimental value (Ref.[60]) are also shown for comparison. The calculation conditions are given in the caption of Fig. 26 and the input file is *GaAs.dat* in the directory 'work'.

Level	OpenMX	LMTO	PP	Expt.
Γ_{15v}	0.348	0.351	0.35	0.34
L_{3v}	0.218	0.213	0.22	

Then, the spin-orbit coupling can be self-consistently incorporated within the pseudopotential scheme rather than a perturbation scheme. Due to the spin-orbit coupling, α and β spin components in the two components spinor can directly interact. In order to determine the absolute spin orientation in the non-collinear DFT calculations, you have to include the spin-orbit coupling, otherwise the spin orientation is not uniquely determined in the real space. As an illustration of spin-orbit splitting, we show band structures of a bulk GaAs calculated by the non-collinear DFT without and with spin-orbit coupling in Fig. 26, where the input file is *GaAs.dat* in the directory 'work'. In Fig. 26(b) we can see that there are spin-orbit splittings in the band dispersion, while no spin-orbit splitting is not observed in Fig. 26(a). The spin-orbit splittings at two k-points, Γ and L, are listed together with the other calculations and experimental values in Table 3. We see a good agreement in this table.

30.2 Scalar relativistic treatment

A simple way to incorporate a scalar relativistic treatment is to use scalar relativistic pseudopotentials which can be generated by ADPACK. The another way is to use fully relativistic j-dependent pseudopotentials and to switch off the keyword 'scf.SpinOrbit.Coupling' as follows:

```
scf.SpinOrbit.Coupling      off         # On|Off, default=off
```

Then, the j-dependent pseudopotentials are automatically averaged with a weight of j-degeneracy when they are read by OpenMX, which corresponds to scalar relativistic pseudopotentials. So, once j-dependent pseudopotentials are generated, you can utilize the pseudopotentials for both the fully and scalar relativistic treatment. Thus, we recommend that you make a fully relativistic j-dependent pseudopotential rather than a scalar relativistic pseudopotential, when relativistic effects are taken into account. In fact, the calculation in Fig. 26(a) was performed with 'scf.SpinOrbit.Coupling=off' and the same pseudopotential as in Fig. 26(b).

31 Orbital magnetic moment

The orbital magnetic moment at each atomic site is calculated as default in the non-collinear DFT. Since the orbital magnetic moment appears as a manifestation of spin-orbit coupling (SOC), the calculated values become finite when the SOC is included [63, 64]. As an example, a non-collinear LDA+U (U=5eV) calculation of iron monoxide bulk is illustrated using an input file *FeO_NC.dat* in the directory 'work'. As for the LDA+U calculation, see the Section 'LDA+U'. The calculated orbital and spin magnetic moments at the Fe site are listed in Table 4. Also, you can find the orientation of the (decomposed) orbital moment in *.out, where * means 'System.Name' as follows:

```
***********************************************************
***********************************************************
                    Orbital moments
***********************************************************
***********************************************************

  Total Orbital Moment (muB)   0.000000070   Angles  (Deg) 113.644105951  -65.722115195

        Orbital moment (muB)   theta (Deg)  phi (Deg)
   1   Fe   1.01127           128.64444   50.80973
   2   Fe   1.01127            51.35556  230.80973
   3    O   0.00000           122.13287    8.40916
   4    O   0.00000            58.29296  151.31925


 Decomposed Orbital Moments

   1   Fe         Orbital Moment(muB)    Angles (Deg)
          multiple
s           0    0.000000000           90.0000    0.0000
 sum over m      0.000000000           90.0000    0.0000
s           1    0.000000000           90.0000    0.0000
 sum over m      0.000000000           90.0000    0.0000
px          0    0.000032282           44.0757   90.0000
py          0    0.000027194           31.5419   -0.0000
pz          0    0.000026842           90.0000   57.4970
 sum over m      0.000070741           49.0444   57.5709
px          1    0.004596036           10.8026  -90.0000
py          1    0.004533432            5.2237  180.0000
pz          1    0.000955444           90.0000  244.3929
 sum over m      0.009229130           11.9479  244.3959
d3z^2-r^2   0    0.045401124           90.0000  224.3492
dx^2-y^2    0    0.075657665           24.3023  228.5632
dxy         0    0.453606172           81.2632   50.2745
dxz         0    0.495766350          143.9475  -10.8730
```

```
dyz          0    0.531382963          138.9632   98.7434
 sum over m       0.997255210          131.7287   51.1391
d3z^2-r^2    1    0.001075694           90.0000  254.7742
dx^2-y^2     1    0.012694575           26.6388  225.7504
dxy          1    0.036086417           71.5849   49.3240
dxz          1    0.031150186          132.6513  -13.0079
dyz          1    0.033740724          128.7200   99.3874
 sum over m       0.058459849          109.4476   49.1020
f5z^2-3r^2   0    0.007365273           90.0000   39.4321
f5xz^2-xr^2  0    0.005659459           26.2551  124.3549
f5yz^2-yr^2  0    0.006152658           34.4173  -38.4581
fzx^2-zy^2   0    0.015290504           34.2465  224.2021
fxyz         0    0.012904266           11.6263  244.9193
fx^3-3*xy^2  0    0.004957037           43.3387  -84.7645
f3yx^2-y^3   0    0.004826463           41.6700  183.4396
 sum over m       0.043385660           10.6323  246.7139
.....
...
```

As shown in Table 4, OpenMX gives a good agreement for both the spin and orbital magnetic moments of a series of 3*d*-transition metal oxides with other calculation results. However, it is noted that the absolute value of orbital magnetic moment seems to be significantly influenced by calculation conditions such as basis functions and on-site 'U' in the LDA+U method, while the spin magnetic moment is relatively insensitive to the calculation conditions, and that a rather rich basis set including polarization functions will be needed for convergent calculations of the orbital magnetic moment.

Table 4: Spin magnetic moment $M_s(\mu_B)$ and orbital magnetic moment $M_o(\mu_B)$ of transition metal oxides, MO (M=Mn,Fe,Co,Ni). In the LDA+U scheme [16], for the first d-orbital of M, the effective U of 4.0 (eV) for Mn, 5.0 (eV) for Fe, Co for 7.0 (eV), and Ni for 7.0 (eV) were used. For the others zero. The local spin moment was calculated by the Voronoi decomposition discussed in the Section 'Voronoi charge' rather than Mulliken charge, since the Mulliken analysis tends to give a larger spin moment in the use of multiple basis functions. The input files are *MnO_NC.dat*, *FeO_NC.dat*, *CoO_NC.dat*, and *NiO_NC.dat* in the directory 'work'. The other theoretical value [41] and experimental value [41] are also shown for comparison.

	M_s		M_o		
Compound	OpenMX	Other calc.	OpenMX	Other calc.	Expt. in total
MnO	4.560	4.49	0.001	0.00	4.79,4.58
FeO	3.582	3.54	1.010	1.01	3.32
CoO	2.684	2.53	1.088	1.19	3.35,3.8
NiO	1.594	1.53	0.173	0.27	1.77,1.64,1.90

32 LDA+U

LDA+U methods with different definitions of the occupation number operator [16] are available for both the collinear and non-collinear calculations by the following keyword 'scf.Hubbard.U':

```
scf.Hubbard.U              on       # On|Off, default=off
```

The occupation number operator is specified by the following keyword 'scf.Hubbard.Occupation':

```
scf.Hubbard.Occupation     dual     # onsite|full|dual, default=dual
```

Among three occupation number operators, only the 'dual' operator satisfies a sum rule that the trace of occupation number matrix gives the total number of electrons which is the most primitive conserved quantity in a Hubbard model. For the details of the operator 'onsite', 'full', and 'dual', see Ref. [16]. The effective U-value in eV on each orbital of species defined by

```
<Definition.of.Atomic.Species
 Ni  Ni5.5-s2p2d2      Ni_LDA
 O   O4.5-s2p2d1       O_LDA
Definition.of.Atomic.Species>
```

is specified by

```
<Hubbard.U.values                  #  eV
 Ni  1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 4.0 2d 0.0
 O   1s 0.0 2s 0.0 1p 0.0 2p 0.0 1d 0.0
Hubbard.U.values>
```

The beginning of the description must be <Hubbard.U.values, and the last of the description must be Hubbard.U.values>. For all the basis orbitals, you have to give an effective U-value in eV in above format. The '1s' and '2s' mean the first and second s-orbital, and the number behind '1s' is the effective U-value for the first s-orbital. The same rule is applied to p- and d-orbitals. As an example of the LDA+U calculation, the density of states for a nickel monoxide bulk is shown for cases with an effective U-value of 0 and 4 (eV) for d-orbitals of Ni in Fig. 27, where the input file is *Crys-NiO.dat* in the directory 'work'. We see that the gap increases due to the introduction of a Hubbard term on the d-orbitals. The occupation number for each orbital is output to *.out file in the same form as that of decomposed Mulliken populations which starts from the title 'Occupation Number in LDA+U' as follows:

```
***********************************************************
***********************************************************
       Occupation Number in LDA+U and Constraint DFT

    Eigenvalues and eigenvectors for a matrix consisting
          of occupation numbers on each site
***********************************************************
***********************************************************
```

```
  1   Ni

   spin= 0

Sum =  8.674098295491

                  1       2       3       4       5       6       7       8
Individual     -0.0016  0.1334  0.1334  0.1349  0.2903  0.9948  0.9948  0.9953

s           0  -0.0111  0.0000 -0.0004 -0.0004  0.9999  0.0000 -0.0059 -0.0045
s           1   0.9999  0.0000  0.0003  0.0077  0.0111 -0.0000  0.0023  0.0016
px          0   0.0019  0.0383  0.0201 -0.0324 -0.0042 -0.6993 -0.7016 -0.0055
py          0   0.0020  0.0000 -0.0448 -0.0278 -0.0043 -0.0000  0.0059 -0.9850
pz          0   0.0019 -0.0383  0.0201 -0.0324 -0.0042  0.6995 -0.7014 -0.0055
px          1  -0.0044 -0.7060 -0.3724  0.5996 -0.0002 -0.0374 -0.0396 -0.0029
py          1  -0.0042 -0.0002  0.8486  0.5259  0.0003 -0.0000 -0.0019 -0.0539
pz          1  -0.0044  0.7062 -0.3720  0.5996 -0.0002  0.0374 -0.0395 -0.0029
d3z^2-r^2   0   0.0000  0.0028 -0.0016  0.0001  0.0003  0.1080 -0.0367  0.0589
dx^2-y^2    0   0.0000 -0.0016 -0.0028  0.0001  0.0004 -0.0623 -0.0636  0.1021
dxy         0  -0.0000 -0.0034  0.0036  0.0229  0.0006 -0.0414  0.0590  0.0406
dxz         0   0.0002  0.0000 -0.0024  0.0232  0.0006 -0.0000  0.0168  0.0976
dyz         0  -0.0000  0.0034  0.0036  0.0229  0.0006  0.0414  0.0590  0.0406

                  9      10      11      12      13
Individual     0.9989  0.9989  0.9995  1.0006  1.0007

s           0  -0.0000  0.0000 -0.0000 -0.0000  0.0004
.....
...
```

The eigenvalues of the occupation number matrix of each atomic site correspond to the occupation number to each local state given by the eigenvector.

The LDA+U functional possesses multiple minima in the degree of freedom of the orbital occupation, leading to that the SCF calculation tends to be trapped to some local minimum. To find the ground state with an orbital polarization, a way of enhancing explicitly the orbital polarization is available by the following switch :

```
For collinear cases

<Atoms.SpeciesAndCoordinates           # Unit=AU
 1  Ni   0.0       0.0       0.0       10.0  6.0  on
 2  Ni   3.94955   3.94955   0.0        6.0 10.0  on
 3   O   3.94955   0.0       0.0        3.0  3.0  on
 4   O   3.94955   3.94955   3.94955    3.0  3.0  on
```

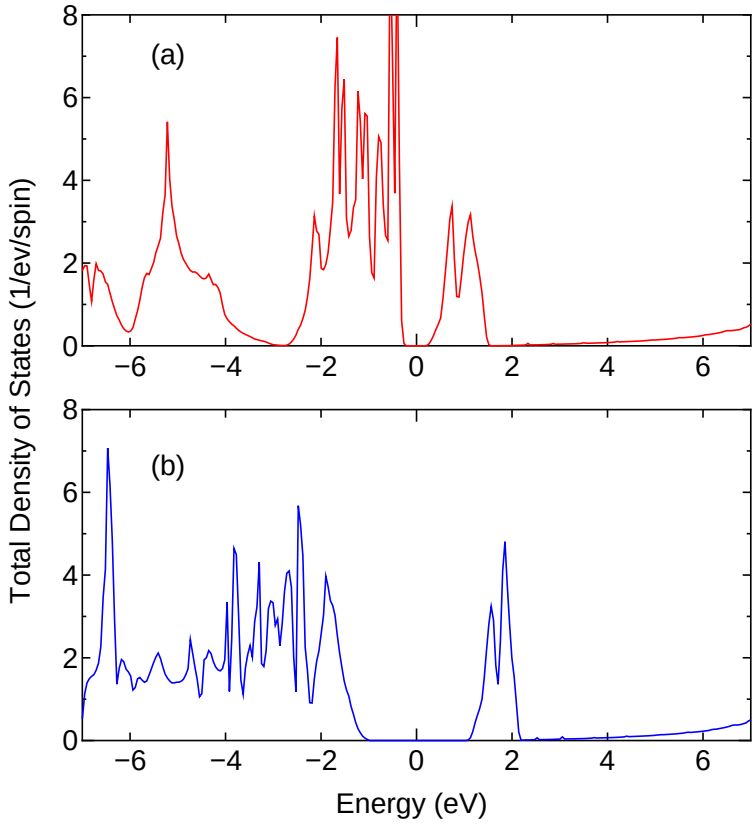


Figure 27: The total density of states for up-spin in NiO bulk calculated with (a) U=0 (eV) and (b) U=4 (eV) in the LDA+U method. The input file is *Crys-NiO.dat* in the directory 'work'.

```
Atoms.SpeciesAndCoordinates>

For non-collinear cases

<Atoms.SpeciesAndCoordinates            # Unit=AU
 1  Ni   0.0       0.0       0.0       10.0  6.0  40.0 10.0 0 0 on
 2  Ni   3.94955   3.94955   0.0        6.0 10.0  40.0 10.0 0 0 on
 3   O   3.94955   0.0       0.0        3.0  3.0  10.0 40.0 0 0 on
 4   O   3.94955   3.94955   3.94955    3.0  3.0  10.0 40.0 0 0 on
Atoms.SpeciesAndCoordinates>
```

The specification of each column can be found in the section 'Non-collinear DFT'. Since the enhancement treatment for the orbital polarization is performed on each atom, you have to set the switch for all the atoms in the specification of atomic coordinates as given above. The enhancement for the atoms switched on is applied during the first few self-consistent (SC) steps, then no more enhancement are required during the subsequent SC steps. It is also emphasized that the enhancement does not always give the ground state, and that it can work badly in some case. See Ref. [16] for the details.

33 Constraint DFT for non-collinear spin orientation

To calculate an electronic structure with an arbitrary spin orientation in the non-collinear DFT, OpenMX Ver. 3.5 provides a constraint functional which gives a penalty unless the difference between the calculated spin orientation and the initial one is zero [11]. The constraint DFT for the non-collinear spin orientation is available by the following keywords:

```
scf.Constraint.NC.Spin       on      # on|off, default=off
scf.Constraint.NC.Spin.v     0.2     # default=0.0(eV)
```

You can switch on the keyword 'scf.Constraint.NC.Spin' and give a magnitude by 'scf.Constraint.NC.Spin.v' which determines the strength of constraint, when the constraint for the spin orientation is introduced. The constraint is applied on each atom by specifying a switch as follows:

```
<Atoms.SpeciesAndCoordinates
   1   Cr    0.00000   0.00000   0.00000  7.0  5.0 -20.0 0.0  1  off
   2   Cr    0.00000   2.00000   0.00000  7.0  5.0  20.0 0.0  1  off
Atoms.SpeciesAndCoordinates>
```

The '1' in the 12th column means that the constraint is applied, and '0' no constraint. The method constrains only the spin orientation. Therefore, the magnitude of spin can vary. Also the constraint scheme is compatible with the LDA+U calculation explained in the Section 'LDA+U'. As an illustration of this method, the dependence of the total energy and magnetic moment in a chromium dimer on the relative angle between two local spins is shown in Fig. 28. You can trace the calculation using an input file *Cr2_CNC.dat* in the directory 'work'.

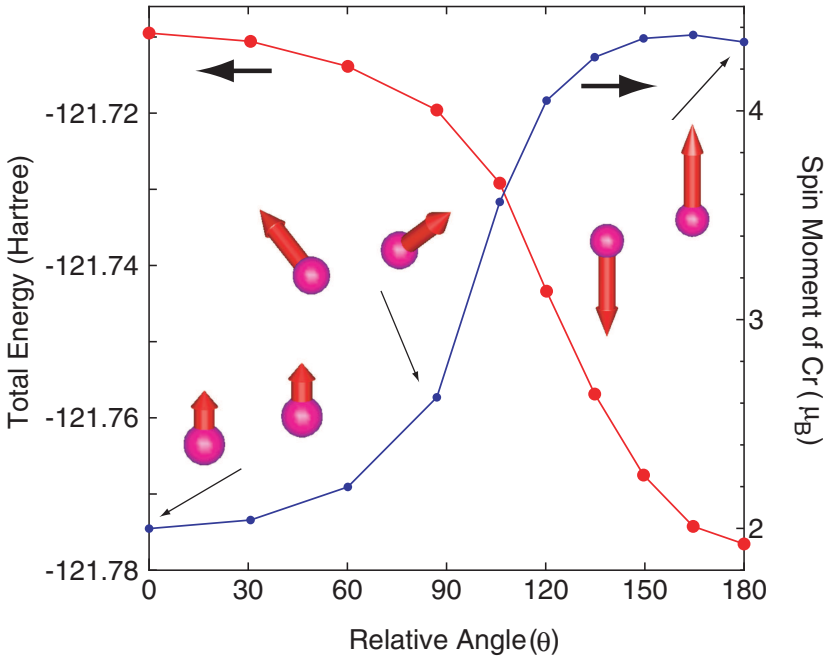


Figure 28: The total energy and magnetic moment of Cr atom for a chromium dimer of which bond length is 2.0 Å. The input file is *Cr2_CNC.dat* in the directory 'work'.

34 Zeeman terms

It is possible to apply Zeeman terms to spin and orbital magnetic moments.

34.1 Zeeman term for spin magnetic moment

The Zeeman term for spin magnetic moment is available as an interaction with a uniform magnetic field by the following keywords:

```
scf.NC.Zeeman.Spin            on        # on|off, default=off
scf.NC.Mag.Field.Spin        100.0      # default=0.0(Tesla)
```

When you include the Zeeman term for spin magnetic moment, switch on the keyword 'scf.NC.Zeeman.Spin'. The magnitude of the uniform magnetic field can be specified by the keyword 'scf.NC.Mag.Field.Spin' in units of Tesla. Moreover, we extend the scheme as a constraint scheme in which the direction of the magnetic field can be different from each atomic site atom by atom. Then, the direction of magnetic field for spin magnetic moment can be controlled, for example, by the keyword 'Atoms.SpeciesAndCoordinates':

```
<Atoms.SpeciesAndCoordinates
 1  Sc  0.000  0.000  0.000   6.6 4.4  10.0 50.0  160.0 20.0  1  on
 2  Sc  2.000  0.000  0.000   6.6 4.4  80.0 50.0  160.0 20.0  1  on
Atoms.SpeciesAndCoordinates>
```

The 8th and 9th columns give the Euler angles, theta and phi, in order to specify the magnetic field for spin magnetic moment. The 12th column is a switch to the constraint. '1' means that the constraint is applied, and '0' no constraint. Since for each atomic site a different direction of the magnetic field can be applied, this scheme provides a way of studying non-collinear spin configuration. It is noted that the keyword 'scf.NC.Zeeman.Spin' and the keyword 'scf.Constraint.NC.Spin' are mutually exclusive. Therefore, when 'scf.NC.Zeeman.Spin' is 'on', the keyword 'scf.Constraint.NC.Spin' must be switched off as follows:

```
scf.Constraint.NC.Spin        off       # on|off, default=off
```

Although the Zeeman term and the constraint scheme for spin orientation can be regarded as ways for controlling the spin orientation, it is noted that the magnitude of spin magnetic moment by the Zeeman term tends to be enhanced unlike the constraint scheme.

34.2 Zeeman term for orbital magnetic moment

The Zeeman term for orbital magnetic moment is available as an interaction with a uniform magnetic field by the following keywords:

```
scf.NC.Zeeman.Orbital         on        # on|off, default=off
scf.NC.Mag.Field.Orbital     100.0      # default=0.0(Tesla)
```

When you include the Zeeman term for orbital magnetic moment, switch on the keyword 'scf.NC.Zeeman.Orbital'. The magnitude of the uniform magnetic field can be specified by the keyword 'scf.NC.Mag.Field.Orbital' in units of Tesla. Moreover, we extend the scheme as a constraint scheme in which the direction of the magnetic field can be different from each atomic site atom by atom. Then, the direction of magnetic field for orbital magnetic moment can be controlled, for example, by the keyword 'Atoms.SpeciesAndCoordinates':

```
<Atoms.SpeciesAndCoordinates
 1  Sc  0.000  0.000  0.000   6.6 4.4  10.0 50.0  160.0 20.0  1  on
 2  Sc  2.000  0.000  0.000   6.6 4.4  80.0 50.0  160.0 20.0  1  on
Atoms.SpeciesAndCoordinates>
```

The 10th and 11th columns give the Euler angles, theta and phi, in order to specify the magnetic field for orbital magnetic moment. The 12th column is a switch to the constraint. '1' means that the constraint is applied, and '0' no constraint. Since for each atomic site a different direction of the magnetic field can be applied, this scheme provides a way of studying non-collinear orbital configuration. Also, it is noted that the direction of magnetic field for orbital magnetic moment can be different from that for spin moment.

35 Macroscopic polarization by Berry's phase

The macroscopic electric polarization of a bulk system can be calculated based on a Berry's phase formalism [12]. As an example, let us illustrate a calculation of a Born effective charge of Na in a NaCl bulk via the macroscopic polarization.

(1) SCF calculation

First, perform a conventional SCF calculation using an input file *NaCl.dat* in the directory 'work'. Then, the following keyword 'HS.fileout' should be switched on

```
HS.fileout              on     # on|off, default=off
```

When the calculation is completed normally, then you can find an output file 'nacl.scfout' in the directory 'work'.

(2) Calculation of macroscopic polarization

The macroscopic polarization is calculated by a post-processing code 'polB' of which input data is 'nacl.scfout'. In the directory 'source', compile as follows:

```
% make polB
```

When the compile is completed normally, then you can find an executable file 'polB' in the directory 'work'. Then, move to the directory 'work', and perform as follows:

```
 % polB nacl.scfout
or
 % polB nacl.scfout < in > out
```

In the later case, the file 'in' contains the following ingredients:

```
9 9 9
1 1 1
```

In the former case, you will be interactively asked from the program as follows:

```
*****************************************************************
*****************************************************************
 polB:
 code for calculating the electric polarization of bulk systems
 Copyright (C), 2006-2007, Fumiyuki Ishii and Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*****************************************************************
*****************************************************************

Read the scfout file (nacl.scfout)
 Previous eigenvalue solver = Band
```

```
 atomnum                    = 2
 ChemP                      = -0.234375000000 (Hartree)
 E_Temp                     = 300.000000000000 (K)
 Total_SpinS                =  0.000000000000 (K)
 Spin treatment             = collinear spin-polarized

 r-space primitive vector (Bohr)
  tv1=  0.000000   5.319579   5.319579
  tv2=  5.319579   0.000000   5.319579
  tv3=  5.319579   5.319579   0.000000
 k-space primitive vector (Bohr^-1)
  rtv1= -0.590572   0.590572   0.590572
  rtv2=  0.590572  -0.590572   0.590572
  rtv3=  0.590572   0.590572  -0.590572

  Cell_Volume=301.065992 (Bohr^3)

 Specify the number of grids to discretize reciprocal a-, b-, and c-vectors
 (e.g 2 4 3)  9 9 9
   k1    0.00000    0.11111    0.22222    0.33333    0.44444    0.55556 ....
   k2    0.00000    0.11111    0.22222    0.33333    0.44444    0.55556 ....
   k3    0.00000    0.11111    0.22222    0.33333    0.44444    0.55556 ....

 Specify the direction of polarization as reciprocal a-, b-, and c-vectors
 (e.g 0 0 1 ) 1 1 1
```

Then, the calculation will start like this:

```
calculating the polarization along the a-axis ....
The number of strings for Berry phase : AB mesh=81

  calculating the polarization along the a-axis ....   1/  82
  calculating the polarization along the a-axis ....   2/  82
  .....
  ...
*********************************************************
              Electric dipole  (Debye) : Berry phase
*********************************************************

 Absolute dipole moment      163.93374185

               Background        Core             Electron          Total

 Dx           -0.00000000       94.64718996        0.00000013       94.64719009
```

```
 Dy          -0.00000000       94.64718996        0.00000013       94.64719009
 Dz          -0.00000000       94.64718996       -0.00000018       94.64718978


************************************************************
             Electric polarization (muC/cm^2) : Berry phase
************************************************************

              Background        Core              Electron          Total

 Px          -0.00000000      707.66166752        0.00000095      707.66166847
 Py          -0.00000000      707.66166752        0.00000095      707.66166847
 Pz          -0.00000000      707.66166752       -0.00000138      707.66166614

Elapsed time = 15.445211 (s) for myid=  0
```

Since the Born effective charge $Z^*_{\alpha\beta}$ is defined by a tensor:

$$Z^*_{\alpha\beta} = \frac{V_c}{|e|}\frac{\Delta P_\alpha}{\Delta u_\beta}$$

where V_c is the volume of the unit cell, e the elementary charge, Δu_β displacement along β-coordinate, ΔP_α the change of macroscopic polarization along α-coordinate, therefore we will perform above procedures (1) and (2) at least two or three times by varying the x, y, or z-coordinate of Na atom. Then, for example x-coordinates, we have

```
Px = 94.39158732 (Debye/unit cell) at x= -0.05 (Ang)
Px = 94.64719009 (Debye/unit cell) at x=  0.0  (Ang)
Px = 94.90279293 (Debye/unit cell) at x=  0.05 (Ang)
```

Thus,

$$\begin{aligned} Z^*_{xx} &= \frac{(94.90279293 - 94.39158732)/(2.54174776)}{0.1/0.529177} \\ &= 1.064 \end{aligned}$$

Table 5: Calculated Born effective charge of Na in a NaCl bulk The input file is *NaCl.dat* in the directory 'work'. The other theoretical value (FD: Ref. [61]) and experimental value (Ref. [62]) are also shown for comparison.

	OpenMX	FD	Expt.
Z^*	1.06	1.09	1.12

In the NaCl bulk the off-diagonal terms in the tensor of Born charge are zero, and $Z^*_{xx} = Z^*_{yy} = Z^*_{zz}$. In Table 5 we see that the calculated value is in good agreement with the other calculation [61] and an experimental result [62]. The calculation of macroscopic polarization is supported for both the collinear and non-collinear DFT. It is also noted that the code 'polB' has been parallelized for large-scale systems where the number of processors can exceed the number of atoms in the system.

36 Exchange coupling parameter

To analyze an effective interaction between spins located on two atomic sites, an exchange coupling parameter between two localized spins can be evaluated based on Green's function method [14, 15]. In OpenMX Ver. 3.5 the evaluation is supported for only the collinear calculations of cluster and bulk systems. If you want to calculate the exchange coupling parameter between two spins which are localized to different atomic sites, you can calculate it by the following two steps:

(1) SCF calculation

First, you would perform a collinear DFT calculation using an input file *Fe2.dat* in the directory 'work' as an example. Then, you have to set the following keyword 'HS.fileout' as follows:

```
HS.fileout                    on       # on|off, default=off
```

When the execution is completed normally, then you can find a file 'fe2.scfout' in the directory 'work'.

(2) Calculation of exchange coupling parameter

Let us make a program code for calculating the exchange coupling parameter. Move the directory 'source' and then compile as follows:

```
% make jx
```

When the compile is completed normally, then you can find an executable file 'jx' in the directory 'work'. The exchange coupling parameter can be calculated from the file '*.scfout' using the program 'jx' as follows:

```
% ./jx fe2.scfout
```

where an iron dimer is considered as an example. Then, you are interactively asked from the program as follow:

```
*******************************************************************
*******************************************************************
 jx: code for calculating an effective exchange coupling constant J
 Copyright (C), 2003, Myung Joon Han, Jaejun Yu, and Taisuke Ozaki
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*******************************************************************
*******************************************************************

Read the scfout file (fe2.scfout)
 Previous eigenvalue solver = Cluster
 atomnum                    = 2
 ChemP                      =  0.001924071047 (Hartree)
 E_Temp                     = 600.000000000000 (K)
```

```
Evaluation of J based on cluster calculation
 Diagonalize the overlap matrix
 Diagonalize the Hamiltonian for spin= 0
 Diagonalize the Hamiltonian for spin= 1

 Specify two atoms (e.g 1 2, quit: 0 0)  1 2
 J_ij between 1th atom and 2th atom is 949.978344353523 cm^{-1}
 Specify two atoms (e.g 1 2, quit: 0 0)  2 1
 J_ij between 2th atom and 1th atom is 949.978344353523 cm^{-1}
 Specify two atoms (e.g 1 2, quit: 0 0)  0 0
```

Please specify two atoms you want to calculate the exchange coupling parameter until typing '0 0'.

37 Optical conductivity

The functionality suffers from some program bugs. The revised code will be released in future.

The optical conductivity can be evaluated within linear response theory [42]. OpenMX Ver. 3.5 supports the calculation for only the collinear cluster calculation. If you want to calculate the optical conductivity of molecular systems, you can calculate it by the following two steps:

(1) SCF calculation

First, you would perform a collinear cluster calculation using an input file *Methane_OC.dat* in the directory 'work' as an example. Then, you have to set the following two keywords 'Dos.fileout' and 'OpticalConductivity.fileout' as follows:

```
Dos.fileout                    on    # on|off, default=off
OpticalConductivity.fileout    on    # on|off, default=off
```

When the execution is completed normally, then you can find files, *.optical and *.Dos.val, in the directory 'work'.

(2) Calculation of optical conductivity

Let us make a program code for calculating the optical conductivity. Move the directory 'source' and then compile as follows:

```
% make OpticalConductivityMain
```

When the compile is completed normally, then you can find a executable file 'OpticalConductivityMain' in the directory 'work'. The optical conductivity can be calculated from the files '*.optical' and '*.Dos.val' using the program 'OpticalConductivityMain' as follows:

```
% ./OpticalConductivityMain met.optical met.Dos.val met.optout
```

where a methane molecule is considered as an example. Then, you are interactively asked from the program as follow:

```
 # freqmax=100.000000
 # gaussian=0.036749
 freqmax (Hartree)=? 3
 freq mech=? 1000
```

In the output file 'met.optout' the second, third, and fourth columns correspond to the frequency (Hartree) and optical conductivity (arbitrary unit) for up- and down-spins, respectively.

38 Electric transport calculations

38.1 General

Electronic transport properties of molecules, nano-wires, and bulks such as superlattice structures can be calculated based on a non-equilibrium Green function (NEGF) method within the collinear DFT. The features and capabilities are listed below:

- SCF calculation of system with two leads under zero and a finite bias voltages
- SCF calculation under gate bias voltage
- Compatible with the LDA+U method
- Spin-dependent transmission and current
- **k**-resolved transmission and current along perpendicular to the current axis
- Calculation of current-voltage curve
- Accurate and efficient contour integration scheme
- Interpolation of the effect by the bias voltage
- Quick calculation for periodic systems under zero bias

The details of the implementation can be found in Ref. [43]. The usage of the functionalities is explained in the following subsections.

System we consider

In the current implementation of OpenMX, a system shown in Fig. 29(a) is treated by the NEGF method. The system consists of a central region connected with infinite left and right leads, and the two dimensional periodicity spreads over the **bc**-plane. Considering the two dimensional periodicity, the system can be cast into a one-dimensional problem depending on the Bloch wave vector **k** shown in Fig. 29(b). Also, the Green function of the region $C(\equiv L_0|C_0|R_0)$ is self-consistently determined in order to take account of relaxation of electronic structure around the interface between the central region C_0 and the region $L_0(R_0)$. It should be noted that the electronic transport is assumed to be along the **a**-axis in the current implementation. Thus, users have to keep in mind the specification when the geometrical structure is constructed. See also the subsection 'Step 1: The calculations for leads'.

Computational flow

The NEGF calculation is performed by the following three steps:

Step 1 → Step 2 → Step 3

Each step consists of

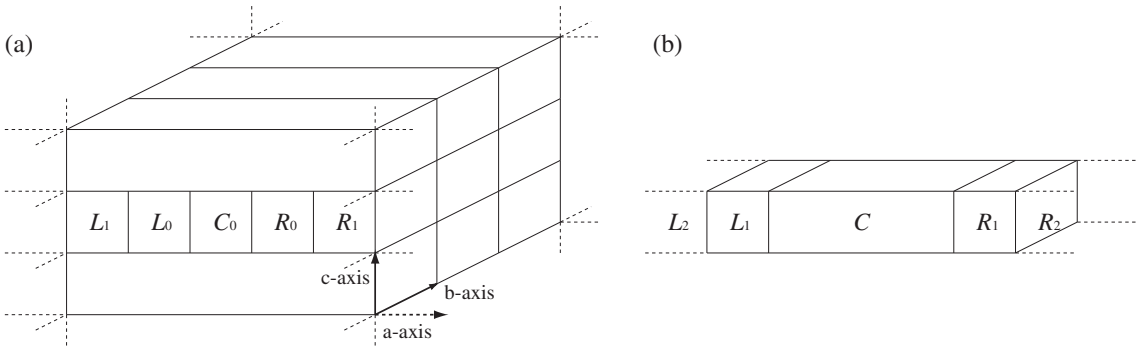


Figure 29: (a) Configuration of the system, treated by the NEGF method, with infinite left and right leads along the **a**-axis under a two dimensional periodic boundary condition on the **bc**-plane. (b) One dimensional system compacted from the configuration of (a) by considering the periodicity on the **bc**-plane, where the region C is an extended central region consisting of C_0, L_0, and R_0.

- **Step 1**

  The band structure calculations are performed for the left and right leads using a program code 'openmx'. The calculated results will be used to represent the Hamiltonian of the leads in the NEGF calculation of the step 2.

- **Step 2**

  The NEGF calculation is performed for the structure shown in Fig. 29 under zero or a finite voltage using a program code 'openmx', where the result in the step 1 is used for the construction of the leads.

- **Step 3**

  By making use of the result of the step 2, the transmission and current are calculated by a program code 'TranMain'.

An example: carbon chain

As a first trial, let us illustrate the three steps by employing a carbon chain. Before going to the illustration, a code 'TranMain' used in the step 3 has to be compiled in the source directory as follows:

```
% make TranMain
```

If the compilation is successful, you will find the executable file 'TranMain', and may copy it your work directory, possibly 'work'. Then, you can proceed the following three calculations:

Step 1

```
%./openmx Lead-Chain.dat | tee lead-chain.std
```

A file 'negf-chain.hks' is generated by the step 1.

Step 2

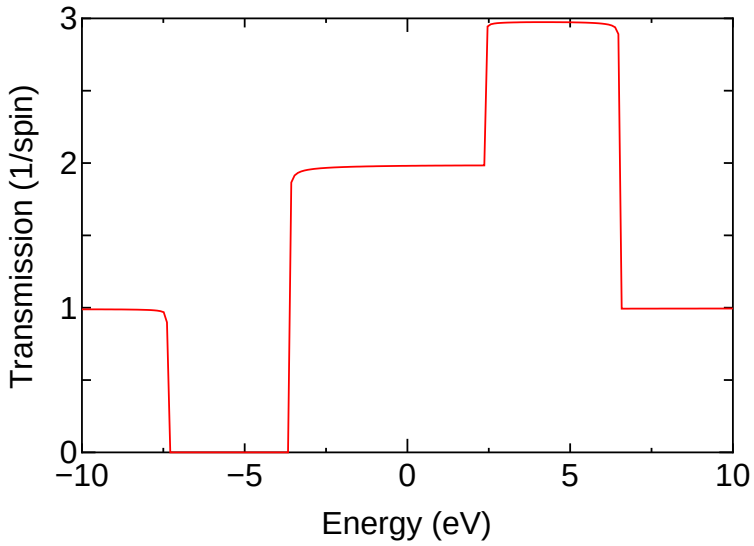


Figure 30: Transmission of a carbon chain as a function of energy. The origin of energy is set to the chemical potential of the left lead.

```
%./openmx NEGF-Chain.dat | tee negf-chain.std
```

A file 'negf-chain.tranb' is generated by the step 2.

Step 3

```
%./TranMain NEGF-Chain.dat
```

'negf-chain.tran0_0', 'negf-chain.current', and 'negf-chain.conductance' are generated by the step 3.

The calculations can be traced by using the input files stored in a directory of 'work/negf_example'. By plotting the sixth column in 'negf-chain.tran0_0' as a function of the fourth column, you can see a transmission curve as shown Fig. 30.

38.2 Step 1: The calculations for leads

The calculation of the step 1 is the conventional band structure calculation to construct information of the lead except for adding the following two keywords 'NEGF.output_hks' and 'NEGF.filename.hks':

```
NEGF.output_hks     on
NEGF.filename.hks   lead-chain.hks
```

The calculated results such as Hamiltonian matrix elements, charge distribution, and difference Hartree potential are stored in a file specified by the keyword 'NEGF.filename.hks'. In this case, a file 'lead-chain.hks' is generated. The '*.hks' file is used in the calculation of the step 2. Since the electronic transport is assumed to be along the **a**-axis in the current implementation, you have to set the **a**-axis for the direction of electronic transport in the band structure calculation. However, you do not need rotate your structure. All you have to do is to change the specification of the lattice vectors. For example, if you want to specify a vector (0.0, 0.0, 10.0) as the **a**-axis in the following lattice vectors:

```
<Atoms.UnitVectors
  3.0  0.0  0.0
  0.0  3.0  0.0
  0.0  0.0 10.0
Atoms.UnitVectors>
```

you only have to specify as follows:

```
<Atoms.UnitVectors
  0.0  0.0 10.0
  3.0  0.0  0.0
  0.0  3.0  0.0
Atoms.UnitVectors>
```

Then, the direction of (0.0, 0.0, 10.0) becomes the direction of electronic transport. As shown above the example, when you change the order of the lattice vectors, please make sure that the keyword 'scf.Kgrid' has to be changed as well.

In the calculation of the step 2, the semi-infiniteness of the leads is taken into account by using the surface Green function which allows us to treat the semi-infiniteness without introducing any discretization. Thus, it would be better to use a large number of **k**-points along the **a**-axis to keep the consistency between the steps 1 and 2 with respect to treatment of the semi-infiniteness of the **a**-axis. Also it is noted that the number of **k**-points for the **bc**-plane should be consistent in the steps 1 and 2.

38.3 Step 2: The NEGF calculation

A. Setting up Lead|Device|Lead

You can set up the regions L_0, C_0, and R_0 in the structural configuration shown in Fig. 29 in the following way:

The geometrical structure of the central region C_0 is specified by the following keywords 'Atoms.Number' and 'Atoms.SpeciesAndCoordinates':

```
Atoms.Number        18
<Atoms.SpeciesAndCoordinates
  1  C  3.000  0.000  0.000  2.0 2.0
   .....
 18  C 28.500  0.000  0.000  2.0 2.0
Atoms.SpeciesAndCoordinates>
```

The geometrical structure of the left lead region L_0 is specified by the following keywords 'LeftLeadAtoms.Number' and 'LeftLeadAtoms.SpeciesAndCoordinates':

```
LeftLeadAtoms.Number         2
<LeftLeadAtoms.SpeciesAndCoordinates
  1  C  0.000  0.000  0.000  2.0 2.0
  2  C  1.500  0.000  0.000  2.0 2.0
LeftLeadAtoms.SpeciesAndCoordinates>
```

The geometrical structure of the right lead region R_0 is specified by the following keywords 'RightLeadAtoms.Number' and 'RightLeadAtoms.SpeciesAndCoordinates'

```
RightLeadAtoms.Number        2
<RightLeadAtoms.SpeciesAndCoordinates
  1  C 30.000  0.000  0.000  2.0 2.0
  2  C 31.500  0.000  0.000  2.0 2.0
RightLeadAtoms.SpeciesAndCoordinates>
```

This is the case of carbon chain which is demonstrated in the previous subsection. The central region C_0 is formed by 18 carbon atoms, and the left and right regions L_0 and R_0 contains two carbon atoms, respectively, where every bond length is 1.5 Å. Following the geometrical specification of device and leads, OpenMX will construct an extended central region $C(\equiv L_0|C_0|R_0)$ as shown in Fig. 29. The Green function for the extended central region C is self-consistently determined in order to take account of relaxation of electronic structure around the interface between the central region C_0 and the region $L_0(R_0)$. In addition, we impose two conditions so that the central Green function can be calculated in the NEGF method [43]:

1. The localized basis orbitals ϕ in the region C_0 overlap with those in the regions L_0 and R_0, but do not overlap with those in the regions L_1 and R_1.

2. The localized basis orbitals ϕ in the L_i (R_i) region has no overlap with basis orbitals in the cells beyond the nearest neighboring cells L_{i-1} (R_{i-1}) and L_{i+1} (R_{i+1}).

In our implementation the basis functions are strictly localized in real space because of the generation of basis orbitals by a confinement scheme [23, 24]. Therefore, once the localized basis orbitals with specific cutoff radii are chosen for each region, the two conditions can be always satisfied by just adjusting the size of the unit cells for L_i and R_i.

Although the specification of unit cells for the regions L_0, C_0, and R_0 is not required, it should be noted that some periodicity is implicitly assumed. The construction of infinite leads is made by employing the unit cells used in the band structure calculations by the step 1, and the informations are stored in a file '*.hks'. Also, due to the structural configuration shown in Fig. 29, the unit vectors on the **bc**-plane for the left and right leads should be consistent. Thus, the unit vector on the **bc**-plane for the extended central region C is implicitly assumed to be same as that of the leads. Within the structural limitation, you can set up the structural configuration.

The unit in the specification of the geometrical structure can be given by

```
Atoms.SpeciesAndCoordinates.Unit   Ang # Ang|AU
```

In the NEGF calculation, either 'Ang' or 'AU' for 'Atoms.SpeciesAndCoordinates.Unit' is supported, but 'FRAC' is not.

How OpenMX analyzes the geometrical structure can be confirmed by the standard output as shown below:

```
<TRAN_Calc_GridBound>

*******************************************************
```

```
The extended cell consists of Left0-Center-Right0.
The cells of left and right reads are connected as.
...|Left2|Left1|Left0-Center-Right0|Right1|Right2...

Each atom in the extended cell is assigned as follows:
where '12' and '2' mean that they are in 'Left0', and
'12' has overlap with atoms in the Left1,
and '13' and '3' mean that they are in 'Right0', and
'13' has overlap with atoms in the 'Right1', and also
'1' means atom in the 'Center'.
*******************************************************

Atom1  = 12 Atom2  =  2 Atom3  =  1 Atom4  =  1 Atom5  =  1 Atom6  =  1 Atom7  =  1
Atom8  =  1 Atom9  =  1 Atom10 =  1 Atom11 =  1 Atom12 =  1 Atom13 =  1 Atom14 =  1
Atom15 =  1 Atom16 =  1 Atom17 =  1 Atom18 =  1 Atom19 =  1 Atom20 =  1 Atom21 =  3
Atom22 = 13
```

The atoms in the extended cell consisting of $L_0|C_0|R_0$ are assigned by the numbers, where '12' and '2' mean that they are in L_0, and '12' has overlap with atoms in L_1, and '13' and '3' mean that they are in R_0, and '13' has overlap with atoms in R_1, and also '1' means atom in C_0. By checking the analysis you may confirm whether the structure is properly constructed or not.

B. Keywords

The NEGF calculation of the step 2 is performed by the keyword 'scf.EigenvalueSolver'

```
scf.EigenvalueSolver       NEGF
```

For the NEGF calculation the following keywords are newly added.

```
NEGF.filename.hks.l     lead-chain.hks
NEGF.filename.hks.r     lead-chain.hks

NEGF.Num.Poles             100       # defalut=150
NEGF.scf.Kgrid             1 1       # defalut=1 1

NEGF.bias.voltage          0.0       # default=0.0 (eV)
NEGF.bias.neq.im.energy    0.01      # default=0.01 (eV)
NEGF.bias.neq.energy.step  0.02      # default=0.02 (eV)
```

An explanation for each keyword is given below.

```
NEGF.filename.hks.l     lead-chain.hks
NEGF.filename.hks.r     lead-chain.hks
```

The files containing information of leads are specified by the above two keywords, where 'NEGF.filename.hks.l' and 'NEGF.filename.hks.r' are for the left and right leads, respectively.

```
NEGF.Num.Poles              100       # defalut=150
```

The equilibrium density matrix is evaluated by a contour integration method [43, 44]. The number of poles used in the method is specified by the keyword 'NEGF.Num.Poles'.

```
NEGF.scf.Kgrid              1 1       # defalut=1 1
```

The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.scf.Kgrid'. Since no periodicity is assumed along **a**-axis, you do not need to specify that for the **a**-axis.

```
NEGF.bias.voltage           0.0       # default=0.0 (eV)
```

The source-drain bias voltage applied to the left and right leads is specified by the keyword 'NEGF.bias.voltage' in units of eV, corresponding to Volt. Noting that only the difference between applied bias voltages has meaning, you only have to give a single value as the source-drain bias voltage.

```
NEGF.bias.neq.im.energy     0.01      # default=0.01 (eV)
NEGF.bias.neq.energy.step   0.02      # default=0.02 (eV)
```

When a finite source-drain bias voltage is applied, a part of the density matrix is contributed by the non-equilibrium Green function. Since the non-equilibrium Green function is not analytic in general in the complex plane, the contour integration method used for the equilibrium Green function cannot be applied. Thus, in the current implementation the non-equilibrium Green function is evaluated on the real axis with a small imaginary part using a simple rectangular quadrature scheme. Then, the imaginary part is given by the keyword 'NEGF.bias.neq.im.energy' and the step width is given by the keyword 'NEGF.bias.neq.energy.step' in units of eV. In most cases, the default values are sufficient, while the detailed analysis of the convergence property can be found in Ref. [43]. How many energy points on the real axis are used for the evaluation of the non-equilibrium Green function can be confirmed in the standard output and the file '*.out'. In case of 'NEGF-Chain.dat', if the bias voltage of 0.5 V is applied, you will see in the standard output that the energy points of 120 are allocated for the calculation as follows:

```
Intrinsic chemical potential (eV) of the leads
  Left lead:  -3.940690039841
  Right lead: -3.940690039841
  add voltage =  0.0000 (eV) to the  left lead: new ChemP (eV):  -3.9407
  add voltage =  0.5000 (eV) to the right lead: new ChemP (eV):  -3.4407

Parameters for the integration of the non-equilibrium part
  lower bound:           -4.894690039841 (eV)
  upper bound:           -2.486690039841 (eV)
  energy step:            0.020000000000 (eV)
  number of steps:           120
```

The total number of energy points where the Green function is evaluated is given by the sum of the number of poles and the number of energy points on the real axis determined by the two keywords 'NEGF.bias.neq.im.energy' and 'NEGF.bias.neq.energy.step', and you should notice that the computational time is proportional to the total number of energy points.

C. SCF criterion

In the NEGF method, the SCF criterion given by the keyword 'scf.criterion' is applied to the residual norm between the input and output charge densities 'NormRD', while in the other cases 'dUele' is monitored.

D. Gate bias voltage

In our implementation, the gate voltage $V_{\rm g}(x)$ is treated by adding an electric potential defined by

$$V_{\rm g}(x) = V_{\rm g}^{(0)} \exp\left[-\left(\frac{x - x_{\rm c}}{d}\right)^8\right],$$

where $V_{\rm g}^{(0)}$ is a constant value corresponding to the gate voltage, and is specified by the keyword 'NEGF.gate.voltage' as follows:

```
NEGF.gate.voltage   1.0    # default=0.0 (in eV)
```

x_c the center of the region C_0, and d the length of the unit vector along **a**-axis for the region C_0. Due to the form of the equation, the applied gate voltage affects mainly the region C_0 in the central region C. The electric potential may resemble the potential produced by the image charges [45].

E. Density of States (DOS)

In the NEGF calculation, the density of states can be calculated by setting the following keywords:

```
Dos.fileout                 on              # on|off, default=off
NEGF.Dos.energyrange     -15.0 25.0 5.0e-3  #default=-10.0 10.0 5.0e-3 (eV)
NEGF.Dos.energy.div        200              # default=200
NEGF.Dos.Kgrid             1 1              # default=1 1
```

When you want to calculate DOS, the keyword 'Dos.fileout' should be set 'on' as usual. Also, the energy range where DOS is calculated is given by the keyword 'NEGF.Dos.energyrange', where the first and second numbers correspond to the lower and upper bounds, and the third number is an imaginary number used for smearing out DOS. The energy range specified by 'NEGF.Dos.energyrange' is divided by the number specified by the keyword 'NEGF.Dos.energy.div'. The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.Dos.Kgrid'. The set of numbers given by 'NEGF.Dos.Kgrid' tends to be larger than that by 'NEGF.scf.Kgrid' because of computational efficiency. After the NEGF calculation with these parameters, you will find two files '*.Dos.val' and '*.Dos.vec', and can analyze those by the same procedure as usual. Also, it should be noted that the origin of energy is set to the chemical potential of the **left** lead.

38.4 Step 3: The transmission and current

After the calculations of the steps 2 and 3, you can proceed calculations of transmission and current by adding the following keywords to the input file used in the calculation of the step 2:

```
NEGF.tran.energyrange -10 10 1.0e-3  # default=-10.0 10.0 1.0e-3 (eV)
NEGF.tran.energydiv        200       # default=200
NEGF.tran.Kgrid            1 1       # default= 1 1
```

The energy range where the transmission is calculated is given by the keyword 'NEGF.tran.energyrange', where the first and second numbers correspond to the lower and upper bounds, and the third number is an imaginary number used for smearing out the transmission. The energy range specified by 'NEGF.tran.energyrange' is divided by the number specified by the keyword 'NEGF.tran.energydiv'. The numbers of **k**-points to discretize the reciprocal vectors $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$ are specified by the keyword 'NEGF.tran.Kgrid'. The set of numbers given by 'NEGF.tran.Kgrid' can be different and tends to be larger than that by 'NEGF.scf.Kgrid' because of computational efficiency.

The calculations of the transmission and current are performed by a program code 'TranMain', which can be compiled in the source directory as follows:

```
% make TranMain
```

If the compilation is successful, you will find the executable file 'TranMain', and may copy it your work directory, possibly 'work'. Using the code 'TranMain' you can perform the calculation of the step 3, for example, as follows:

```
%./TranMain NEGF-Chain.dat

*******************************************************
*******************************************************
 Welcome to TranMain
 This is a post-processing code of OpenMX to calculate
 electronic transmission and current.
 Copyright (C), 2002-2008, H.Kino and T.Ozaki
 TranMain comes with ABSOLUTELY NO WARRANTY.
 This is free software, and you are welcome to
 redistribute it under the constitution of the GNU-GPL.
*******************************************************
*******************************************************


Chemical potentials used in the SCF calculation
  Left lead:  -3.940690039841 (eV)
  Right lead: -3.940690039841 (eV)
NEGF.current.energy.step 1.0000e-02 seems to be large for the calculation of current in
The recommended Tran.current.energy.step is 0.0000e+00 (eV).

Parameters for the calculation of the current
```

```
  lower bound:    -3.940690039841 (eV)
  upper bound:    -3.940690039841 (eV)
  energy step:     0.010000000000 (eV)
  imginary energy  0.001000000000 (eV)
  number of steps:   0

  calculating...

  myid0= 0 i2= 0 i3= 0  k2=  0.0000 k3= -0.0000

Transmission:  files

  ./negf-chain.tran0_0

Current:  file

  ./negf-chain.current


Conductance:  file

  ./negf-chain.conductance
```

After the calculation, in this case you will obtain three files 'negf-chain.tran0_0', 'negf-chain.current', and 'negf-chain.conductance':

- *.tran#_%

  The file stores transmissions for up- and down-spin states. The fourth column is the energy relative to the chemical potential of the **left** lead, and the sixth and eighth columns are transmission for up- and down-spin states, respectively. When you employ a lot of **k**-points which is given by 'NEGF.tran.Kgrid', a file with a different set of '#' and '%' in the file extension is generated for each **k**-point. The correspondence between the numbers and the **k**-points can be found in the file.

- *.current

  The file stores **k**-resolved currents and its average for up- and down-spin states in units of ampere.

- *.conductance

  The file stores **k**-resolved conductance at 0K and its average for up- and down-spin states in units of quantum conductance ($G_0 \equiv \frac{e^2}{h}$). Thus, the conductance G is proportional to the transmission T at the chemical potential of the **left** lead, μ_L, as follows:

$$G = \frac{e^2}{h} T(\mu_L)$$

As an example, the **k**-resolved transmission drawn by using the file '*.conductance' is shown in Fig. 31.

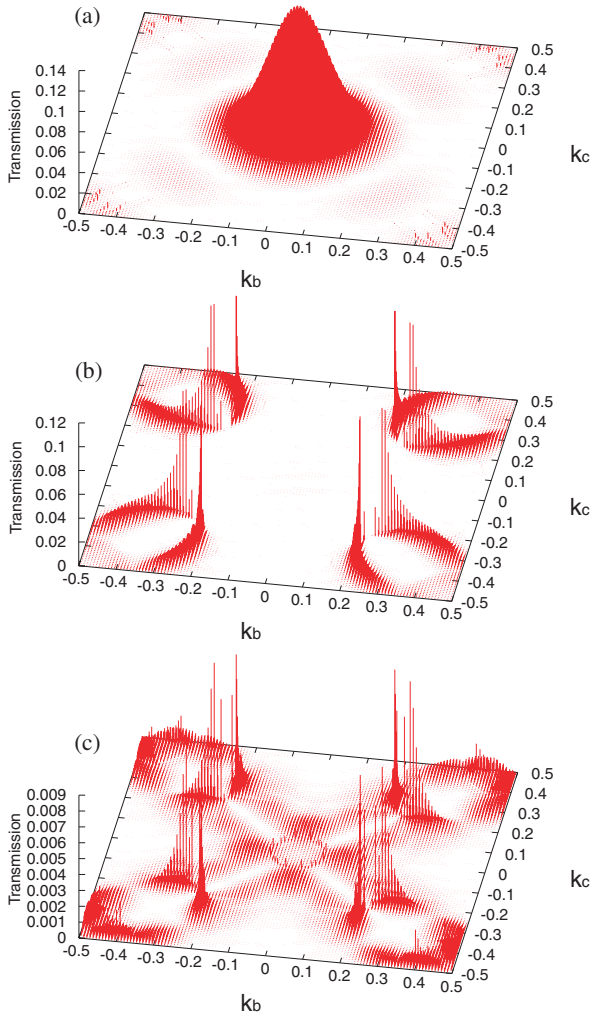


Figure 31: **k**-resolved Transmission at the chemical potential for (a) the majority spin state of the parallel configuration, (b) the minority spin state of the parallel configuration, and (c) a spin state of the antiparallel configuration of Fe|MgO|Fe, respectively. For the calculations **k**-points of 120×120 were used.

38.5 Periodic system under zero bias

When the transmission of a system with the periodicity along the **a**-axis as well as the periodicity of the **bc**-plane is evaluated under zero bias voltage, it can be easily obtained by making use of the Hamiltonian calculated by the conventional band structure calculation without employing the Green function method. This scheme enables us to explore transport properties for a wide variety of possible geometric and magnetic structures with a low computational cost, and thereby can be very useful for many materials such as supperlattice structures. The calculation is performed by adding a keyword 'NEGF.Output.for.TranMain':

```
NEGF.Output.for.TranMain    on
```

in the band structure calculation of the step 1. Then, after the calculation of the step 1, you will obtain a file '*.tranb' which can be used in the calculation of the step 3, which means that you can skip the calculation of the step 2.

38.6 Interpolation of the effect by the bias voltage

Since for large-scale systems it is very time-consuming to perform the SCF calculation at each bias voltage, an interpolation scheme is available to reduce the computational cost in the calculations by the NEGF method. The interpolation scheme is performed in the following way: (i) the SCF calculations are performed for a few bias voltages which are selected in the regime of the bias voltage of interest. (ii) when the transmission and current are calculated, a linear interpolation is made for the Hamiltonian block elements, $H^{(\mathbf{k})}_{\sigma,C}$ and $H^{(\mathbf{k})}_{\sigma,R}$, of the central scattering region and the right lead, and the chemical potential, μ_R, of the right lead by

$$\begin{aligned} H^{(\mathbf{k})}_{\sigma,C} &= \lambda H^{(\mathbf{k},1)}_{\sigma,C} + (1-\lambda) H^{(\mathbf{k},2)}_{\sigma,C}, \\ H^{(\mathbf{k})}_{\sigma,R} &= \lambda H^{(\mathbf{k},1)}_{\sigma,R} + (1-\lambda) H^{(\mathbf{k},2)}_{\sigma,R}, \\ \mu_R &= \lambda \mu^{(1)}_R + (1-\lambda)\mu^{(2)}_R, \end{aligned}$$

where the indices 1 and 2 in the superscript mean that the quantities are calculated or used at the corresponding bias voltages where the SCF calculations are performed beforehand. In general, λ should range from 0 to 1 for the moderate interpolation.

In the calculation of the step 3, the interpolation is made by adding the following keywords in the input file:

```
NEGF.tran.interpolate          on                # default=off, on|off
NEGF.tran.interpolate.file1   c1-negf-0.5.tranb
NEGF.tran.interpolate.file2   c1-negf-1.0.tranb
NEGF.tran.interpolate.coes     0.7 0.3           # default=1.0 0.0
```

When you perform the interpolation, the keyword 'NEGF.tran.interpolate' should be 'on'. In this case, files 'c1-negf-0.5.tranb' and 'c1-negf-1.0.tranb' specified by the keywords 'NEGF.tran.interpolate.file1' and 'NEGF.tran.interpolate.file2' are the results under bias voltages of 0.5 and 1.0 V, respectively, and the transmission and current at $V = 0.7 * 0.5 + 0.3 * 1.0 = 0.65[V]$ are evaluated by the interpolation scheme, where the weights of 0.7 and 0.3 are specified by the keyword 'NEGF.tran.interpolate.coes'.

A comparison between the fully self consistent and the interpolated results is shown with respect to the current and transmission in the linear carbon chain in Figs. 32(a) and (b). In this case, the SCF calculations at three bias voltages of 0, 0.5, and 1.0 V are performed, and the results at the other bias voltages are obtained by the interpolation scheme. For comparison we also calculate the currents via the SCF calculations at all the bias voltages. It is confirmed that the simple interpolation scheme gives notably accurate results for both the calculations of the current and transmission. Although the proper selection of bias voltages used for the SCF calculations may depend on systems, the result suggests that the simple scheme is very useful to interpolate the effect of the bias voltage while keeping the accuracy of the calculations.

38.7 Parallelization of NEGF

In the current implementation the NEGF calculation is parallelized by MPI. In addition to the MPI parallelization, if you use ACML or MKL, the matrix multiplication and the inverse calculation of matrix in the evaluation of the Green function are also parallelized by OpenMP. In this case, you can perform a hybrid parallelization by OpenMP/MPI which may lead to shorter computational time. The way for the parallelization is completely same as before.

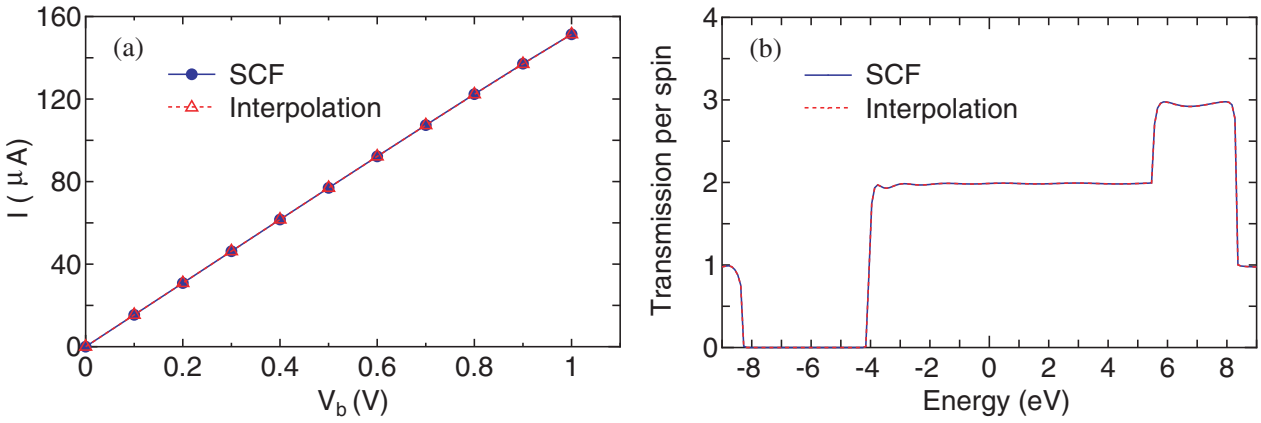


Figure 32: (a) Currents of the linear carbon chain calculated by the SCF calculations (solid line) and the interpolation scheme (dotted line). (b) Transmission of the linear carbon chain under a bias voltage of 0.3 V, calculated by the SCF calculations (solid line) and the interpolation scheme (dotted line). The imaginary part of 0.01 and the grid spacing of 0.01 eV are used for the integration of the nonequilibrium term in the density matrix.

In Fig. 33 we show the speed-up ratio in the elapsed time for the evaluation of the density matrix of 8-zigzag graphene nanoribbon(ZGNR) under a finite bias voltage of 0.5 eV. The energy points of 197 (101 and 96 for the equilibrium and nonequilibrium terms, respectively) are used for the evaluation of the density matrix. Only the Γ point is employed for the **k**-point sampling, and the spin polarized calculation is performed. Thus, the combination of 394 for the three indices are parallelized by MPI. It is found that the speed-up ratio of the flat MPI parallelization, corresponding to 1 thread, reasonably scales up to 64 processes. Furthermore, it can be seen that the hybrid parallelization, corresponding to 2 and 4 threads, largely improves the speed-up ratio. By fully using 64 quad core processors, corresponding to 64 processes and 4 threads, the speed-up ratio is about 140, demonstrating the good scalability of the NEGF method. For the details see also Ref.[43].

38.8 Examples

For user's convenience, input files for four examples can be found in 'work/negf_example' as follows:

- Carbon chain under zero bias voltage

  Step 1: Lead-Chain.dat
  Step 2: NEGF-Chain.dat

- Graphene sheet under zero bias voltage

  Step 1: Lead-Graphene.dat
  Step 2: NEGF-Graphene.dat

- 8-zigzag graphene nanoribbon with an antiferromagnetic junction under a finite bias voltage of 0.3 V

  Step 1: Lead-L-8ZGNR.dat, Lead-R-8ZGNR.dat
  Step 2: NEGF-8ZGNR-0.3.dat

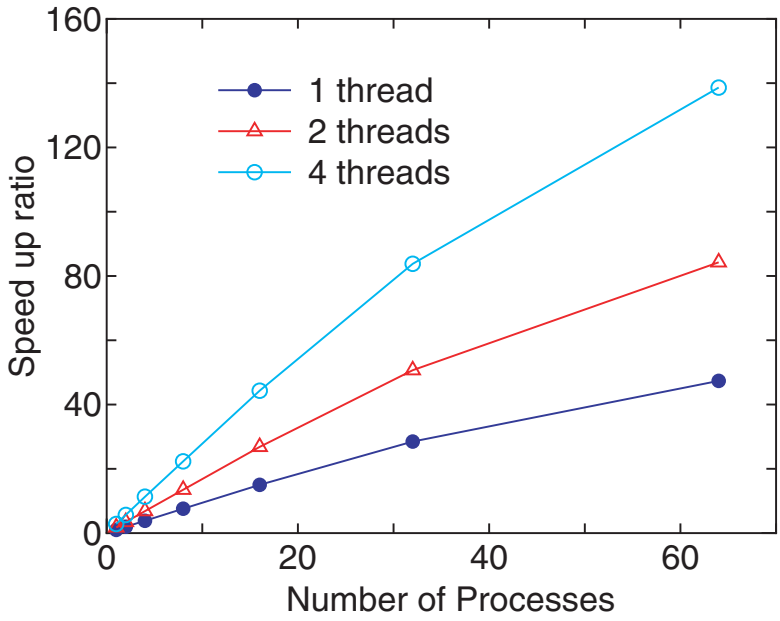


Figure 33: Speed-up ratio in the parallel computation of the calculation of the density matrix for the FM junction of 8-zigzag graphene nanoribbon (ZGNR) by a hybrid scheme using MPI and OpenMP. The speed-up ratio is defined by T_1/T_p, where T_1 and T_p are the elapsed times by a single core and a parallel calculations. The cores used in the MPI and OpenMP parallelizations are called *process* and *thread*, respectively. The parallel calculations were performed on a Cray XT5 machine consisting of AMD opteron quad core processors (2.3GHz). In the benchmark calculations, the number of processes is taken to be equivalent to that of processors. Therefore, in the parallelization using 1 or 2 threads, 3 or 2 cores are idle in a quad core processor.

- Fe|MgO|Fe tunneling junction with an antiferromagnetic configuration between two iron leads under zero bias voltage

  Step 1: Lead-L-Fe.dat, Lead-R-Fe.dat
  Step 2: NEGF-AP-Fe-MgO-Fe.dat

38.9 Automatic running test of NEGF

To check whether the NEGF calculation part is properly installed or not, an automatic running test for the NEGF calculation can be performed by

For the serial running

```
% ./openmx -runtestNEGF
```

For the MPI parallel running

```
% mpirun -np 16 openmx -runtestNEGF
```

For the OpenMP/MPI parallel running

```
% mpirun -np 8 openmx -runtestNEGF -nt 2
```

Then, OpenMX will run with four test cases including calculations of the steps 1 and 2, and compare calculated results with the reference results which are stored in 'work/negf_example'. The comparison (absolute difference in the total energy and force) is stored in a file 'runtestNEGF.result' in the directory 'work'. The reference results were calculated using 40 MPI processes of a 2.6GHz Opteron cluster machine. If the difference is within last seven digits, we may consider that the installation is successful.

39 Maximally Localized Wannier Function

39.1 General

The following are descriptions on how to use OpenMX to generate maximally localized Wannier function (MLWF) [74, 75]. Keywords and settings for controlling the calculations are explained. The style of key words are closely following those originally in OpenMX. Throughout the section, a couple of results for silicon in the diamond structure will be shown for convenience. The calculation can be traced by openmx code with an input file 'Si.dat' in 'openmx*.*/work/wf_example'. There is no additional post processing code. After users may get the convergent result for the conventional SCF process for the electronic structure calculation, the following procedure explained below will be repeated by changing a couple of parameters with the restart file until desired MLWFs are obtained.

To acknowledge in any publications by using the functionality, the citation of the reference [46] would be appreciated:

Switching on generating MLWFs

To switch on the calculation, keyword 'Wannier.Func.Calc' should be explicitly set as 'on'. Its default value is 'off'.

```
Wannier.Func.Calc        on         #default off
```

Setting the number of target MLWFs

The number of target MLWFs should be given explicitly by setting a keyword 'Wannier.Func.Num' and no default value for it.

```
Wannier.Func.Num       4          #no default
```

Energy window for selecting Bloch states

The MLWFs will be generated from a set of Bloch states, which are selected by defining an energy window covering the eigen energies of them. Following Ref. [75], two energy windows are introduced. One is so-called outer window, defined by two keywords, 'Wannier.Outer.Window.Bottom' and 'Wannier.Outer.Window.Top', indicating the lower and upper boundaries, respectively. The other one is inner window, which is specified by two similar key words, 'Wannier.Inner.Window.Bottom' and 'Wannier.Inner.Window.Top'. All these four values are given in units eV relative to Fermi level. The inner window should be fully inside of the outer window. If the two boundaries of inner window are equal to each other, it means inner window is not defined and not used in calculation. There is no default values for outer window, while 0.0 is the default value for two boundaries of inner window. One example is as following:

```
Wannier.Outer.Window.Bottom   -14.0   #lower boundary of outer window, no default value
Wannier.Outer.Window.Top        0.0   #upper boundary of outer window, no default value
Wannier.Inner.Window.Bottom     0.0   #lower boundary of inner window, default value 0.0
Wannier.Inner.Window.Top        0.0   #upper boundary of outer window, default value 0.0
```

To set these two windows covering interested bands, it is usually to plot band structure and/or density of states before the calculation of MLWFs. If you want to restart the minimization of MLWFs by reading the overlap matrix elements from files, the outer window should not be larger than that used for calculating the stored overlap matrix. Either equal or smaller is allowed. The inner window can be varied within the outer window as you like when the restart calculation is performed. This would benefit the restarting calculation or checking the dependence of MLWFs on the size of both the windows. For the restarting calculation, please see also the section (7) 'Restart optimization without calculating overlap matrix'.

Initial guess of MLWFs

User can choose whether to use initial guess of target MLWFs or not by setting the keyword 'Wannier.Initial.Guess' as 'on' or 'off'. Default value is 'on', which means we recommend user to use an initial guess to improve the convergence or avoid local minima during the minimization of spread function.

If the initial guess is required, a set of local functions with the same number of target MLWFs should be defined. Bloch wave functions inside the outer window will be projected on to them. Therefore, these local functions are also called as projectors. The following steps are required to specify a projector.

A. Define local functions for projectors

Since the pseudo-atomic orbitals are used for projectors, the specification of them is the same as for the basis functions. An example setting, for silicon in diamond structure, is as following:

```
Species.Number          2

<Definition.of.Atomic.Species
  Si       Si5.5-s2p2d1    Si_CA
  proj1    Si5.5-s1p1d1f1  Si_CA
Definition.of.Atomic.Species>
```

In this example, since we employ PAOs from Si as projectors, an additional specie 'proj1' is defined as shown above. Inside the pair keywords '<Definition.of.Atomic.Species' and 'Definition.of.Atomic.Species>', in addition to the first line used for Si atoms, one species for the projectors is defined. Its name is 'proj1' defined by 'Si5.5-s1p1d1f1' and the pseudopotential 'Si_CA'. In fact, the pseudopotential defined in this line will not be used. It is given just for keeping the consistence of inputting data structure. One can use any PAO as projector. Also the use of only a single basis set is allowed for each l-component. We strongly recommend user to specify 's1p1d1f1' in all cases to avoid possible error.

B. Specify the orbital, central position and orientation of a projector

Pair keywords '<Wannier.Initial.Projectos' and 'Wannier.Initial.Projectos>' will be used to specify the projector name, local orbital function, center of local orbital, and the local z-axis and x-axis for orbital orientation.

An example setting is shown here:

```
<Wannier.Initial.Projectors
   proj1-sp3   0.250  0.250  0.250   -1.0 0.0 0.0    0.0  0.0 -1.0
   proj1-sp3   0.000  0.000  0.000    0.0 0.0 1.0    1.0  0.0  0.0
Wannier.Initial.Projectors>
```

Each line contains the following items. For example, in the first line, the species name, 'proj1', is defined in pairing keywords 'Definition.of.Atomic.Species'. '-' is used to connect the projector name and the selected orbitals. 'sp3' means the sp3 hybridized orbitals of this species is used as the initial guess of four target Wannier functions (see also Table 6 for all the possible orbitals and their hybrids). The projectors consisting of hybridized orbitals are centered at the position given by the following 3 numbers, '0.25 0.25 0.25', which are given in unit defined by keyword 'Wannier.Initial.Projectors.Unit' (to be explained below). The next two sets of three numbers define the z-axis and x-axis of the local coordinate system, respectively, where each axis is specified by the vector defined by three components in xyz-coordinate. In this example, in the first line the local z-axis defined by '-1.0 0.0 0.0' points to the opposite direction to the original x-axis, while the local x-axis defined by '0.0 0.0 -1.0' points to the opposite direction to the original z-axis. In the second line the local axes are the same as the original coordinate system.

The orbital used as projector can be the original PAOs or any hybrid of them. One must be aware that the total number of projectors defined by 'sp3' is 4. Similarly, 'sp' and 'sp2' contain 2 and 3 projectors, respectively. A list of supported PAOs and hybridizations among them can be found in Table 6. Any name other than those listed is not allowed.

The projector can be centered anywhere inside the unit cell. To specify its location, we can use the fractional (FRAC) coordinates relative to the unit cell vectors or Cartesian coordinates in atomic unit (AU) or in angstrom (ANG). The corresponding keyword is 'Wannier.Initial.Projectors.Unit'.

```
Wannier.Initial.Projectors.Unit     FRAC     #AU, ANG or FRAC
```

K grid mesh and b vectors connecting neighboring k-points

The Monkhorst-Pack k grid mesh is defined by keyword 'Wannier.Kgrid'. There is no default setting for it. To use finite difference approach for calculating k-space differentials, **b** vectors connecting neighboring k points are searched shell by shell according to the distance from a central k point. The maximum number of searched shells is defined by keyword 'Wannier.MaxShells'. Default value is 12 and it should be increased if failure in finding a set of proper **b** vectors. The problem may happen in case of a system having a large aspect ratio among unit vectors, and in this case you will see an error message, while the value 12 works well in most cases. A proper setting of 'Wannier.Kgrid' will also help to find **b** vectors, where the grid spacing by the discretization for each reciprocal lattice vector should be nearly equivalent to each other.

Table 6: Orbitals and hybrids used as projector. The hybridization is done within the new coordinate system defined by z-axis and x-axis.

Orbital name	Number of included projector	Description
s	1	s orbital from PAOs
p	3	p_x, p_y, p_z from PAOs
px	1	p_x from PAOs
py	1	p_y from PAOs
pz	1	p_z from PAOs
d	5	$d_{z^2}, d_{x^2-y^2}, d_{xy}, d_{xz}, d_{yz}$ from PAOs
dz2	1	d_{z^2} from PAOs
dx2-y2	1	$d_{x^2-y^2}$ from PAOs
dxy	1	d_{xy} from PAOs
dxz	1	d_{xy} from PAOs
dyz	1	d_{xy} from PAOs
f	7	$f_{z^3}, f_{xz^2}, f_{yz^2}, f_{zx^2}, f_{xyz}, f_{x^3-3xy^2}, f_{3yx^2-y^3}$ from PAOs
fz3	1	f_{z^3} from PAOs
fxz2	1	f_{xz^2} from PAOs
fyz2	1	f_{yz^2} from PAOs
fzx2	1	f_{zx^2} from PAOs
fxyz	1	f_{xyz} from PAOs
fx3-3xy2	1	$f_{x^3-3xy^2}$ from PAOs
f3yx2-y3	1	$f_{3yx^2-y^3}$ from PAOs
sp	2	Hybridization between s and px orbitals, including $\frac{1}{\sqrt{2}}(s+p_x)$ and $\frac{1}{\sqrt{2}}(s-p_x)$
sp2	3	Hybridization among s, px, and py orbitals, including $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x-\frac{1}{\sqrt{2}}p_y$ and $\frac{1}{\sqrt{3}}s+\frac{2}{\sqrt{6}}p_x$
sp3	4	Hybridization among s, px, py and pz orbitals: $\frac{1}{\sqrt{2}}(s+p_x+p_y+p_z)$, $\frac{1}{\sqrt{2}}(s+p_x-p_y-p_z)$ $\frac{1}{\sqrt{2}}(s-p_x+p_y-p_z)$, $\frac{1}{\sqrt{2}}(s-p_x-p_y+p_z)$
sp3dz2	5	Hybridization among s, p_x, p_y, p_z and d_{z^2} orbitals: $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{1}{\sqrt{6}}p_x+\frac{1}{\sqrt{2}}p_y$, $\frac{1}{\sqrt{3}}s-\frac{2}{\sqrt{6}}p_x$ $\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{2}}d_{z^2}$, $-\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{2}}d_{z^2}$
sp3deg	6	Hybridization among s, p_x, p_y, p_z and $d_{z^2}, d_{x^2-y^2}$ orbitals: $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_x-\frac{1}{\sqrt{12}}d_{z^2}+\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_x-\frac{1}{\sqrt{12}}d_{z^2}+\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_y-\frac{1}{\sqrt{12}}d_{z^2}-\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_y-\frac{1}{\sqrt{12}}d_{z^2}-\frac{1}{2}d_{x^2-y^2}$, $\frac{1}{\sqrt{6}}s-\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{3}}d_{z^2}$, $\frac{1}{\sqrt{6}}s+\frac{1}{\sqrt{2}}p_z+\frac{1}{\sqrt{3}}d_{z^2}$

```
Wannier.MaxShells          12          # default value is 12.
Wannier.Kgrid           8  8  8        # no default value
```

Minimizing spread of WF

For entangled band case [75], two steps are needed to find the MLWFs. The first step is to minimize the gauge invariant part of spread function by disentangling the non-isolated bands. The second step is the same as isolated band case [74]. The gauge dependent part is optimized by unitary transformation of the selected Bloch wave functions according to the gradient of spread function. For the first step, three parameters are used to control the self-consistence loop. They are 'Wannier.Dis.SCF.Max.Steps', 'Wannier.Dis.Conv.Criterion', and 'Wannier.Dis.Mixing.Para'. They are the maximum number of SCF loops, the convergence criterion, and the parameter to control the mixing of input and output subspace projectors, respectively.

```
Wannier.Dis.SCF.Max.Steps         2000         # default 200
Wannier.Dis.Conv.Criterion        1e-12        # default 1e-8
Wannier.Dis.Mixing.Para           0.5          # default value is 0.5
```

For the second step, three minimization methods are available. One is a steepest decent (SD) method, and the second one is a conjugate gradient (CG) method. The third one is a hybrid method which uses the SD method firstly and then switches to the CG method. The keyword 'Wannier.Minimizing.Scheme' indicates which method to be used. '0', '1', and '2' mean the simple SD method, the CG method, and hybrid method, respectively. The step length for the SD method is set by the keyword 'Wannier.Minimizing.StepLength'. In the CG method, a secant method is used to determine the optimized step length. The maximum secant steps and initial step length is specified by 'Wannier.Minimizing.Secant.Steps' and 'Wannier.Minimizing.Secant.StepLength', respectively. The maximum number of minimization step and convergence criterion are controlled by 'Wannier.Minimizing.Max.Steps' and 'Wannier.Minimizing.Conv.Criterion', respectively.

```
Wannier.Minimizing.Scheme             2     # default 0, 0=SD 1=CG 2=hybrid
Wannier.Minimizing.StepLength        2.0    # default 2.0
Wannier.Minimizing.Secant.Steps       5     # default 5
Wannier.Minimizing.Secant.StepLength 2.0    # default 2.0
Wannier.Minimizing.Conv.Criterion   1e-12   # default 1e-8
Wannier.Minimizing.Max.Steps         200    # default 200
```

In the hybrid minimization scheme, SD and CG have the same number of maximum minimization steps as specified by 'Wannier.Minimizing.Max.Steps'.

Restarting optimization without calculating overlap matrix

If the overlap matrix $M_{mn}^{(\mathbf{k},\mathbf{b})}$ has been calculated and stored in a disk file, the keyword 'Wannier.Readin.Overlap.Matrix' can be set as 'on' to restart generating MLWF without calculating $M_{mn}^{(\mathbf{k},\mathbf{b})}$ again.

```
Wannier.Readin.Overlap.Matrix       off      # default is on
```

This can save the computational time since the calculation of overlap matrix is time consuming. The code will read the overlap matrix as well as the eigenenergies and states from the disk file. One should keep in mind that the outer window and k grid should be the same as those used for calculating the stored overlap matrix and eigenvalues. Consistence will be checked in the code. The inner window, initial guess of MLWF as well as the convergence criteria can be adjusted for restarting optimization. If 'Wannier.Readin.Overlap.Matrix' is set as 'off', the overlap matrix will be calculated and automatically stored into a disk file. The file name is defined by 'System.Name' with extension '.mmn'. The eigenenergies and states are also stored in the disk file with extension '.eigen'.

39.2 Analysis

Plotting interpolated band structure

To plot the interpolated band structure, set 'Wannier.Interpolated.Bands' to be 'on'.

```
Wannier.Interpolated.Bands              on    # on|off, default=off
```

Other necessary settings, like k-path and sampling density along each path, are borrowed from those for plotting band dispersion in OpenMX. Therefore, the keyword 'Band.dispersion' should be set as 'on' in order to draw interpolated band structure. After convergence, interpolated band dispersion data will be found in a file with the extension name '.Wannier_Band', which has the same format as '.Band' file. As an example, the interpolated band structure of Si in diamond structure is shown together with its original band structure in Fig. 34(a).

Plotting MLWF

To plot the converged MLWFs, change the keyword 'Wannier.Function.Plot' to be 'on'. The default value of it is 'off'.

```
Wannier.Function.Plot                   on          # default off
Wannier.Function.Plot.SuperCells      1 1 1         # default=0 0 0
```

If it is turned on, all the MLWFs will be plotted. They are written in Gaussian Cube file format with the extension file name like '.mlwf1_4_r.cube'. The file is named in the same style as HOMO or LUMO molecular orbitals files. The first number after '.mlwf' indicates the spin index and the following one are index of MLWFs and the last letter 'r' or 'i' means the real or imaginary part of the MLWF. Users can set the supercell size for plotting MLWF. It is defined by the keyword 'Wannier.Function.Plot.SuperCells'. '1 1 1' in the above example means that the unit cell is extended by one in both the plus and minus directions along the a-, b-, and c-axes by putting the home unit cell at the center, and therefore the MLWFs are plotted in an extended cell consisting of 27 (= (1∗2+1)∗(1∗2+1)∗(1∗2+1)) cells in this case. Figure 34(b) shows one of the eight converged MLWFs from four valence states and four conduction states near Fermi level of Si in diamond structure.

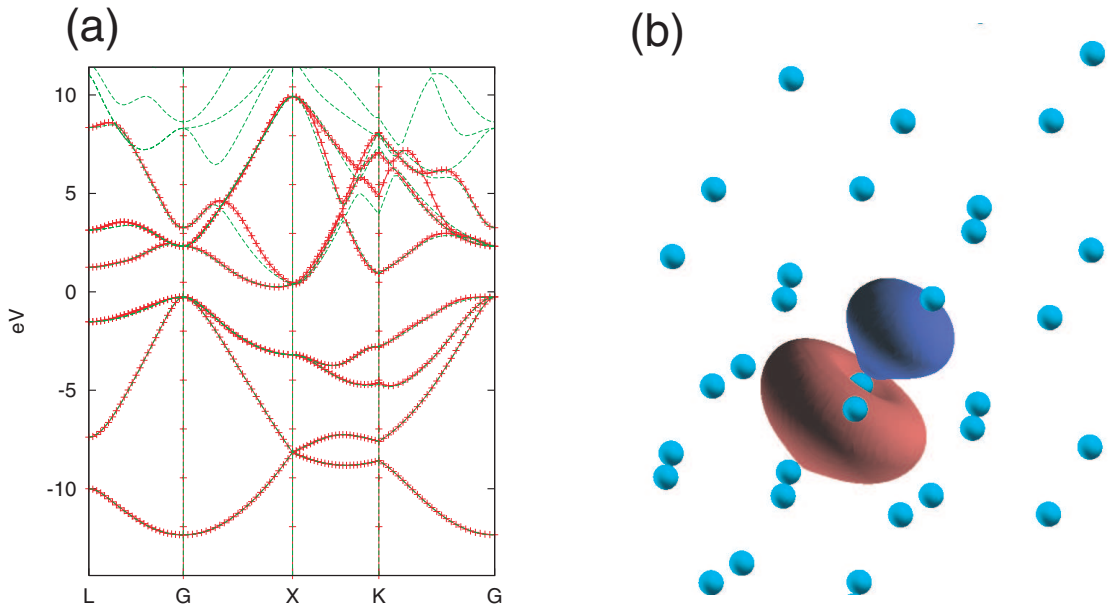


Figure 34: (a) The interpolated band structure (symbolic line) of Si in diamond structure is compared with original band structure (solid line). (b) One of the eight converged MLWFs from four valence states and four conduction states near Fermi level of Si in diamond structure. It is obtained with an initial guess of sp3 hybrid.

39.3 Monitoring Optimization of Spread Function

The output during optimization steps is printed to standard output. To monitor the optimization progress, the following method may be helpful. For convenient, we assume the standard output is stored in a file 'stdout.std'. The following example is for Si.dat which can be found in openmx*.*/work/wf_example, and each user can trace the same calculation.

DISE

Monitor the self-consistent loops for disentangling progress (the first step of optimization):

```
% grep "DISE" stdout.std

|  Iter  | Omega_I (Angs^2) | Delta_I (Angs^2) |  ---> DISE
|     1  |   15.729405734992|   15.729405734992|  ---> DISE
|     2  |   15.472899644675|   -0.256506090316|  ---> DISE
|     3  |   15.298904204761|   -0.173995439915|  ---> DISE
|     4  |   15.184358975818|   -0.114545228943|  ---> DISE
|     5  |   15.109144558995|   -0.075214416823|  ---> DISE
|     6  |   15.059536215864|   -0.049608343132|  ---> DISE
|     7  |   15.026555029591|   -0.032981186273|  ---> DISE
|     8  |   15.004405549290|   -0.022149480301|  ---> DISE
|     9  |   14.989360674543|   -0.015044874747|  ---> DISE
```

```
|    10  |   14.979019019700|   -0.010341654843|  ---> DISE
........
.....
...
.
```

where 'Iter', 'Omega_I', and 'Delta_I' mean the iteration number, the gauge invariant part of the spread function, and its difference between two neighboring steps. The criterion given by the keyword 'Wannier.Dis.Conv.Criterion' is applied to 'Delta_I'.

CONV

Monitor the optimization of the gauge dependent part of the spread function (the second step of optimization):

```
 % grep "CONV" stdout.std

|Opt Step |Modu.of Gradient|d_Omega_in_steps|     d_Omega    | (in Angs^2) ---> CONV
| SD    1 | 1.01158918E+00 | 1.49290922E-03 |-1.49143228E-03|  ---> CONV
| SD    2 | 1.00759578E+00 | 1.48701574E-03 |-1.48555300E-03|  ---> CONV
 .....
 .....
| SD  200 | 5.84949196E-01 | 8.63271441E-04 |-8.62809578E-04|  ---> CONV
|Opt Step |Mode of Gradient|     d_Omega    | (Angs^2) ---> CONV
| CG    1 | 3.60406914E-01 | -4.00536821E-01|  ---> CONV
 .....
 .....
| CG   55 | 1.00046240E-12 | -4.85492314E-13|  ---> CONV
| CG   56 | 3.40196568E-13 | -2.46725602E-13|  ---> CONV
************************************************************ ---> CONV
                  CONVERGENCE ACHIEVED !                     ---> CONV
************************************************************ ---> CONV
                  CONVERGENCE ACHIEVED !                     ---> SPRD
```

where 'Opt Step' and 'Modu.of Gradient' are the optimization step in either 'SD' or 'CG' method and the modulus of gradient of the spread function. The difference between two neighboring steps in the gauge dependent spread functions is calculated in two different way in the SD method, giving 'd_Omega_in_steps' and 'd_Omega'. 'd_Omega_in_steps' is given by

$$d\Omega = \epsilon \sum_{\mathbf{k}} ||G^{(\mathbf{k})}||^2,$$

where ϵ is the step length, $G^{(\mathbf{k})}$ is the gradient of the spread function. The details of the equation can be found in Ref. [74]. On the other hand, 'd_Omega' is given by

$$d\Omega = \Omega^{(n+1)} - \Omega^{(n)},$$

where n is the iteration number. In the CG method, only 'd_Omega' is evaluated. The criterion given by the keyword 'Wannier.Minimizing.Conv.Criterion' is applied to 'Modu.of Gradient'.

SPRD

Monitor the variation of spread of the Wannier functions:

```
% grep "SPRD" stdout.std

|Opt Step |     Omega_I    |     Omega_D    |     Omega_OD  |    Tot_Omega  | (in Angs^2) ---> SPRD
| SD    1 |   14.95330056  |    0.13750102  |    6.49032814 |   21.58112972 |  ---> SPRD
| SD    2 |   14.95330056  |    0.13744541  |    6.48889820 |   21.57964416 |  ---> SPRD
| SD    3 |   14.95330056  |    0.13739052  |    6.48747336 |   21.57816443 |  ---> SPRD
| SD    4 |   14.95330056  |    0.13733636  |    6.48605355 |   21.57669047 |  ---> SPRD
.....
.....
| SD  199 |   14.95330056  |    0.13449510  |    6.27229121 |   21.36008687 |  ---> SPRD
| SD  200 |   14.95330056  |    0.13450210  |    6.27142140 |   21.35922406 |  ---> SPRD
|Opt Step |     Omega_I    |     Omega_D    |     Omega_OD  |    Tot_Omega  | (Angs^2) ---> SPRD
| CG    1 |    14.95330056 |     0.16848639 |    5.83690029 |   20.95868724 |  ---> SPRD
| CG    2 |    14.95330056 |     0.16421934 |    5.78500985 |   20.90252974 |  ---> SPRD
| CG    3 |    14.95330056 |     0.16106859 |    5.77547389 |   20.88984303 |  ---> SPRD
.....
.....
| CG   55 |    14.95330056 |     0.15987820 |    5.77203385 |   20.88521260 |  ---> SPRD
| CG   56 |    14.95330056 |     0.15987820 |    5.77203385 |   20.88521260 |  ---> SPRD
************************************************************ ---> SPRD
                  CONVERGENCE ACHIEVED !                     ---> SPRD
************************************************************ ---> SPRD
```

where 'Opt Step' is the optimization step in either 'SD' or 'CG' method. 'Omega_I' is the gauge invariant part of spread function. 'Omega_D' and 'Omega_OD' are the gauge dependent diagonal and off-diagonal contribution, respectively. 'Tot_Omega' is the sum up of all the above three components of the spread function.

CENT

Monitor the variation of Wannier function center:

```
% grep "CENT" stdout.std
WF   1 ( 1.14465704, 1.14465689, 1.14465697) |   2.69781828  --->CENT
WF   2 ( 1.55414640, 1.55414634, 1.14465803) |   2.69783245  --->CENT
WF   3 ( 1.55414741, 1.14465636, 1.55414731) |   2.69783410  --->CENT
WF   4 ( 1.14465805, 1.55414638, 1.55414610) |   2.69781605  --->CENT
WF   5 ( 0.20474553, 0.20474549, 0.20474559) |   2.69782821  --->CENT
WF   6 ( 0.20474381,-0.20474431,-0.20474417) |   2.69782871  --->CENT
WF   7 (-0.20474538, 0.20474519,-0.20474564) |   2.69783184  --->CENT
WF   8 (-0.20474446,-0.20474477, 0.20474501) |   2.69783151  --->CENT
Total Center ( 5.39760841, 5.39760756, 5.39760921) sum_spread  21.58262115 --->CENT
SD     1 ------------------------------------------------------------------------> CENT
WF   1 ( 1.14466694, 1.14466679, 1.14466688) |   2.69763185  --->CENT
```

```
WF   2 ( 1.55413650, 1.55413644, 1.14466794) |   2.69764602  --->CENT
WF   3 ( 1.55413751, 1.14466626, 1.55413741) |   2.69764767  --->CENT
WF   4 ( 1.14466796, 1.55413647, 1.55413620) |   2.69762962  --->CENT
WF   5 ( 0.20473563, 0.20473559, 0.20473568) |   2.69764178  --->CENT
WF   6 ( 0.20473391,-0.20473440,-0.20473426) |   2.69764228  --->CENT
WF   7 (-0.20473548, 0.20473528,-0.20473574) |   2.69764541  --->CENT
WF   8 (-0.20473455,-0.20473487, 0.20473510) |   2.69764508  --->CENT
Total Center ( 5.39760841, 5.39760756, 5.39760921) sum_spread  21.58112972 --->CENT
SD     2 ------------------------------------------------------------------------> CENT
 .....
 .....
CG    56 ------------------------------------------------------------------------> CENT
WF   1 ( 1.14827796, 1.14827609, 1.14827941) |   2.61064261  --->CENT
WF   2 ( 1.55052871, 1.55052733, 1.14827518) |   2.61064883  --->CENT
WF   3 ( 1.55052569, 1.14827674, 1.55052521) |   2.61066105  --->CENT
WF   4 ( 1.14827660, 1.55052607, 1.55052876) |   2.61063511  --->CENT
WF   5 ( 0.20111752, 0.20112435, 0.20112008) |   2.61067382  --->CENT
WF   6 ( 0.20113239,-0.20112644,-0.20113155) |   2.61063347  --->CENT
WF   7 (-0.20112056, 0.20112674,-0.20111898) |   2.61067162  --->CENT
WF   8 (-0.20112985,-0.20112333, 0.20113122) |   2.61064610  --->CENT
Total Center ( 5.39760846, 5.39760756, 5.39760933) sum_spread  20.88521260 --->CENT
```

where the optimization method and step is indicated by starting with 'SD' or 'CG'. Lines starting with 'WF' show the center of each Wannier function with (x, y, z) coordinates in $\mathring{A}$ unit. and its spread in $\mathring{A}^2$. The sum up of all the Wannier functions center and spread are given in the the line starting with 'Total Center'.

39.4 Examples for generating MLWFs

Examples for different materials are prepared in the installation directory: work/wf_example.

- Benzene.dat

  for generating six p_z-orbital like Wannier functions from benzene's six π molecular orbitals.

- GaAs.dat

  for generating maximally localized Wannier functions from four valence bands of GaAs.

- Si.dat

  for generating eight Wannier functions by including both valence and conduction bands of Si. The initial guess is sp3 hybrids.

- symGra.dat

  for generating the Wannier function for graphene sheet. The initial guess is sp2 hybrids and p_z orbitals on carbon atoms.

- pmSVO.dat

  for generating t_{2g}-like Wannier functions for cubic perovskite $SrVO_3$ without spin polarization calculation.

- NC_SVO.dat

  similar to the case of pmSVO.dat except for the inclusion of spin-orbit coupling.

- GaAs_NC.dat

  similar to the case of GaAs.dat but spin-orbit coupling is included.

- VBz.dat

  for generating Wannier functions for Vanadium-Benzene infinite chain, which is studied in Ref. [46].

39.5 Output files

Additional four files generated by the calculation are explained below. They have different extension names. '.mmn' file is for storing the overlap matrix elements $M_{mn}^{(\mathbf{k},\mathbf{b})}$. '.amn' is for the initial guess projection matrix element $A_{mn}^{(\mathbf{k})}$. '.eigen' is for the eigenenergies and eigenstates at each k point. The '.HWR' file is for the hopping integrals among MLWFs on a set of lattice vectors which lies in the Wigner-Seitz supercells conjugated with the sampled k grids. For restarting optimization calculation, '.mmn' file will be read instead of written. More detailed information of the four files will be given below.

A. File format of '.mmn' file

This file structure is closely following that in Wannier90 [76]. The first line of this file is the description of the numbers in the second line. The numbers from left to right in the second line are the number (N_{win}) of included bands within the outer window, the number of k points, the number of **b** vectors, the number of spin component, respectively. The next lines are data blocks of $M_{mn}^{(\mathbf{k},\mathbf{b})}$. The most outer loop is for spin component. The next is the loop of **k** points and then **b** vectors. The most inner loops are the band index n and m, respectively. In each block, the first line are 5 numbers. The first two numbers are the index of present **k** point and the index of neighboring point **k+b**, respectively. The next three numbers indicates in which unit cell **k+b** point lies. From the second line are the real and imaginary part of each matrix element. In each block, there are $N_{win} \times N_{win}$ complex numbers. An example file, generated by the input file 'Si.dat', is shown here:

```
Mmn_zero(k,b). band_num, kpt_num, bvector num, spinsize
           11          216            8            1
    1  216   -1   -1   -1
   -0.823117171036    0.565190443061
    0.002558098488   -0.001953947544
   -0.004406884297    0.003453571792
    0.009938520910   -0.007650660168
   -0.000871168877   -0.043168845537
   ... ...
   ... ...
   -0.000000007241    0.000000004251
    0.773274428859   -0.320260848984
```

```
 1  181   -1    0    0
-0.832487848180   -0.551294952424
-0.002473424886   -0.001467549142
 ........
 .....
 ...
 .
```

B. File format of '.amn' file

This file structure is closely following that in Wannier90 [76]. The first line of the file is the description of the whole file. Obviously, the four numbers in the second line are the number (N_{win}) of bands within the outer window, the number of k points, the number of target MLWFs and the number of spin component, respectively. Similarly, the data blocks are written in loops. The most outer loop is spin component and then **k** points, target MLWFs and number of bands. As described in the first line of this file. In each block, the first three integers are the band index, the index of MLWFs and index of k points, respectively. The next are real and imaginary of that matrix element. An example file, generated by the input file 'Si.dat', is shown here:

```
Amn. Fist line BANDNUM, KPTNUM, WANNUM, spinsize. Next is m n k and elements.Spin is the most outer loop.
          11          216           8           1
    1    1    1   -0.073309492802    0.043149612274
    2    1    1   -0.019286068732    0.012707420659
    3    1    1    0.033269068148   -0.022518524313
    ... ...
    ... ...
   10    1    1   -0.000000000626   -0.000000000724
   11    1    1   -0.000495942376    0.000166021125
    1    2    1   -0.073309471090    0.043149650676
    2    2    1    0.067585375965   -0.044489471852
    3    2    1   -0.049713011395    0.033692981021
    ........
    .....
    ...
    .
```

C. File format of '.eigen' file

This file contains the eigenenergies and eigenstates at each k point. The first line is the Fermi level of system. The number of bands is indicated in the second line of the file. The next data are mainly in two parts. The first part is the eigenenergies and the second one is the corresponding eigenstates. In each part, the loop of spin component is the most outer one. The next loop is k points, followed by band index. For eigenstates, there is one more inner loop for the basis set. An example file, generated by the input file 'Si.dat', is shown here:

```
Fermi level -0.130592
Number of bands 11
    1    1   -0.584590174255                          <-- 1st part: eigenenergies
    2    1   -0.140040771077
    3    1   -0.140040754871
    4    1   -0.140040734084
```

```
    5    1   -0.045528995979
    ... ...
    ... ...
WF kpt 1 (0.00000000,0.00000000,0.00000000)          <-- 2nd part: eigenstate
1 1  -0.4302513644  -0.2532440791
1 2  -0.0031362502  -0.0018459522
1 3  -0.0000122578  -0.0000183141
    ........
    .....
    ...
    .
```

D. File format of '.HWR' file

This file contains the hopping integrals between the mth MLWF, $|m, \mathbf{0}\rangle$, in the home unit cell and the nth MLWF, $|n, \mathbf{R}\rangle$, in the unit cell at $\mathbf{R}$. The matrix element $\langle m, \mathbf{0}|\hat{H}|n, \mathbf{R}\rangle$ is written in the following way. In '.HWR' file, the first line is just a description. The number of MLWFs, number of lattice vectors inside of Wigner-Seitz supercell are in the second and third line, respectively. The unit cell vectors are given in the fifth, sixth and seventh lines. Spin polarization, whether it is a non-spin polarized calculation or a spin polarized one with collinear or noncollinear magnetic configuration, is given in the eighth line. The ninth line gives the Fermi level. From the tenth line, the block data starts. The outer most loop is spin component. The next loop is for $\mathbf{R}$ and the last two are loops of m and n, respectively. Each $\mathbf{R}$ is written at the first line of each block together with its degeneracy. The index of m and n is printed and followed by the real and imaginary parts of hopping integrals in each line. An example file, generated by the input file 'Si.dat', is shown here:

```
Real-space Hamiltonian in Wannier Gauge on Wigner-Seitz supercell.  <--L01
Number of Wannier Function 8                                        <--L02
Number of Wigner-Seitz supercell 279                                <--L03
Lattice vector (in Bohr)                                            <--L04
   5.10000    0.00000    5.10000                                    <--L05
   0.00000    5.10000    5.10000                                    <--L06
   5.10000    5.10000    0.00000                                    <--L07
collinear calculation spinsize 1                                    <--L08
Fermi level -0.130592                                               <--L09
R (   -4    0    2 )    3                                          <--L10
   1     1     -0.000127004469   -0.000000003113                   <--L11
   1     2      0.000027279116   -0.000000007315                   <--L12
   1     3      0.000027305237   -0.000000006012                   <--L13
   1     4     -0.000020534349   -0.000000004246                   <--L14
    ... ...                                                         ... ...
    ... ...                                                         ... ...
   8     7      0.000027246023   -0.000000017704
   8     8     -0.000127075660   -0.000000007929
R (   -4    1    1 )    2
```

```
1     1     -0.000430127120    0.000000002912
1     2      0.000125799035    0.000000008133
1     3      0.000013650458    0.000000003690
1     4      0.000125829051    0.000000004688
 ........
 .....
 ...
 .
```

39.6 Automatic running test of MLWF

To check whether the MLWF calculation part is properly installed or not, an automatic running test for the NEGF calculation can be performed by

For the serial running

```
% ./openmx -runtestWF
```

For the MPI parallel running

```
% mpirun -np 16 openmx -runtestWF
```

For the OpenMP/MPI parallel running

```
% mpirun -np 8 openmx -runtestWF -nt 2
```

Then, OpenMX will run with eight test cases, and compare calculated results with the reference results which are stored in 'work/wf_example'. The comparison (absolute difference in the spread and Ω functions) is stored in a file 'runtestWF.result' in the directory 'work'. The reference results were calculated using a Xeon cluster machine. If the difference is within last seven digits, we may consider that the installation is successful.

40 Analysis of difference in two Gaussian cube files

A utility tool is provided to generate a Gaussian cube file which stores the difference between two Gaussian cube files for total charge density, spin density, and potentials. If you analyze the difference between two states, this tool would be helpful.

(1) Compiling of diff_gcube.c

There is a file 'diff_gcube.c' in the directory 'source'. Compile the file as follows:

```
% gcc diff_gcube.c -lm -o diff_gcube
```

When the compile is completed normally, then you can find an executable file, diff_gcube, in the directory 'source'. Please copy the executable file to the directory 'work'.

(2) Calculation of the difference

If you want to know the difference between two Gaussian cube files, input1.cube and input2.cube, and output the result to a file, output.cube, then perform as follows:

```
% ./diff_gcube input1.cube input2.cube output.cube
```

The difference is output to 'output.cube' in the Gaussian cube format. Thus, you can easily visualize the difference using many softwares, such as gOpenMol [48], Molekel [49], and XCrysDen [50]. In fact, Fig. 22 in the Section 'Electric field' was made by this procedure.

41 Analysis of difference in two geometrical structures

A utility tool is provided to analyze the difference between two geometrical coordinates in two xyz files which store Cartesian coordinates. The following three analyses are supported: a root mean square of deviation (RMSD) between two Cartesian coordinates defined by

$$\mathrm{RMSD} = \sqrt{\frac{\sum_i^{N_{\mathrm{atom}}} (R_i - R_i^0)^2}{N_{\mathrm{atom}}}}$$

a mean deviation (MD) between two Cartesian coordinates defined by

$$\mathrm{MD} = \frac{\sum_i^{N_{\mathrm{atom}}} |R_i - R_i^0|}{N_{\mathrm{atom}}}$$

and a mean deviation between bond lengths (MDBL) defined by

$$\mathrm{MDBL} = \frac{\sum_i^{N_{\mathrm{bond}}} |BL_i - BL_i^0|}{N_{\mathrm{bond}}}$$

where N_{atom} and N_{bond} are the number of atoms and the number of bonds with bond length (BL) within a cutoff radius. Also, the deviation vector between xyz coordinate of each atom is output to a vector file 'dgeo_vec.txt' in a gOpenMol format. If you analyze the difference between two geometries, this tool would be helpful.

(1) Compiling of diff_gcube.c

There is a file 'diff_gcube.c' in the directory 'source'. Compile the file as follows:

```
% gcc diff_geo.c -lm -o diff_geo
```

When the compile is completed normally, then you can find an executable file 'diff_geo' in the directory 'source'. Please copy the executable file to the directory 'work'.

(2) Calculation of the difference

You can find the following usage in the header part of diff_geo.c.

```
usage:
        ./diff_geo file1.xyz file2.xyz -d rmsd

   option
      -d rmsd      a root mean square of deviation
      -d md        a mean deviation
      -d mdbl 2.2  a mean deviation between bond lengths,
                   2.2 (Ang) means a cutoff bond length which
                   can be taken into account in the calculation
```

If you want to know RMSD between two Cartesian coordinates, run as follows:

```
% ./diff_geo file1.xyz file2.xyz -d rmsd
```

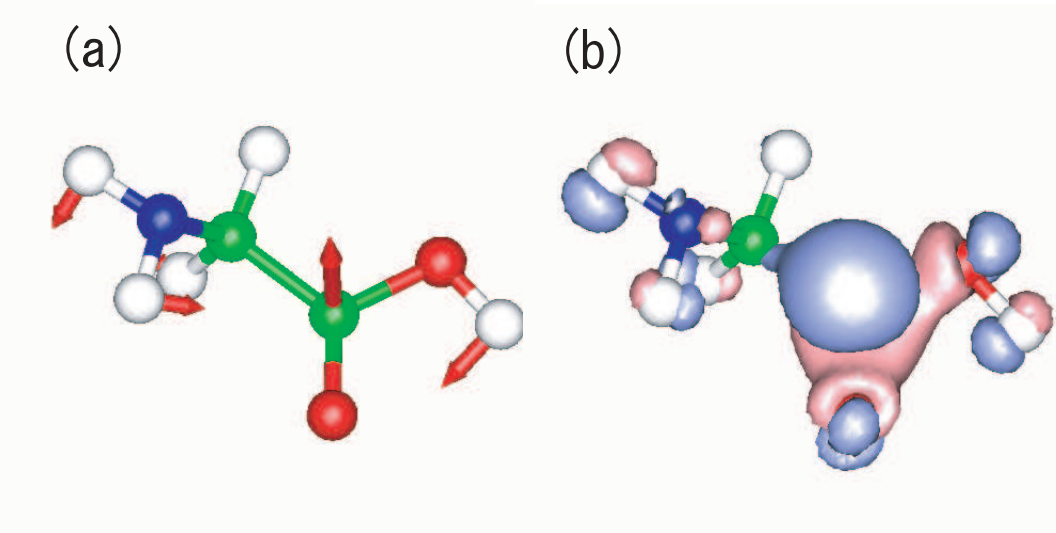


Figure 35: (a) Vectors corresponding to the deviation of atomic coordinates in optimized structures and (b) the difference of total charge density between a neutral and one electron doping glycine molecules. These figures were visualized by gOpenMol [48]. In Fig. (b) blue and red colors indicate the decrease and increase of total charge density, respectively.

The calculated result appears in the standard output (your display). Also, a vector file 'dgeo_vec.txt' is generated in a gOpenMol format, which stores the difference between Cartesian coordinates of each atom in a vector form. This file can be visualized using 'Plot Vector File' in gOpenMol. When MDBL is calculated, please give a cutoff bond length (Å). Bond lengths below the cutoff bond length are taken account of this RMSD calculation. Figure 35 shows vectors corresponding to the deviation of atomic coordinates in optimized structures and the difference of total charge density between a neutral and one electron doping glycine molecules. We see that the large structural change takes place together with the large charge deviation. This example illustrates that the tool would be useful when we want to know how the structure is changed by the charge doping, the electric field, and the basis set.

42 Analysis of difference charge density induced by the interaction

The redistribution of charge (spin) density induced by the interaction between two systems A and B can be analyzed by the following procedure:

(i) calculate the composite system consisting of A and B

Then, you will have a cube file for charge (spin) density. Let it 'AB.cube'. Also, you will find 'Grid_Origin' in the standard output which gives x-, y-, and z-components of the origin of the regular grid as:

```
Grid_Origin  xxx  yyy  zzz
```

The values will be used in the following calculations (ii) and (iii).

(ii) calculate the system A

This calculation must be performed by the same calculation condition with the same unit cell as in the composite system consisting of A and B. Also, the coordinates of the system A must be the same as in the calculation (i). To use the same origin as in the calculation (i) rather than the use of an automatically determined origin, you have to include the following keyword in your input file.

```
scf.fixed.grid  xxx  yyy  zzz
```

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation (i). Then, you will have a cube file for charge (spin) density. Let it 'A.cube'.

(iii) calculate the system B

As well as the calculation (ii), this calculation must be performed by the same calculation condition with the same unit cell as in the composite system consisting of A and B. Also, the coordinates of the system B must be the same as in the calculation (i). To use the same origin as in the calculation (i) rather than the use of an automatically determined origin, you have to include the following keyword in your input file.

```
scf.fixed.grid  xxx  yyy  zzz
```

where 'xxx yyy zzz' is the coordinate of the origin you got in the calculation (i). Then, you will have a cube file for charge (spin) density. Let it 'B.cube'.

(iv) compile two codes

compile two codes as follows:

```
% gcc diff_gcube.c -lm -o diff_gcube
% gcc add_gcube.c -lm -o add_gcube
```

(v) generate a cube file for difference charge (spin) density

First, generate a cube file for the superposition of two charge (spin) densities of the systems A and B by

```
% ./add_gcube A.cube B.cube A_B.cube
```

The file 'A_B.cube' is the cube file for the superposition of charge (spin) density of two isolated systems. Then, you can generate a cube file for the difference charge (spin) density induced by the interaction as follows:

```
% ./diff_gcube AB.cube A_B.cube dAB.cube
```

The file 'dAB.cube' is the cube file for the difference charge (spin) density induced by the interaction, where the difference means (AB - A_B).

43 Automatic determination of the cell size

When you calculate an isolated system, you are required to provide a super cell so that the isolated system does not overlap with the image systems in the repeated cells. The larger cell size can cause a numerical inefficiency, since a larger number of grids are used in the solution of the Poisson's equation in this case. Therefore, the use of the minimum cell size is desirable in terms of computational efficiency. OpenMX supports the requirement. If you remove the specification for the cell size, that is, from '<Atoms.UnitVectors' to 'Atoms.UnitVectors>', then OpenMX automatically determines an appropriate cell size which does not overlap the next cells and fulfills the required cutoff energy. The determined cell vectors are displayed in the standard output like this:

```
<Set_Cluster_UnitCell> automatically determined UnitCell(Ang.)
<Set_Cluster_UnitCell> from atomic positions and Rc of PAOs (margin= 10.00%)
<Set_Cluster_UnitCell> 6.614718 0.000000 0.000000
<Set_Cluster_UnitCell> 0.000000 6.041246 0.000000
<Set_Cluster_UnitCell> 0.000000 0.000000 6.614718

widened unit cell to fit energy cutoff (Ang.)
A = 6.744142 0.000000 0.000000 (48)
B = 0.000000 6.322633 0.000000 (45)
C = 0.000000 0.000000 6.744142 (48)
```

44 Selection of lapack routine

In all the calculations: cluster, band, and O(N) calculations, a lapack routine is used to solve eigenvalues and eigenvectors of the tridiagonalized matrix. However, we see a platform dependency of lapack routines to solve the tridiagonalized matrix with respect to computational robustness. So, three different lapack routines are available in OpenMX Ver. 3.5 by the following keyword 'scf.lapack.dste':

```
scf.lapack.dste    dstevx    # dstegr|dstedc|dstevx, default=dstevx
```

These lapack routines, dstegr, dstedc, and dstevx, are based on a multiple relatively robust representation (MR3) scheme [66], a divide and conquer (DC) algorithm [67], and QR and inverse iteration algorithm, respectively. For further details, see the lapack website [68]. Our experiences suggest that the computational speed is as follows:

```
dstevx < dstedc < dstegr
```

In contrast to the computational speed, the computational robustness seems to be opposite as follows:

```
dstegr < dstedc < dstevx
```

So, an appropriate one (robuster and faster) on your computational environment should be selected by this keyword 'scf.lapack.dste'. The default is 'dstevx'.

In the cluster and band calculations, only eigenvectors of occupied and lower exited states are evaluated for saving the computation time when 'dstevx' is used. Thus, it is highly recommended to use 'dstevx' instead of 'dstedc' and 'dstegr' in the cases.

45 Interface for developers

An interface for developers is provided. If you want to use the Kohn-Sham Hamiltonian, the overlap, and the density matrices, Then these data can be utilized by the following steps.

1. HS.fileout

   Include the keyword, HS.fileout, in your input file as follows:

   ```
   HS.fileout                      on      # on|off, default=off
   ```

   Then, these data are output to a file '*.scfout' where * means System.Name in your input file.

2. make analysis_example

   In the directory 'source' compile by

   ```
   % make analysis_example
   ```

   Then, an executable file, analysis_example, is generated in the directory, 'work'.

3. ./analysis_example *.scfout

   Move to the directory 'work', and then perform the program as follows:

   ```
   % ./analysis_example *.scfout
   or
   % ./analysis_example *.scfout > HS.out
   ```

   You can find the elements of the Hamiltonian, the overlap, and the density matrices in a file 'HS.out'

4. explanation of analysis_example

   In a file 'analysis_example.c' you can find a detailed description for these data. A part of the description is as follows:

   ```
   ************************************************************
     You can utilize a filename.scfout which is generated by the SCF
     calculation of OpenMX by the following procedure:

      1. Define your main routine as follows:

            int main(int argc, char *argv[])

      2. Include a header file, "read_scfout.h", in your main routine
         (if you want, also in other routines) as follows:

            #include "read_scfout.h"

      3. Call a function, read_scfout(), in the main routine as follows:

            read_scfout(argv);
   ************************************************************
   ```

46 Automatic force tester

An effective way of assuring the reliability of implementation of many functionalities is to compare analytic and numerical forces. If any program bug is introduced, they will not be consistent with each other. To do this, one can run an automatic tester by

For serial running

```
% ./openmx -forcetest 0
```

For parallel running

```
% ./openmx -forcetest 0 "mpirun -np 4 openmx"
```

where '0' is a flag to specify energy terms to be included in the consistency check, and one can change 0 to 8. Each number corresponds to

flag	0	1	2	3	4	5	6	7	8
Kinetic	1	0	1	0	0	0	0	0	0
Non-local	1	0	0	1	0	0	0	0	0
Neutral atom	1	0	0	0	1	0	0	0	0
diff Hartree	1	0	0	0	0	1	0	0	0
Ex-Corr	1	0	0	0	0	0	1	0	0
E. Field	1	0	0	0	0	0	0	1	0
Hubbard U	1	0	0	0	0	0	0	0	1

where '1' means that it is included in the force consistency check. In a directory 'work/force_example', there are 37 test inputs which are used for the force consistency check. After finishing the run, a file 'forcetest.result' is generated in the directory 'work'. You will see results of the comparison as follows:

```
force_example/C2_GGA.dat
  flag= 0
  Numerical force= -(Utot(s+ds)-Utot(s-ds))/(2*ds)
  ds=   0.0001000000
  Forces (Hartree/Bohr) on atom 1
                               x              y              z
  Analytic force       -1.514128677000 -1.262159787942 -1.025428240858
  Numerical force      -1.514450620572 -1.262264605408 -1.025386683997
  diff                  0.000321943572  0.000104817467 -0.000041556861

force_example/C2_LDA.dat
  flag= 0
  Numerical force= -(Utot(s+ds)-Utot(s-ds))/(2*ds)
  ......
  ....
```

47 Automatic memory leak tester

In OpenMX, the memory used is dynamically allocated when it is required. However, the dynamic memory allocation causes often a serious memory leak which wastes the memory used as the MD steps increase. To check the memory leak, one can run OpenMX as follows:

For serial running

```
% ./openmx -mltest
```

For parallel running

```
% ./openmx -mltest "mpirun -np 4 openmx"
```

By monitoring VSZ and RSS actually used at the same monitoring point in the program code for 13 test inputs in a directory 'work/ml_example', one can find whether the memory leak takes place or not. After finishing the run, a file 'mltest.result' is generated in the directory 'work'. You will see the monitored VSZ and RSS as a function of MD steps as follows:

```
   1     ml_example/Co4.dat

                 CPU (%)     VSZ (kbyte)    RSS (kbyte)
MD_iter=   1     99.600      42692          18348
MD_iter=   2     99.600     176072         168496
MD_iter=   3     99.300     181624         174052
MD_iter=   4     99.100     176060         168488
MD_iter=   5     99.000     181624         174052
MD_iter=   6     98.800     176084         168512
MD_iter=   7     98.800     181624         174052
MD_iter=   8     99.900     176060         168488
MD_iter=   9     99.900     181624         174052
MD_iter=  10     99.600     176084         168512
MD_iter=  11     99.700     181624         174052
MD_iter=  12     99.600     176084         168512
MD_iter=  13     99.600     181624         174052
MD_iter=  14     99.500     181624         174052
MD_iter=  15     99.500     181628         174056
MD_iter=  16     99.400     181628         174056
MD_iter=  17     99.300     181624         174052
MD_iter=  18     99.200     181628         174056
MD_iter=  19     99.200     181620         174048
MD_iter=  20     99.100     181628         174056
MD_iter=  21     99.100     181620         174048
MD_iter=  22     99.200     181628         174056
MD_iter=  23     99.200     181620         174048
MD_iter=  24     98.800     181628         174056
```

```
MD_iter=  25     98.800     181620         174048
MD_iter=  26     98.800     181628         174056
MD_iter=  27     99.200     181624         174052
MD_iter=  28     99.200     181628         174056
MD_iter=  29     99.100     181624         174052
MD_iter=  30     99.100     181628         174056
```

```
   2     ml_example/Co4+U.dat

                 CPU (%)     VSZ (kbyte)    RSS (kbyte)
MD_iter=   1     99.800      42800           18456
MD_iter=   2     99.700     176172         168664
   ......
   ....
```

48 Examples of the input files

For your convenience, the input files of examples shown in the manual are available in the directory 'work' as listed below:

```
Molecules or clusters
    C60.dat             SCF calc. of a C60 molecule
    C60_DC.dat          DC calc. of a C60 molecule
    CG15c_DC.dat        DC calc. of DNA
    Cr2_CNC.dat         Constrained DFT calc. of a Cr2 dimer
    Doped_NT.dat        SCF calc. of doped carbon nanotube
    Fe2.dat             SCF calc. of a Fe2 dimer
    Gly_NH.dat          Nose-Hoover MD of a glycine molecule
    Gly_VS.dat          Velocity scaling MD of a glycine molecule
    H2O.dat             Geometry opt. of a water molecule
    MCCN.dat            DC calc. of a a multiply connected carbon nanotube
    Methane2.dat        Geometry opt. of a distorted methane molecule
    Methane.dat         SCF calc. of a  methane molecule
    Methane_OO.dat      Orbital optimization of a methane molecule
    Mn12.dat            SCF calc. of a single molecular magnet, Mn12
    Mol_MnO_NC.dat      Non-collinear SCF calc. of a MnO molecule
    Nitro_Benzene.dat   SCF calc. of a nitro benzene molecule under E-field
    Pt13.dat            SCF calc. of a Pt13 cluster
    Pt63.dat            SCF calc. of a Pt63 cluster
    SialicAcid.dat      SCF calc. of a sialic acid molecule
    Valorphin_DC.dat    DC calc. of valorphin molecule

Bulk
    Cdia.dat            SCF calc. of bulk diamond
    MnO_NC.dat          Non-collinear SCF calc. of bulk MnO
    FeO_NC.dat          Non-collinear SCF calc. of bulk FeO
    CoO_NC.dat          Non-collinear SCF calc. of bulk CoO
    NiO_NC.dat          Non-collinear SCF calc. of bulk NiO
    Crys-NiO.dat        SCF calc. of bulk NiO
    DIA64_Band.dat      SCF calc. of bulk diamond including 64 atoms
    DIA8_DC.dat         DC calc. of bulk diamond including 8 atoms
    DIA64_DC.dat        DC calc. of bulk diamond including 64 atoms
    DIA216_DC.dat       DC calc. of bulk diamond including 216 atoms
    DIA512_DC.dat       DC calc. of bulk diamond including 512 atoms
    DIA512-1.dat        Krylov O(N) calc. of bulk diamond including 512 atoms
    Febcc2.dat          SCF calc. of bcc Fe
    GaAs.dat            Non-collinear calc. of bulk gallium arsenide
    NaCl.dat            SCF calc. of bulk NaCl
    NaCl_FC.dat         SCF calc. of bulk NaCl with a Cl-site vacancy
    Si8.dat             Geometry opt. of distorted Si bulk
```

49 Known problems

- Overcompleteness of basis functions

  When a large number of basis functions is used for dense bulk systems with fcc, hcp, and bcc like structures, the basis set tends to be overcomplete. In such a case, you may observe erratic eigenvalues. To avoid the overcompleteness, a small number of optimized basis functions should be used.

- Instability of lapack routines for high symmetry systems

  For a system with a highly symmetric structure, the lapack diagonalization routines may fail to find the correct eigenvectors. while this phenomenon strongly depends on the computational environment. For such a case, try to find a nicely working routine by 'scf.lapack.dste'.

- Difficulty in getting the SCF convergence

  For large-scale systems with a complex (non-collinear) magnetic structure, a metallic electric structure, or the mixture, it is quite difficult to get the SCF convergence. In such a case, one has to mix the charge density very slowly, indicating that the number of SCF steps to get the convergence becomes large unfortunately.

- Difficulty in getting the optimized structure

  For weak interacting systems such as molecular systems, it is not easy to obtain a completely optimized structure, leading that the large number of iteration steps is required. Although the default value of criterion for geometrical optimization is 10^{-4} Hartree/bohr for the largest force, it would be a compromise to increase the criterion from 10^{-4} to 5×10^{-4} in such a case.

50 OpenMX Forum

For discussion of technical issues on OpenMX and ADPACK, there is a forum (http://www.openmx-square.org/forum/patio.cgi). It is expected that the forum is utilized for sharing tips in use of OpenMX and for further code development. Points of concern for use of this forum can be found in http://www.openmx-square.org/forum/note.html

51 Others

Program

The program package is written in the C language, including one makefile

```
makefile,
```

nine header files

```
f77func.h
Inputtools.h
lapack_prototypes.h
mimic_mpi.h
mimic_omp.h
openmx_common.h
read_scfout.h
tran_prototypes.h
tran_variables.h
```

and 248 routines

```
add_gcube.c Allocate_Arrays.c analysis_example.c AngularF.c
AtomicDenF.c AtomicPCCF.c Band_DFT_Col.c Band_DFT_Dosout.c
Band_DFT_kpath.c Band_DFT_MO.c Band_DFT_NonCol.c bandgnu13.c
BentNT.c BroadCast_ComplexMatrix.c BroadCast_ReMatrix.c
check_lead.c Cluster_DFT.c Cluster_DFT_Dosout.c Cont_Matrix0.c
Cont_Matrix1.c Cont_Matrix2.c Cont_Matrix3.c Contract_Hamiltonian.c
Contract_iHNL.c Cutoff.c dampingF.c deri_dampingF.c DFT.c
diff_gcube.c diff_geo.c DIIS_Mixing_DM.c DIIS_Mixing_Rhok.c
Divide_Conquer.c Divide_Conquer_Dosout.c DosMain.c
Dr_AtomicDenF.c Dr_AtomicPCCF.c Dr_RadialF.c Dr_VH_AtomF.c
Dr_VNAF.c dtime.c Eff_Hub_Pot.c EigenBand_lapack.c
Eigen_lapack.c Eigen_PHH.c Eigen_PReHH.c esp.c EulerAngle_Spin.c
File_CntCoes.c Find_ApproxFactN.c Find_CGrids.c find_Emin0.c
find_Emin2.c find_Emin.c find_Emin_withS.c Force.c
Force_test.c frac2xyz.c Free_Arrays.c FT_NLP.c FT_PAO.c
FT_ProductPAO.c FT_ProExpn_VNA.c FT_VNA.c Fuzzy_Weight.c
Gaunt.c Gauss_Legendre.c GDivide_Conquer.c GDivide_Conquer_Dosout.c
Generate_Wannier.c Generating_MP_Special_Kpt.c Get_Cnt_dOrbitals.c
Get_Cnt_Orbitals.c Get_dOrbitals.c Get_OneD_HS_Col.c Get_Orbitals.c
GR_Pulay_DM.c Hamiltonian_Band.c Hamiltonian_Band_NC.c
Hamiltonian_Cluster.c Hamiltonian_Cluster_NC.c Hamiltonian_Cluster_SO.c
init_alloc_first.c init.c Initial_CntCoes.c Init_List_YOUSO.c
Input_std.c Inputtools.c io_tester.c IS_Hotelling.c IS_Lanczos.c
IS_LU.c IS_Taylor.c iterout.c iterout_md.c jx.c
Kerker_Mixing_Rhok.c Krylov.c lapack_dstedc1.c lapack_dstedc2.c
```

lapack_dstegr1.c lapack_dstegr2.c lapack_dsteqr1.c
lapack_dstevx1.c lapack_dstevx2.c Lapack_LU_inverse.c
LU_inverse.c Make_Comm_Worlds.c Make_FracCoord.c
Make_InputFile_with_FinalCoord.c Maketest.c
malloc_multidimarray.c MD_pac.c Memory_Leak_test.c Merge_LogFile.c
mimic_mpi.c mimic_omp.c Mio_tester2.c Mio_tester.c
Mixing_DM.c mpi_multi_world2.c mpi_multi_world.c Mulliken_Charge.c
Nonlocal_Basis.c Nonlocal_RadialF.c Occupation_Number_LDA_U.c
openmx.c openmx_common.c Opt_Contraction.c OpticalConductivityMain.c
Orbital_Moment.c OutData.c Output_CompTime.c outputfile1.c
Overlap_Band.c Overlap_Cluster.c pdb2pao.c PhiF.c
Poisson.c Poisson_ESM.c polB.c Pot_NeutralAtom.c PrintMemory.c
PrintMemory_Fix.c QuickSort.c RadialF.c readfile.c
read_scfout.c RecursionS_B.c RecursionS_C.c RecursionS_D.c
RecursionS_E2.c RecursionS_E.c RecursionS_F.c
RecursionS_G.c RecursionS_H2.c RecursionS_H.c
RecursionS_I.c ReLU_inverse.c RestartFileDFT.c
RF_BesselF.c rmmpi.c rot.c Runtest.c SCF2File.c
Set_Aden_Grid.c Set_Allocate_Atom2CPU.c Set_Density_Grid.c
Set_Hamiltonian.c Set_Nonlocal.c Set_OLP_Kin.c Set_Orbitals_Grid.c
SetPara_DFT.c Set_ProExpn_VNA.c setup_CPU_group.c Set_Vpot.c
Set_XC_Grid.c Show_DFT_DATA.c Simple_Mixing_DM.c Smoothing_Func.c
Spherical_Bessel.c test0.c test2.c test3.c test.c test_mpi2.c
test_mpi3.c test_mpi4.c test_mpi.c test_openmp2.c
test_openmp3.c test_openmp.c Tetrahedron_Blochl.c
Timetool.c Total_Energy.c TRAN_Add_ADensity_Lead.c
TRAN_Add_Density_Lead.c TRAN_adjust_Ngrid.c TRAN_Allocate.c
TRAN_Apply_Bias2e.c TRAN_Calc_CentGreen.c TRAN_Calc_CentGreenLesser.c
TRAN_Calc_GridBound.c TRAN_Calc_Hopping_G.c TRAN_Calc_OneTransmission.c
TRAN_Calc_SelfEnergy.c TRAN_Calc_SurfGreen.c TRAN_Check_Input.c
TRAN_Check_Region.c TRAN_Check_Region_Lead.c TRAN_Credit.c
TRAN_Deallocate_Electrode_Grid.c TRAN_Deallocate_RestartFile.c
TRAN_DFT.c TRAN_DFT_Dosout.c TRAN_Distribute_Node.c
TRAN_Input_std_Atoms.c TRAN_Input_std.c TranMain.c
TRAN_Output_HKS.c TRAN_Output_HKS_Write_Grid.c TRAN_Output_Trans_HS.c
TRAN_Poisson.c TRAN_Print.c TRAN_Print_Grid.c TRAN_Read.c
TRAN_RestartFile.c TRAN_Set_CentOverlap.c TRAN_Set_Electrode_Grid.c
TRAN_Set_IntegPath.c TRAN_Set_MP.c TRAN_Set_SurfOverlap.c
TRAN_Set_Value.c Truncated_System.c truncation.c unit2xyz.c
VH_AtomF.c VNAF.c Voronoi_Charge.c Voronoi_Orbital_Moment.c
XC_CA_LSDA.c XC_Ceperly_Alder.c XC_EX.c XC_PBE.c XC_PW91C.c
xyz2spherical.c zero_cfrac.c zero_fermi.c

In addition, the following library packages are linked:

lapack,

```
blas,
fftw,
MPICH or LAM
omp
```

Copyright of the program package

The distribution of this program package follows the practice of the GNU General Public License [47]. Moreover, the author, Taisuke Ozaki, possesses the copyright of the original version of this program package. We cannot offer any guarantees in your use of this program package. However, when you report program bugs, we will cooperate and work well as much as possible together with you to remove the problems.

Acknowledgment

One of us (T.O.) would like to thank many colleagues in JRCAT and RICS-AIST for helpful suggestions and comments. One of us (T.O.) was partly supported by the following national projects: SYNAF-NEDO [70], ACT-JST [71], NAREGI [72], and CREST-JST [73].

References

[1] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964); W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[2] D. M. Ceperley and B. J. Alder, Phys. Rev. Lett., 45, 566(1980); J. P. Perdew and A. Zunger, Phys. Rev. B **23,** 5048 (1981).

[3] J. P. Perdew and A. Zunger, Phys. Rev. B **23,** 5048 (1981).

[4] J. P. Perdew and Y. Wang, Phys.Rev.B **45**, 13244 (1992).

[5] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

[6] U. Von. Barth and L. Hedin, J. Phys. C: Solid State Phys. **5**, 1629 (1972).

[7] J. Kübler, K-H. Höck, J. Sticht, and A. R. Williams, J. Phys. F: Met. Phys. **18**, 469 (1988).

[8] J. Sticht, K-H. Höck, and J. Kübler, J. Phys.: Condens. Matter **1**, 8155 (1989).

[9] T. Oda, A. Pasquarello, and R.Car, Phys. Rev. Lett. **80**, 3622 (1998).

[10] A. H. MacDonald and S. H. Vosko, J. Phys. C: Solid State Phys. 12, 2977 (1979).

[11] Ph. Kurz, F. Forster, L. Nordstrom, G, Bihlmayer, and S. Blugel, Phys. Rev. B **69**, 024415 (2004).

[12] R. D. King-Smith and D. Vanderbilt, Phys. Rev. B **47**, 1651 (1993).

[13] G. Theurich and N. A. Hill, Phys. Rev. B **64**, 073106 (2001).

[14] A. I. Liechtenstein, M. I. Katsnelson, V. P. Antropov, and V. A. Gubanov, J. Mag. Mag. Mat. **67**, 65 (1987).

[15] M. J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **70**, 184421 (2004).

[16] M. J. Han, T. Ozaki, and J. Yu, Phys. Rev. B **74**, 045110 (2006).

[17] L. V. Woodcock, Chem. Phys. Lett. **10** ,257 (1971).

[18] S. Nose, J. Chem. Phys. **81**, 511 (1984); S. Nose, Mol. Phys. **52**, 255 (1984); G. H. Hoover, Phys. Rev. A **31**, 1695 (1985)).

[19] G. B. Bachelet, D. R. Hamann, and M. Schluter, Phys. Rev. B **26**, 4199 (1982).

[20] N. Troullier and J. L. Martine, Phys. Rev. B **43**, 1993 (1991).

[21] L. Kleinman and D. M. Bylander, Phys. Rev. Lett. **48,** 1425 (1982).

[22] P. E. Blochl, Phys. Rev. B **41**, 5414 (1990).

[23] T. Ozaki, Phys. Rev. B. **67**, 155108, (2003); T. Ozaki and H. Kino, Phys. Rev. B **69**, 195113 (2004).

[24] T. Ozaki and H. Kino, Phys. Rev. B **72**, 045121 (2005).

[25] T. Ozaki, Phys. Rev. B **74**, 245101 (2006).

[26] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias and J. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992) and references therein.

[27] O. F. Sankey and D. J. Niklewski, Phys. Rev. B. **40**, 3979 (1989)

[28] W. Yang, Phys.Rev.Lett. **66**, 1438 (1991)

[29] P. Ordejon, E. Artacho, and J. M. Soler, Phys. Rev. B. **53**, 10441 (1996)

[30] D. R. Bowler and M. J. Gillan, Chem. Phys. Lett. **325**, 475 (2000).

[31] G. Kresse and J. Furthmeuller, Phys. Rev. B. **54**, 11169 (1996)

[32] G. P. Kerker, Phys. Rev. B **23**, 3082 (1981).

[33] T. A. Arias, M. C. Payne, and J. D. Joannopoulos, Phys. Rev. B **45**, 1538 (1992).

[34] D. Alfe, Comp. Phys. Commun. **118**, 32 (1999).

[35] P. Csaszar and P. Pulay, J. Mol. Struct. (Theochem) **114**, 31 (1984).

[36] J. Baker, J. Comput. Chem. **7**, 385 (1986)

[37] A. Banerjee, N. Adams, J. Simons, R. Shepard, J. Phys. Chem. **89**, 52 (1985)

[38] C. G. Broyden, J. Inst. Math. Appl. 6, 76 (1970); R. Fletcher, Comput. J. 13, 317 (1970); D. Goldrarb, Math. Comp. 24, 23 (1970); D. F. Shanno, Math. Comp. 24, 647 (1970).

[39] P. E. Blochl, O. Jepsen and O. K. Andersen, Phys. Rev. B **49**, 16223 (1994).

[40] A. D. Becke and R. M. Dickson, J. Chem. Phys. **89**, 2993 (1988).

[41] A. Svane and O. Gunnarsson, Phys. Rev. Lett. **65**, 1148 (1990).

[42] Kino's note.

[43] T. Ozaki, K. Nishio, and H. Kino, arXiv:0908.4142.

[44] T. Ozaki, Phys. Rev. B **75**, 035123 (2007).

[45] G. C. Liang, A. W. Ghosh, M. Paulsson, and S. Datta, Phys. Rev. B. **69**, 115302 (2004).

[46] H. Weng, T. Ozaki, and K. Terakura, Phys. Rev. B **79**, 235118 (2009).

[47] http://www.gnu.org/

[48] http://www.csc.fi/gopenmol/

[49] http://www.cscs.ch/molekel/

[50] http://www.xcrysden.org/

[51] T. Lis, Acta Crystallogra. B **36**, 2042 (1980).

[52] T. P. Davis T. J. Gillespie, F. Porreca, Peptides **10**, 747 (1989).

[53] A. Goldstein, S. Tachibana, L. I. Lowney, M. Hunkapiller, and L. Hood, Proc. Natl. Acad. Sci. U. S. A. **76**, 6666 (1979).

[54] U. C. Singh and P. A. Kollman, J. Comp. Chem. **5**, 129(1984).

[55] L. E. Chirlian and M. M. Francl, J. Com. Chem. **8**, 894(1987).

[56] B. H. Besler, K. M. Merz Jr. and P. A. Kollman, J. Comp. Chem. **11**, 431(1990).

[57] http://www.webelements.com/

[58] M. Cardona, N. E. Christensen, and G. Gasol, Phys. Rev. B **38**, 1806 (1988).

[59] G. Theurich and N. A. Hill, Phys. Rev. B **64**, 073106 (2001).

[60] *Physics of Group IV Elements and III-V Compounds*, edited by O.Madelung, M.Schulz, and H. Weiss, Landolt-Büornstein, New Series, Group 3, Vol. 17, Pt.a (Springer, Berlin, 1982).

[61] T. Ono and K. Hirose, Phys. Rev. B **72**, 085105 (2005).

[62] W. N. Mei, L. L. Boyer, M. J. Mehl, M. M. Ossowski, and H. T. Stokes, Phys. Rev. B **61**, 11425 (2000).

[63] I. V. Solovyev. A. I. Liechtenstein, K. Terakura, Phys. Rev. Lett. **80**, 5758.

[64] K. Knopfle, L. M. Sandratskii, and J. Kubler, J. Phys:Condens. Matter **9**, 7095 (1997).

[65] http://www.openmx-square.org/

[66] I. S. Dhillon and B. N. Parlett, SIAM J. Matrix Anal. Appl. **25**, 858 (2004).

[67] J. J. M. Cuppen, Numer. Math. **36**, 177 (1981); M. Gu and S. C. Eisenstat, SIAM J. Mat. Anal. Appl. **16**, 172 (1995).

[68] http://www.netlib.org/lapack/

[69] http://www.nongnu.org/xmakemol/

[70] http://www.nanoworld.jp/synaf/

[71] http://act.jst.go.jp/

[72] http://ccinfo.ims.ac.jp/nanogrid/

[73] http://www.jst.go.jp/

[74] N. Mazari and D. Vanderbilt, Phys. Rev. B **56**, 12 847 (1997).

[75] I. Souza, N. Marzari and D. Vanderbilt, Phys. Rev. B **65**, 035109 (2001).

[76] http://www.wannier.org/

Index