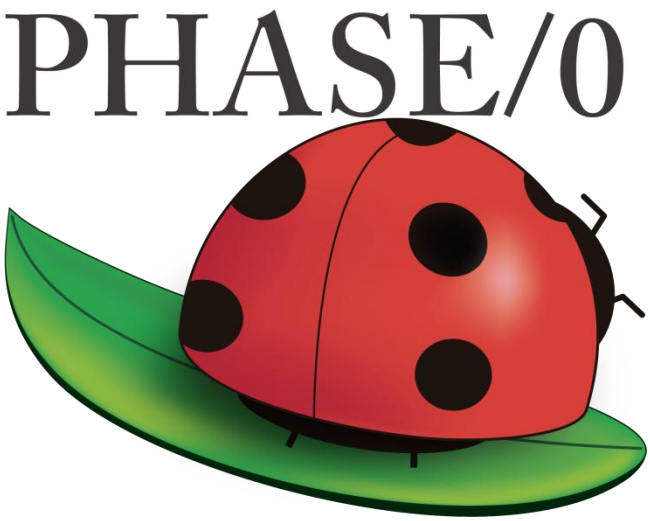
First-principles Electronic Structure Calculation Program
PHASE/0 2015
User's Manual



0.10.2.01 (2014/04/24)
https://azuma.nims.go.jp

PHASE System

COPYRIGHT of the program codes

HISTORY

The original version of this set of the computer programs "PHASE" was developed by the members of the Theory Group of Joint Research Center for Atom Technology (JRCAT), based in Tsukuba, in the period 1993-2001. The names of the contributors to the original version are Hideki Katagiri, K. Kato, T. Miyazaki, Y. Morikawa, H. Sawada, T. Uchiyama, T. Uda and T. Yamasaki. These contributors has agreed with that the Institute of Industrial Science (IIS), the University of Tokyo, distributes this program as a free software.

Since 2002, this program set had been intensively developed as a part of the following national projects supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan; "Frontier Simulation Software for Industrial Science (FSIS)" from 2002 to 2005, "Revolutionary Simulation Software (RSS21)" from 2006 to 2008. "Research and Development of Innovative Simulation Software (RISS)" from 2008 to 2013. These projects is lead by the Center for Research on Innovative Simulation Software (CISS), the Institute of Industrial Science (IIS), the University of Tokyo.

Since 2013, this program set has been further developed centering on PHASE System Consortium.

The activity of development of this program set has been supervised by Takahisa Ohno.

CONTACT ADDRESS

PHASE System Consortium
E-mail: phase_system@nims.go.jp URL https://azuma.nims.go.jp

Contents

1. Introduction

1.1 Overview of PHASE-SYSTEM

PHASE-SYSTEM is a set of program packages for performing simulations of nanosize materials. It consists of PHASE for first-principles electronic structure calculations, UVSOR for dielectric function calculations, ASCOT for quantum transport property calculations, CIAO for all-electron calculations of an atom and the generation of pseudopotentials, and PHASE Viewer, which is a graphical user interface (GUI) for the PHASE package. Note that PHASE-STSTEM, the one program package PHASE, and an executable "phase" are different entities.

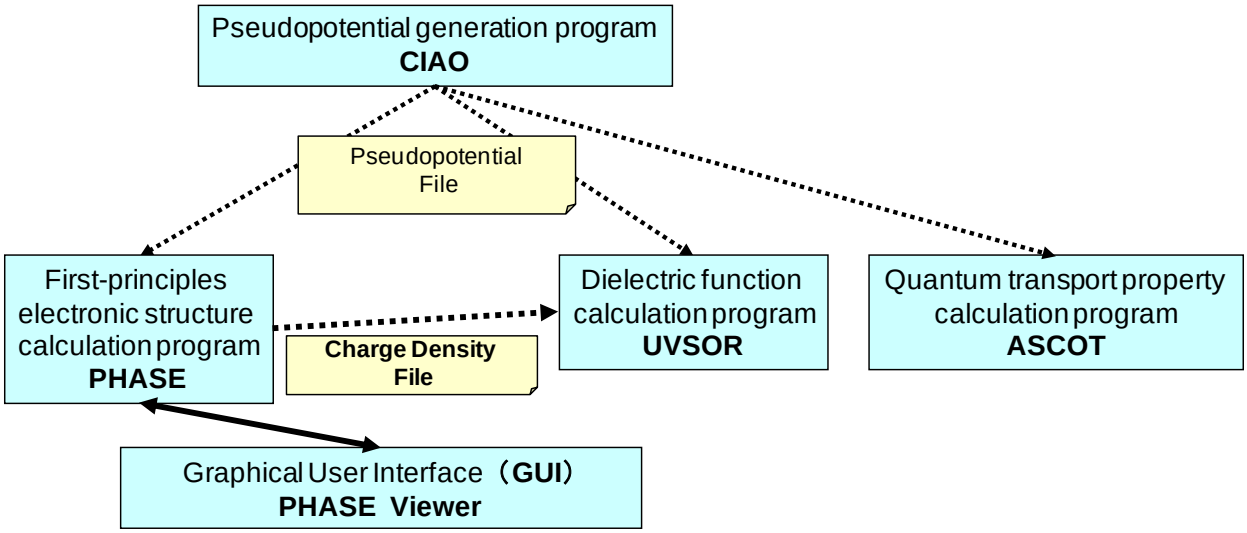


Figure 1.1 Program packages included in PHASE-SYSTEM

Table 1.1 Brief overview of the program packages included in PHASE-SYSTEM

Program/Package	Brief overview
PHASE (First-principles electronic structure calculation program)	PHASE is a first-principles electronic structure calculation program based on the density functional theory (DFT) and the pseudopotential scheme. It can calculate total energy, charge density, density of states, and band structures, and it can perform molecular dynamics simulations.
UVSOR (Dielectric function calculation program)	UVSOR is a program for calculating the dielectric functions of materials based on the DFT and the pseudopotential scheme. It can calculate both electron and lattice dielectric functions quantitatively. It can predict the dielectric function of high-k materials that have large lattice dielectric functions.
CIAO (Pseudopotential generation program)	CIAO is a program package for all-electron calculations of an atom and the generation of pseudopotentials used in PHASE, UVSOR, and ASCOT.
ASCOT (Quantum transport property calculation program)	ASCOT is a program package for calculating electronic structure and quantum transport properties of nanostructures bridging semi-infinite electrodes. It is based on the non-equilibrium Green's function method.
PHASE Viewer (graphical user interface)	This is a graphical user interface (GUI) for the PHASE package. It helps users construct and edit input files, execute calculations, and visualize calculation results.

This manual is for PHASE, a first-principles electronic structure calculation program, and the related tools

included in the package.

1.2 What is PHASE?

1.2.1 Calculation functions of PHASE

PHASE is a first-principles electronic structure calculation program based on DFT [1] and the pseudopotential scheme [2-4]. Using no parameters fitted to experimental results, this program can predict the physical properties of materials that are not found in any experiments, with reasonably high accuracies. It can also calculate various physical quantities using the calculated wave functions. In addition to electronic states, PHASE can calculate the total energy and the forces acting on atoms. Using these, users can obtain stable structures by minimizing forces and perform molecular dynamics simulations to see the time evolution of a system.

The calculation functions available in PHASE are briefly summarized in the following table.

Calculation functions	Corresponding physical quantities (Physical properties, material behaviors, phenomena…)
Electronic structure calculation	Density of states (DOS) Band structure Charge density
Energy, force	Total energy and forces acting on atoms Lattice parameters, elastic parameters Stress tensor
Structure optimization Molecular dynamics simulation	Stable structure Time evolution of atomic geometry
Vibration analysis	Vibration frequency, vibration mode
Positron lifetime calculation	Positron lifetime
STM image analysis	STM image (topographic and differential)
Chemical reaction analysis	Chemical reaction path, activation energy

The features of PHASE are summarized in the following.

Calculation scheme		
	First-principles calculation	Without any parameters fitted to experiments, this program has a reasonably high accuracy in predicting material properties even for unsynthesized materials; this enables users to do "material design." Owing to the use of hybrid functionals, more accurate predictions are feasible.
	Density functional theory	This program is based on the DFT, which is widely used in the field of materials science and is known to be highly predictable. LDA and GGA are featured in the program.
	Pseudopotential	Ion cores are treated using the pseudopotential scheme, which enables users to perform high-accuracy calculations.
Calculation functions		
	Physical properties	A wide variety of physical properties can be computed and compared with those obtained experimentally.
	Structural analyses	Atomic-geometry optimization, molecular dynamics simulation, and reaction-path analysis can be performed.
	Large-scale calculations	Parallel calculations using MPI and OpenMP can be used to perform calculations using hundreds of thousands of computer cores.
User friendliness		

	Input parameter file	An input parameter file consists of blocks and tags so that physical meanings of parameters can be easily understood by users. Users can set a wide variety of parameters to meet computational goals. Since default values are set for most parameters, the minimum size of an input parameter file can be small.
	Tools	This PHASE package bundles useful tools that help users draw a band structure, DOS, charge density distribution, etc.
	Machine architecture	A wide variety of platforms are available from Windows PCs to massively parallel supercomputers.
	User interface	A GUI (PHASE Viewer) is available to execute the PHASE program, edit input/output data, and visualize calculation results.

1.2.2 Contents of program package PHASE

The program package PHASE consists of the following programs and tools.

Program package PHASE		Overview
Program	phase	This is the main program in this program package. By using this, users can perform electronic structure calculations and molecular dynamics simulations. From a converged charge density, users can calculate the DOS, band structures, etc.
	ekcal	This is a subsidiary program that enables users to calculate the DOS and band structure for many $\boldsymbol{k}$-points. Some script files (tools) are available to execute this program.
Tool (script file)	band_kpoint.pl	This is a Perl script for generating a $\boldsymbol{k}$-point file to calculate a band structure.
	dos.pl	This is a Perl script for generating an EPS file for the DOS.
	band.pl	This is a Perl script for generating an EPS file for a band structure.

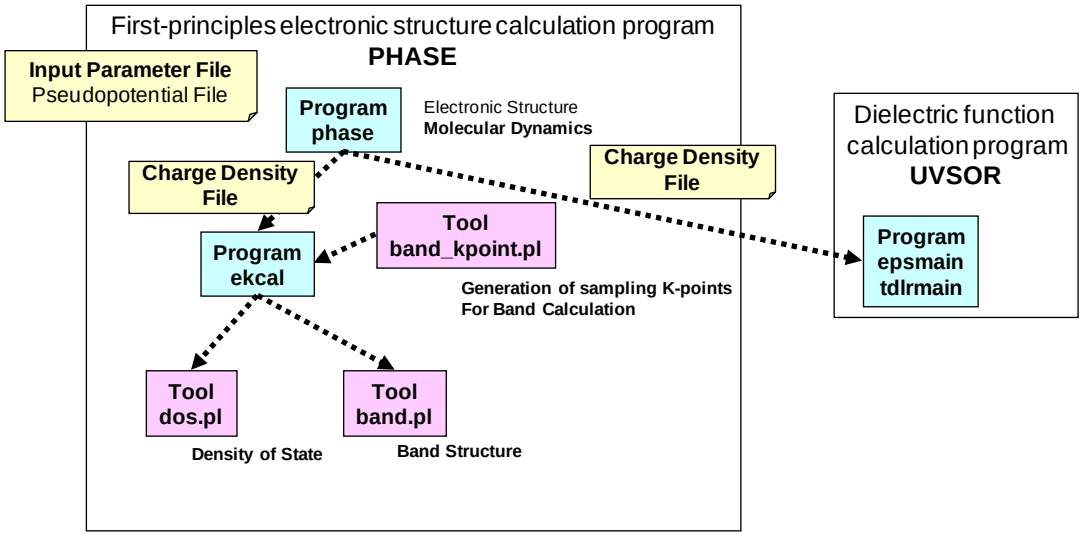


Figure 1.2 Contents of program package PHASE. UVSOR is also included in the figure.

1.2.3 Platforms to use PHASE

Since PHASE is programmed in Fortran90 and C, Fortran90 and C compilers are necessary. Those compilers are usually available in facilities like supercomputer centers in universities. MPI libraries are necessary to perform parallel calculations.

To use PHASE, the following libraries, compilers, and other software are necessary.

- Fortran90 compiler and C compiler (required)
- MPI libraries (optional), which are necessary to execute parallel calculations
- LAPACK and BLAS (optional)
- FFTW (optional)
- Perl (optional), which is necessary to use PHASE TOOLS
- Gnuplot (optional), which is necessary to use PHASE TOOLS

The binary "phase.exe," which is available on Windows PC, is included in the package. This enables users to use PHASE without compiling the program. However, since this executable is not parallelized, it is difficult to perform a large-scale calculation, because it takes significant time or it may fail owing to memory limitations. If it is necessary to use PHASE on parallel machines, you must compile the program yourself.

Note that the description in this manual is based on a platform running the Linux/Unix operating system. On other operating systems, commands, and messages may differ from those in this manual.

1.3 Outline of this manual

This manual consists of the following chapters.

Chap. 1	Introduction
	This chapter contains a brief introduction to the PHASE-SYSTEM and the PHASE program package.
Chap. 2	Directions for the basic use of PHASE
	This chapter gives directions for the basic use of PHASE, giving users an overview of how to use PHASE.
Chap. 3	Input parameter file: nfinp.data (F_INP file)
	This chapter explains all the parameters available in "nfinp.data." For many of the parameters, since default values have been set, users do not need to set all parameters. For the advanced use of PHASE, this chapter should be helpful.
Chap. 4	Examples of basic functions
	This chapter provides some examples of the basic functions of PHASE. This chapter is also available as a tutorial for PHASE. For each parameter, refer to the corresponding item in Chapter 3.
Chap. 5	Advanced analytical functions
	This chapter describes advanced analytical functions of PHASE.
Chap. 6	PAW method
	The projector augmented wave (PAW) method is available in PHASE. This chapter shows how to use PAW and discusses the functions that are available for PAW.
Chap. 7	Miscellaneous
	This chapter is devoted to supplemental material. If necessary, see this chapter.

Users who read this manual for the first time are encouraged to read Chapter 2 first and then read Chapter 4. Chapter 3 is not useful for beginners. Chapters 5–7 are for users who need specialized tools.

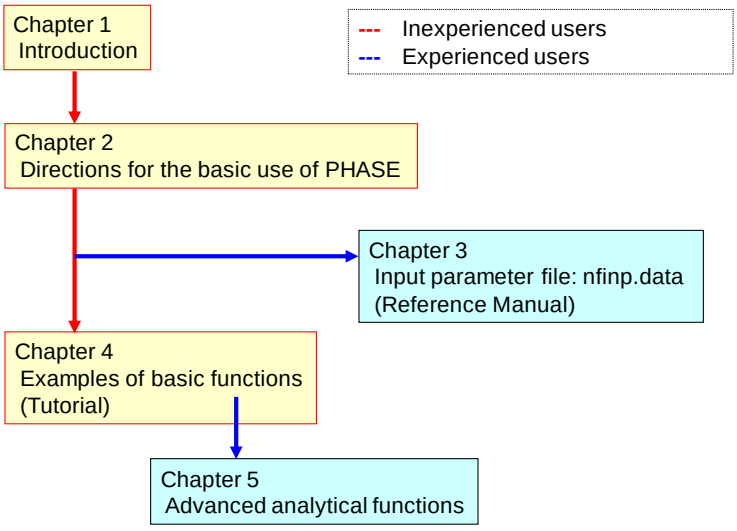


Figure 1.3 Outline of this manual

1.4 Upgrade history of PHASE

version 8.00 2009/06 release	・Constrained MD calculation was implemented. ・ Structure optimization and MD calculation within the DFT+U method was implemented.
version 8.01 2010/03 release	・Computational speed was improved by using BLAS routines.
version 9.00 2010/06 release	・Computational speed was improved by using BLAS routines and cache tuning. ・van der Waals interaction calculation was implemented. ・Free-energy calculation was implemented. ・Band calculation by using the DFT+U method was implemented. ・Hybrid functional was implemented.
version 10.00 2011/06 release	・Efficiency of SCF convergence was improved. ・PAW-type pseudopotential was implemented. ・Metadynamics methods were implemented. ・SCF calculation using van der Waals DFT was implemented. ・BFGS method was implemented for structure optimization. ・New script was added to PHASE TOOLS. ・Bugs related to reading pseudopotential files were fixed. Note that this change causes differences in the total energy compared to values obtained from the previous version.
version 10.01 2011/08 release	・Efficiency of SCF convergence for a system with spin was improved. ・Bug related to GGA was fixed. Note that this change causes differences in the total energy compared to values obtained from the previous version.
version 11.00 2012/06 release	・New wave function solvers were implemented. ・Hybrid functional calculation was improved. Treatment for ultrasoft pseudopotentials, reduced k-points calculations, etc. ・Continuation of optimization calculation due to the GDIIS or BFGS method was implemented. ・Computational speed for the calculation of the DOS using ultrasoft pseudopotential was improved. ・Writing the DOS and charge density during optimization calculation or molecular dynamics calculation was implemented. ・Some bugs were fixed. ・3-axis-parallelized version was released in which G-point-parallelization was implemented.
PHASE/0 2014 2014/04 release	・Automatic method for wave function solver and charge mixing was implemented. ・Prediction method of wave function solver and charge mixing during optimization calculation or molecular dynamics calculation was implemented. ・Time-dependent density functional theory (TDDFT) calculation were implemented. ・The interface of ESM was implemented. ・Optimization of unitcell was implemented. ・Work function calculation was implemented. ・Wave function solvers were improved. ・Hybrid functional calculation was improved.

	・Calculation of vdW interaction was improved. ・noncollinear calculation and spin orbit coupling calculation were implemented. ・Optimization calculation was improved. (CG method) ・Phonon calculation was improved. ・Real space calculation of nonlocal potential was implemented. ・UVSOL was integrated. ・3-axis-parallelized version was improved. ・Some bugs were fixed.

2. Directions for the basic use of PHASE

In this section, the directions for the basic use of PHASE are described. Since the main purpose of this section is to show the procedures for using PHASE, detailed directions for each calculation function are sometimes omitted. If detailed explanations are necessary, see Chapter 3 or later chapters.

Before reading this section, the installation of PHASE on your computer system is recommended; see the installation manual. Reading the PHASE tutorial is also recommended.

2.1 Outline of the calculation procedures of PHASE

The outline of the calculation procedure for using PHASE is as follows.

1. Prepare input files
2. Execute PHASE
3. Check the progress of the calculation
4. Analyze calculation results and/or visualize results.

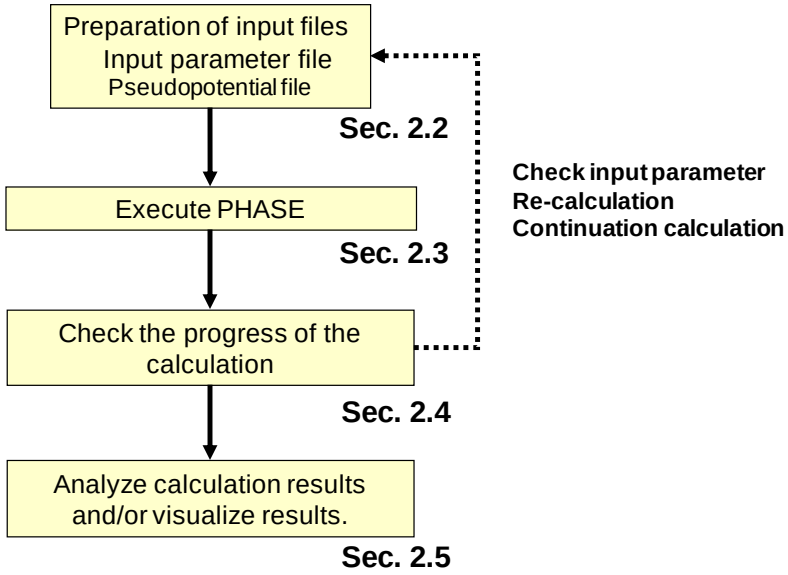


Figure 2.1 Outline of the calculation procedure of PHASE

2.2 Preparation of input files

2.2.1 Minimum set of input files

The minimum set of input files for executing PHASE consists of an input parameter file and pseudopotential files. These files must reside in the execution directory of the computer system.

Users can use "file_names.data" to change a file name from the default to a user-defined one and to put those files in a directory other than the execution one.

Input file

File	Brief overview
Input parameter file	This file specifies a model structure (e.g., atomic positions), calculation conditions (e.g., methods), and so on. The default name of this file is "nfinp.data," but by using "file_names.data," the name can be changed. An example involving many default parameters is introduced in section 2.2.2. For a detailed explanation of all parameters, see Chapter 3. Chapter 4 is devoted to examples that help users learn how to setup "nfinp.data." Chapter 5 is devoted to the application functions.
Pseudopotential file	To use PHASE, pseudopotential files for the elements identified in "nfinp.data" are necessary. For a detailed explanation, see section 2.2.3. The default names of pseudopotential files are "pot.01," "pot.02"… These names can be changed by using "file_names.data." Pseudopotential files can be downloaded through the website of PHASE, or they can be generated using CIAO codes. The maximum number of elements in a calculation is 16.

File-names setting file

file_names.data	This file is used to set the file names used in PHASE calculations. Since all files used in PHASE have default names, it is not always necessary to use this file. By using this file, users can change (i) file names, and (ii) directories in which those files are contained, except for this file itself. For detailed explanations, see Section 2.2.4.

2.2.2 Input parameter file: nfinp.data (simplified version)

The input parameter file "nfinp.data" specifies the model structure you want to calculate, calculation method you want to use, etc. For a detailed explanation of each parameter, see Chapter 3. For many parameters, default values are available. Therefore, it is not necessary for users to set all the parameters. In the following, a typical example of a parameter file is described.

2.2.2.1 Example of an input parameter file

An input parameter file consists of hierarchical blocks and tags (keywords). Each block is indicated by a block name and delimited by curly brackets {}. Parameters are usually specified in the format '`tag_keyword = value`'.

The following is an input parameter file for the calculation of a diamond-structure Si crystal in which two Si atoms are included in the unit cell.

```
control{
  condition = initial
  cpumax = 86400 sec
  max_iteration = 10000
}

accuracy{
  cutoff_wf = 25.0 rydberg
  cutoff_cd = 100.0 rydberg
  num_bands = 8
  ksampling{
    method = monk
    mesh{
      nx = 10
      ny = 10
      nz = 10
    }
  }
  initial_wavefunctions = atomic_orbitals
  initial_charge_density = atomic_charge_density
  scf_convergence{
    delta_total_energy = 1e-10
    succession = 3
  }
  force_convergence{
    max_force = 0.001 hartree/bohr
  }
}

structure{
  element_list{
    #tag    element    atomicnumber
          Si    14
  }
  unit_cell{
    #units angstrom
    a_vector = 0 2.732299538 2.732299538
    b_vector = 2.732299538 0 2.732299538
    c_vector = 2.732299538 2.732299538 0
  }
  unit_cell_type = bravais
```

```
  atom_list{
    atoms{
      #tag    element    rx    ry    rz    imove
            Si    0.125  0.125  0.125     0
            Si   -0.125 -0.125 -0.125     0
    }
    coordinate_system = internal
  }
}

wavefunction_solver{
       solvers{
          #tag    sol    till_n  prec cmix submat
                davidson    1    on   1    on
                rmm3       -1    on   1    on
       }
       rmm{
           edelta_change_to_rmm=5e-5
       }
}

charge_mixing{
       mixing_methods{
       #tag no   method    rmxs   rmxe   istr  prec nbmix
            1   pulay   0.40   0.40   3     on   15
       }
}

Postprocessing{
  dos{
    sw_dos = ON
    deltaE = 1.e-4 hartree
  }
  charge{
    sw_charge_rspace    = ON
    filetype = cube  !{cube|density_only}
    title  = "This is a title line for the bulk Si"
  }
}
```

The following blocks are available for most of the above blocks.

Block name	Contents
control	Setting for calculation conditions
accuracy	Setting for calculation accuracy
structure	Setting for atomic geometry
wavefunction_solver	Setting for wavefunction solver
charge_mixing	Setting for charge-mixing method
structure_evolution	Setting for optimization or molecular dynamics simulation
postprocessing	Setting for postprocess analysis
printlevel	Setting for log output

In the following sections, input parameters available in each of these blocks are described.

2.2.2.2 Control block

In the '`control`' block, users can set parameters that control the entire calculation process and can specify general options.

```
control{
  condition = initial
  cpumax = 86400 sec
  max_iteration = 10000
}
```

condition	This tag specifies a calculation condition: '`initial`' means that the user starts the calculation from scratch, and '`continuation`' means that the user continues the calculation from a previous one. This mode is necessary when a previous calculation does not reach completion.
cpumax	Upper limit on CPU time (defaults to 86,400 s). Available units are {s, min, h, day}.
max_iteration	Maximum number of SCF iterations (defaults to 10,000)

2.2.2.3 Accuracy block

In the '`accuracy`' block, users can set parameters related to calculational accuracy.

```
accuracy{
  cutoff_wf = 25.0 rydberg
  cutoff_cd = 100.0 rydberg
  num_bands = 8
  ksampling{
    method = monk
    mesh{
      nx = 10
      ny = 10
      nz = 10
    }
  }
  initial_wavefunctions = atomic_orbitals
  initial_charge_density = atomic_charge_density
  scf_convergence{
    delta_total_energy = 1e-10
    succession = 3
  }
  force_convergence{
    max_force = 0.001 hartree/bohr
  }
}
```

cutoff_wf	Cut-off energy for wavefunction expansion.
cutoff_cd	Cut-off energy for charge-density expansion.
num_bands	Number of bands.
ksampling block	This is a sub-block for setting $\boldsymbol{k}$-point sampling.
method	Specify $\boldsymbol{k}$-point sampling method; '`monk`' means the Monkhorst–Pack method [5].
mesh	Number of partitions for the division of the Brillouin Zone.

initial_wavefunctions	Initial wavefunction generation method; 'atomic_charge_density' means that the initial wavefunction is calculated from charge-density data in the pseudopotential files.
scf_convergence block	This sub-block specifies convergence criteria for an SCF calculation.
delta_total_energy	Convergence criteria for an SCF calculation. If the difference between the current total energy and the total energy of the previous SCF iteration is smaller than the specified value, the convergence criterion is satisfied.
succession	SCF iterations are terminated if the energy difference is smaller than the criterion 'delta_total_energy" ***n***-times in succession. The variable 'succession' specifies the value of ***n***.
force_convergence block	This sub-block specifies the convergence criterion for structure optimization.
max_force	Convergence criterion for structure optimization. When the maximum value among forces acting on all atoms becomes smaller than this value, an optimized structure is obtained.

2.2.2.4 Structure block

In the 'structure' block, users can set parameters related to the atomic structure.

```
structure{
  element_list{
    #tag    element    atomicnumber
           Si    14
  }
  unit_cell{
    #units angstrom
    a_vector = 0 2.732299538 2.732299538
    b_vector = 2.732299538 0 2.732299538
    c_vector = 2.732299538 2.732299538 0
  }
  unit_cell_type = bravais
  atom_list{
    atoms{
      #tag    element    rx    ry    rz    imove
            Si    0.125  0.125  0.125     0
            Si   -0.125 -0.125 -0.125     0
    }
    coordinate_system = internal
  }
}
```

element_list block	This sub-block specifies the elements used in the calculation. In this case, Si is specified as an element and '14' is its atomic number.
unit_cell block	This sub-block specifies the unit cell. "#units angstrom" specifies the unit of angstrom. "a_vector," "b_vector," and "c_vector" are lattice vectors.
atom_list block	This sub-block specifies the coordinates and elements of atoms. In this case, two Si atoms are set, and their coordinates are "0.125 0.125 0.125"

and "-0.125 -0.125 -0.125."

coordinate_system — This tag specifies a coordinate type. "Internal" means the internal coordinate based on the lattice vectors.

2.2.2.5 Wavefunction_solver block

In the "wavefunction_solver" block, users can set parameters related to solvers of wavefunctions.

```
wavefunction_solver{
       solvers{
          #tag    sol    till_n  prec cmix submat
                davidson    1    on   1    on
                rmm3       -1    on   1    on
       }
       rmm{
           edelta_change_to_rmm=5e-5
       }
}
```

Solvers block — This sub-block specifies which solver is used to calculate wave functions. In this case, "davidson" [6] is used first, and then "rmm3" is used; "rmm3" is a residual minimization method (RMM) solver [7].

Rmm block — This sub-block is used to set parameters related to RMM solvers.

edelta_change_to_rmm — This tag indicates the criterion for changing the solver. If the difference between the current total energy and the total energy of the previous SCF iteration is smaller than the specified value, the solver is changed. In this case, if the energy difference becomes smaller then 5e-5, the solver is changed from "davidson" to "rmm3."

2.2.2.6 Charge_mixing block

In the "charge_mixing" block, users can set parameters related to charge mixing during an SCF calculation.

```
charge_mixing{
       mixing_methods{
       #tag no   method    rmxs   rmxe   istr  prec nbmix
            1   pulay   0.40   0.40   3     on   15
       }
}
```

mixing_methods block — This sub-block specifies the charge-mixing method. In this case, the Pulay method [8] is used, and the mixing ratio is "0.40."

2.2.2.7 Postprocessing block

In the "postprocessing" block, users can set parameters related to postprocess analysis.

```
Postprocessing{
  dos{
    sw_dos = ON
    deltaE = 1.e-4 hartree
  }
  charge{
    sw_charge_rspace = ON
    filetype = cube
    title  = "This is a title line for the bulk Si"
  }
}
```

dos block	This sub-block specifies parameters related to the DOS.
sw_dos	"ON" means that the DOS is calculated.
deltaE	This indicates the energy-mesh width for the DOS.
Charge block	This sub-block specifies parameters related to the output of the charge density distribution.
sw_charge_rspace	"ON" means that the charge density distribution is calculated.
filetype	This indicates a file type for the charge density distribution; "cube" means the "Gaussian cube" style [9].
title	This specifies the first line of a "Gaussian cube" style file. In this case, "This is a title line for the bulk Si" is the output as the first line of the file.

2.2.2.8 Minimum set of input parameters

In the example above, many parameters are set explicitly. However, users do not need to change all those parameters to calculate other materials because many of the parameters also apply to other materials.

It is necessary for users to set parameters related to cut-off energies, number of bands, ***k***-points, atomic structure, and unit cell. Those parameters are shaded in the example below. Users can perform PHASE calculations for most materials by just changing those few parameters.

Note that if users want an efficient calculation, it may be necessary to change other parameters, such as "wavefunction_solver" and "charge_mixing."

```
control{
  condition = initial
  cpumax = 86400 sec
  max_iteration = 10000
}

accuracy{
  cutoff_wf = 25.0 rydberg
  cutoff_cd = 100.0 rydberg
  num_bands = 8
  ksampling{
    method = monk
    mesh{
      nx = 10
      ny = 10
      nz = 10
    }
  }
```

```
  initial_wavefunctions = atomic_orbitals
  initial_charge_density = atomic_charge_density
  scf_convergence{
    delta_total_energy = 1e-10
    succession = 3
  }
  force_convergence{
    max_force = 0.001 hartree/bohr
  }
}

structure{
  element_list{
    #tag    element    atomicnumber
          Si    14
  }
  unit_cell{
    #units angstrom
    a_vector = 0 2.732299538 2.732299538
    b_vector = 2.732299538 0 2.732299538
    c_vector = 2.732299538 2.732299538 0
  }
  unit_cell_type = bravais
  atom_list{
    atoms{
      #tag    element    rx    ry    rz    imove
            Si    0.125  0.125  0.125     0
            Si   -0.125 -0.125 -0.125     0
    }
    coordinate_system = internal
  }
}

wavefunction_solver{
       solvers{
          #tag    sol    till_n  prec cmix submat
                davidson    1    on   1    on
                rmm3       -1    on   1    on
       }
       rmm{
           edelta_change_to_rmm=5e-5
       }
}

charge_mixing{
       mixing_methods{
       #tag no   method    rmxs   rmxe   istr  prec nbmix
            1   pulay   0.40   0.40   3     on   15
       }
}

Postprocessing{
  dos{
    sw_dos = ON
    deltaE = 1.e-4 hartree
  }
  charge{
    sw_charge_rspace    = ON
    filetype = cube  !{cube|density_only}
    title  = "This is a title line for the bulk Si"
```

```
  }
}
```

For most parameters, default values are pre-set. Then, even if users omit such parameters in the parameter file, they can still calculate with PHASE.

Note that the input parameter file shown above and the one shown below give the same energy value if the SCF calculations converge; however, the number of SCF iterations needed to reach convergence may differ.

```
accuracy{
  cutoff_wf = 25.0 rydberg
  cutoff_cd = 100.0 rydberg
  num_bands = 8
  ksampling{
    mesh{
      nx = 10
      ny = 10
      nz = 10
    }
  }
}
structure{
  element_list{
    #tag    element    atomicnumber
           Si    14
  }
  unit_cell{
    #units angstrom
    a_vector = 0 2.732299538 2.732299538
    b_vector = 2.732299538 0 2.732299538
    c_vector = 2.732299538 2.732299538 0
  }
  atom_list{
    atoms{
      #tag    element    rx    ry    rz    imove
             Si    0.125  0.125  0.125     0
             Si   -0.125 -0.125 -0.125     0
    }
  }
}
```

2.2.3 Pseudopotential files

Pseudopotential files must be prepared for all elements used in a calculation. For example, in a calculation for H_2O, pseudopotential files for O and H atoms are necessary. Pseudopotential files can be downloaded from the download page of the RISS project, which is the same web page used for the PHASE download. Alternatively, pseudopotential files can be generated using CIAO. For the CIAO code, see the manual book for CIAO.

2.2.3.1 Types of pseudopotentials

Pseudopotential files for PHASE are classified into two types. One is a frozen core type, and the other is a PAW type [4].

Frozen core type	Core electrons and the atom core are treated together as an ion core and are fixed at the same states as those in an isolated atom. With these pseudopotentials, PHASE calculates electronic states by considering only valence electrons. These types of pseudopotentials are further classified into two types: one is norm-conserving [2] and the other is ultrasoft [3].
PAW type	In PAW-type pseudopotentials, electronic states are calculated by partly considering core electron states.

Note that both types of pseudopotential cannot be used in the same calculation. Determine which type of pseudopotentials is to be used before starting a calculation.

2.2.3.2 How to get pseudopotential files?

Pseudopotential files can be downloaded from the website "http://www.ciss.iis.u-tokyo.ac.jp/dl/" operated by the Center for Research on Innovative Simulation Software, Institute of Industrial Science, the University of Tokyo. All elements in the periodic table are available.

A pseudopotential file name is created by the following naming rule.

Element_Exchange Correlation term method_Pseudopotential type_identification number.pp

For example, "Si_ldapw91_nc_01.pp" is a pseudopotential file for an element Si (silicon) with its exchange-correlation term being "ldapw91," its type being "nc," and its identification number being "01."
If the pseudopotential type is "ultrasoft," "us" is used instead of "nc," for "PAW," "paw" is used.

Element	Si (silicon)
Exchange-correlation term method	ldapw91 [10]
Pseudopotential type	Norm-conserving pseudopotential. "nc" is the abbreviation for "norm conserving"
Identification number	Sequential serial number for the identification.

2.2.3.3 How to indicate pseudopotential files?

The default names of pseudopotential files are "pot.01," "pot.02,"… for the elements identified in the input

parameter file; use the same order as in the file.

By using "file_names.data," users can freely set pseudopotential file names and directories in which pseudopotential files are stored.

2.2.4 file_names.data

The file "file_names.data" is used for setting file names of an input parameter file, pseudopotential files, etc. Users can use PHASE without this file. In this case, default names will be used for all files.

By using this file, users can freely change (i) file names and (ii) directories where files are stored. However, the file name of "file_names.data" itself cannot be changed. This file must be placed in the execution directory.

The format of "file_names.data" is as follows.

```
&fnames
File_keyword = 'file_name(and path for the file)'
…
…
/
```

Note that "/" is necessary on the last line. The following is an example.

```
&fnames
F_INP = './nfinp.data'
F_POT(1) = './Si_ggapbe_nc_01.pp'
F_POT(2) = './O_ggapbe_us_02.pp'
F_CHGT = './nfchgt.data'
F_CHR = './nfchr.cube'
/
```

For the path to a file, users may provide either an absolute path or a path relative to the execution directory.

"`F_POT(n)`" is used to set the pseudopotential file name for the *n*-th element indicated in an input parameter file. In the example, the pseudopotential file for the first element indicated in an input parameter file is "`Si_ggapbe_nc_01.pp`" and that for the second is "`O_ggapbe_us_02.pp`."

Available file_keywords are listed in Table 2.1.

Table 2.1 Files settable in "file_names.data"

File_keyword	program	Input/output	Default name	Overview
F_INP	phase ekcal	Input	nfinp.data	This file keyword is used to assign an input parameter file.
F_POT(n)	phase ekcal	Input	pot.01, pot.02, ・・・	These file keywords are used to assign pseudopotential files. Each element identified in an input file needs one pseudopotential file.
F_STOP	phase ekcal	Input	nfstop.data	This file is used to stop the PHASE execution at a certain SCF iteration number.
F_KPOINT	phase ekcal	Input	kpoint.data	This file is used to set k-point sampling. This file is available only when "file" is selected for k-point sampling in an input parameter file.
F_DYNM	phase	Output	nfdynm.data	This file contains atomic geometries and forces acting on atoms at each step during a geometry optimization calculation or an MD

				calculation.
F_ENF	phase	Output	nfefn.data	This file contains the total energy value and the maximum force value among those acting on all the atoms at each step during a geometry optimization calculation or an MD calculation.
F_CHR	phase	Output	nfchr.data	This file is an output file for PHASE. It contains the charge density distribution. The default file style is "Gaussian cube."
F_DOS	phase ekcal	Output	dos.data	This file contains the DOS.
F_ENERG	ekcal	Output	nfenergy.data	This file contains eigenvalues that result from a band calculation.
F_ZAJ	phase ekcal	Input/output	zaj.data	This file contains wavefunction data. In a continuation calculation, this file is used as an input file for wavefunctions. This is a binary file.
F_CHGT	phase ekcal	Input/output	nfchgt.data	This file contains charge-density data. In a continuation calculation, this file is used as an input file for charge density. This is a binary file.
F_CNTN	phase	Input/output	continue.data	This file contains some data needed in a continuation calculation.
F_CNTN_BIN	phase	Input/output	continue bin.data	This file contains some data needed in a continuation calculation. This is a binary file.
F_STATUS	phase ekcal	Output	jobstatus00x	In this file, the status of a calculation is recorded.

2.3 How to calculate with PHASE?

2.3.1 Execution of program PHASE

First, put an input parameter file and the pseudopotential files in the execution directory. If the user uses the file "file_names.data," put it in the same directory.

When performing a serial calculation (a calculation with one computer core), execute the PHASE executable as follows, where " .././phase_v1200/bin/" means the directory in which the PHASE executable has been placed.

```
% ../../phase_v1200/bin/phase
```

When performing a parallel calculation, execute the PHASE executable as follows. Here "mpirun" is used as a command for parallel calculations; this is the most common command. However, this command depends on the MPI library. For more details, see the manual for the computer system.

```
% mpirun -np NP ../../phase_v1200/bin/phase ne=NE nk=NK
```

Here, "NP" means the number of MPI processes, "NE" means the number for band parallelization, and "NK" means the number for ***k***-point parallelization. NP must be equal to NE * NK.

2.3.2 How to check the calculation status?

The SCF convergence progress is printed to a log file "output000." The total energy at each step during SCF convergence is printed on a line that starts with "TOTAL ENERGY FOR."

These lines can be found using the "grep" command as follows.

```
% grep TH output000
```

The output of this command is

```
TOTAL ENERGY FOR 1 -TH ITER= -30.856896066222 edel = -0.308569D+02 : SOLVER = MATDIAGON
TOTAL ENERGY FOR 2 -TH ITER= -31.552303846339 edel = -0.695408D+00 : SOLVER = DAVIDSON
TOTAL ENERGY FOR 3 -TH ITER= -31.585336745971 edel = -0.330329D-01 : SOLVER = DAVIDSON
TOTAL ENERGY FOR 4 -TH ITER= -31.587689791426 edel = -0.235305D-02 : SOLVER = SUBMAT + RMM3
TOTAL ENERGY FOR 5 -TH ITER= -31.587917474699 edel = -0.227683D-03 : SOLVER = SUBMAT + RMM3
TOTAL ENERGY FOR 6 -TH ITER= -31.587936742564 edel = -0.192679D-04 : SOLVER = SUBMAT + RMM3
TOTAL ENERGY FOR 7 -TH ITER= -31.587937115320 edel = -0.372756D-06 : SOLVER = SUBMAT + RMM3
..............
..............
```

The integer appearing after "FOR" identifies the SCF calculation iteration, and the value appearing after "ITER=" is the total energy at that iteration. Energy values are displayed in Hartree. Generally, these total energy values are negative. In the above example, the total energy values are about −31 Ht.

After "edel =," the energy difference between the current SCF iteration and the previous SCF iteration is displayed. If this energy difference becomes lower than the criterion "delta_total_energy" set in the input parameter file, the convergence criterion is satisfied.

After "SOLVER =," solver information at the iteration is shown. In the example above, at the first iteration, "MATDIAGON" was used, while at the second and third iterations, "DAVIDSON" was used. After the third iteration, "RMM3" was combined with "SUBMAT."

By checking the convergence progress, users can determine whether the convergence calculation was performed accurately.

2.3.3 Continuation calculation

In many cases, one PHASE calculation is not sufficient to complete a calculation because of limits on machine time. In such cases, users can continue a PHASE calculation by setting the "condition" tag in the input parameter file. The following is an example.

```
control{
  condition = continuation
}
```

In a continuation calculation, some output files from the previous calculation are necessary; thus, it is better to perform the continuation calculation in the same execution directory. If "automatic" is set instead of "initial" or "continuation," PHASE automatically sets this tag depending on which files exist in the execution directory.

2.3.4 Ekcal program for the calculation of the DOS and band structure

Ekcal is a subprogram of PHASE for calculating the DOS and band structure for many ***k***-points. A charge density distribution file is necessary to perform an ekcal calculation.

2.3.4.1 How to calculate the DOS by ekcal?

After the completion of a PHASE calculation, a charge-density file, whose default file name is "nfchgt.data," is created. This is an input file for the DOS calculation by the program ekcal.

Copy the file into the execution directory or set the path for the file by the key_word "F_CHG" in "file_names.data."

Edit the input parameter file. In the "control" block, set the condition tag, as follows.

```
control{
  condition = fixed_charge
}
```

In the "accuracy" block, set the "delta_eigenvalue" tag for the convergence of eigenvalues, as follows.

```
accuracy{
  ek_convergence{
    delta_eigenvalue = 1e-5
  }
}
```

Execute the program ekcal as the follows, where "phase_v1100/bin/" is a directory in which the ekcal executable is stored.

```
% ../../phase_v1100/bin/ekcal
```

2.3.4.2 How to calculate the band structure by ekcal?

After the completion of a PHASE calculation, a charge-density file, whose default file name is "nfchgt.data," is created. This is an input file for the DOS calculation by the program ekcal.

Copy the file into the execution directory or set the path for the file by the key_word "F_CHG" in "file_names.data."

A file "kpoint.data," which is a file for ***k***-point data, is necessary to calculate the band structure. A PHASE tool, "band_kpoint.pl" can be used to generate "kpoint.data." Make the file "bandkpt.in," which is an input file for "band_kpoint.pl," as follows.

```
0.04                                        spacing of k points data
-0.8333333  0.8333333  0.8333333
0.8333333  -0.8333333  0.8333333           reciprocal vectors
0.8333333  0.8333333  -0.8333333
3 2 1 4 # W                                 typical k points  n1 n2 n3 nd # Symbol
1 1 1 2 # L
0 0 0 1 # {/Symbol G}
1 1 0 2 # X
3 2 1 4 # W
5 3 0 8 # K
```

For indicating each typical ***k***-point, n1/nd, n2/nd, and n3/nd mean internal coordinates based on reciprocal vectors. For example, "3 2 1 4 # W" means the W point with their internal coordinates 3/4, 2/4, and 1/4 based on the reciprocal vectors.

Execute the tool "band_kpoint.pl," as follows, and "kpoint.data" will be generated.

```
% ../../phase_v1100/tools/bin/band_kpoint.pl  bandkpt.in
```

Edit the input parameter file. In the "control" block, set the condition tag, as follows.

```
control{
  condition = fixed_charge
}
```

In the "accuracy" block, set the "method" tag as "file" to read "kpoint.data" and set the "delta_eigenvalue" tag for the convergence of eigenvalues, as follows.

```
accuracy{
      ksampling{
            method = file
      }
      ek_convergence{
            delta_eigenvalue = 1e-5
      }
}
```

Execute the program ekcal as follows, where "phase_v1100/bin/" is the directory in which the ekcal executable is stored.

```
% ../../phase_v1100/bin/ekcal
```

2.4 How to check the completion of the calculation?

2.4.1 Status of the PHASE calculation, causes, and options

The status of a PHASE calculation and the causes and options associated with each status are listed below.

Status	Cause of the status	options
Successful completion SCF calculation converges. (or structure is optimized)	Energy difference between two consecutive iterations becomes smaller than the convergence criterion (`delta_total_energy`).	Analytical calculation
	In an optimization calculation, the maximum among the forces becomes smaller than the criterion (`max_force`).	
Successful completion SCF calculation does not converge (or structure is not optimized)	Number of SCF iterations reaches the maximum iteration number indicated by the "`max_iteration`" tag in the "`control`" block.	Continuation calculation
	Number of SCF iterations exceeds the value set in the file "nfstop.data." Users can stop a PHASE execution by using this file, even if it is on the process.	
	Elapsed time exceeds the time limit indicated by the tag "`cpumax`" in the "`control`" block.	
Abnormal termination	Possible causes are as follows: Failure in an input parameter file, Pseudopotential files do not exist, Trouble in the computer system, Bugs in the program.	Check files and re-execute the program

2.4.2 How to check successful completion or abnormal termination?

If a PHASE execution ends normally, text like the following is printed to a logfile (output000).

```
       ......
       ......
<<Total elapsed CPU Time until now =    81.69520 (sec.)>>
 closed filenumber =           31
 closed filenumber =           52
 closed filenumber =           53
 closed filenumber =           55
 closed filenumber =           42
 closed filenumber =           43
 closed filenumber =           44
 closed filenumber =           75
 closed filenumber =           65
 closed filenumber =           66
```

After the "`Total elapsed CPU Time until now =`," the calculation time is displayed.

If the last part of the logfile differs from this example, then the PHASE execution failed. In that case, a recalculation is necessary, but before recalculating, check the input parameter file, the execution command, all compile options, etc.

2.4.3 Check the convergence of an SCF calculation and structure optimization

If a PHASE execution ends normally, the calculation may still not have reached the desired completion. Users can find the status of a PHASE calculation by checking the file "continue.data," which is generated after a PHASE execution ends. The last part of this file looks like the following.

```
iteration, iteration_ionic, iteration_electronic
       11       1      11
 Ionic System
  (natm)
        2
  (pos)
  0.1250000000000000D+00  0.1250000000000000D+00  0.1250000000000000D+00
 -0.1250000000000000D+00 -0.1250000000000000D+00 -0.1250000000000000D+00
  (cps)
  0.1290824363824501D+01  0.1290824363824501D+01  0.1290824363824501D+01
 -0.1290824363824501D+01 -0.1290824363824501D+01 -0.1290824363824501D+01
  (cpd)
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  1))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  2))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  3))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
 forcmx_constraint_quench
  0.1000000000000000D+03
 Total Energy
 -0.7878566524513241D+01 -0.7878566524513241D+01
isolver
        5
convergence
        2
edelta_ontheway
  0.1000000000000000D-09
corecharge_cntnbin
      0
neg
        8
```

In the shaded area, "2" appears after "convergence." The "2" means that the SCF calculation has converged, and an optimized structure was obtained. If a different number appears there, then a continuation calculation is necessary.

2.4.4 Calculation status during a calculation (logfile: output000 and jobstatus000)

The file "output000" contains a log of a PHASE execution. The string "000" indicates the number of executions; its value increases as 001, 002, etc., depending on how many times the calculation has been executed in the directory.

This file holds information about the calculation and about physical quantities. In the following, useful parts are explained.

2.4.4.1 Sampling *k*-points

The *k*-points used in a calculation are difficult to know from an input parameter file. Users can find *k*-point data in the logfile "output000." To do so, find the string "kv3" in the logfile.

```
 !kp kv3 =      8 nspin =      1
```

In this case the number of *k*-points was 8. The "1" after "nspin =" means that spin freedom was not considered. If "2" is here, then spin freedom was considered.

2.4.4.2 Total energy

Total energies are printed in a logfile as follows.

```
 TOTAL ENERGY FOR     3 -TH ITER=   -687.253021587082  edel =  -0.215950D+02 : SOLVER = DAVIDSON
 KI=    294.118626755617 HA=   4820.263454482710 XC=   -686.596385560733 LO=  -8452.905431759591
 NL=   -349.620400894588 EW=   3182.022578317359 PC=    505.464805336868 EN=     -0.000268264724
 PHYSICALLY CORRECT ENERGY =    -687.252887454720
```

The value of the total energy is printed after "TOTAL ENERGY FOR …ITER=," and the energy difference between the current iteration and the previous one is printed after "edel =." Following this line, contributions to the total energy are displayed: "KI" means kinetic energy, "HA" Hartree energy, "XC" exchange-correlation energy, "LO" local potential energy, "NL" nonlocal potential energy, "EW" Ewald energy, "PC" partial core correction energy, and "EN" entropy. The summation of all these terms is the total energy.

After "PHYSICALLY CORRECT ENERGY," a corrected total energy is printed for the case of smearing electron occupations.

2.4.4.3 Spin freedom

When the calculation considers spin freedom, majority and minority spin states are shown at each SCF iteration, as follows.

```
 !OLD total charge (UP, DOWN, SUM) =     4.53623488 (+)    3.46376512 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.64907433 (+)    3.35092567 (=)    8.00000000
```

The line starting with "!OLD" shows spin information from the previous iteration and that starting with "!NEW" shows spin information for the current iteration.

2.4.4.4 Eigenvalues and their occupations

Eigenvalues for each *k*-point are printed to the logfile just before the completion of execution. Note that this output is only for the last iteration; results for eigenvalues at previous iterations are not printed.

```
 EFermi =       0.24579615
 ======  Energy Eigen Values ======
     1      0.00000000      0.00000000      0.00000000
    -0.19655861     -0.04839227     -0.04839227     -0.04839227     -0.04839227
    -0.04839227     -0.04839227      0.12584623      0.12584623      0.12584623
     0.12584623      0.12584623      0.12584623      0.23389619      0.23389619
     0.23389619      0.26196708      0.26196708      0.26196708      0.26196708
     2      0.25000000      0.00000000      0.00000000
    -0.18998394     -0.11270106     -0.04555873     -0.04555873     -0.04555873
    -0.04555873      0.02675145      0.10512408      0.10512408      0.10512408
     0.10512408      0.13505063      0.13505063      0.18575457      0.20251681
     0.20251681      0.25769611      0.29275976      0.30811466      0.30811466
     3      0.50000000      0.00000000      0.00000000
    -0.16102016     -0.16102016     -0.04095243     -0.04095243     -0.04095243
    -0.04095243      0.08874423      0.08874423      0.08874423      0.08874423
     0.10781439      0.10781439      0.16184290      0.16184290      0.16184290
     0.16184290      0.27543069      0.27543069      0.35154734      0.35154734
     4      0.75000000      0.00000000      0.00000000
    -0.18998394     -0.11270106     -0.04555873     -0.04555873     -0.04555873
    -0.04555873      0.02675145      0.10512408      0.10512408      0.10512408
     0.10512408      0.13505063      0.13505063      0.18575457      0.20251681
     0.20251681      0.25769611      0.29275976      0.30811466      0.30811466
     5      0.00000000      0.25000000      0.00000000
    -0.18998394     -0.11270106     -0.04555873     -0.04555873     -0.04555873
    -0.04555873      0.02675145      0.10512408      0.10512408      0.10512408
```

```
      0.10512408      0.13505063      0.13505063      0.18575457      0.20251681
      0.20251681      0.25769611      0.29275976      0.30811466      0.30811466
      ....................................................................
      ....................................................................
      ....................................................................
```

Following the output for eigenvalues, the occupations for each ***k***-point are displayed, as follows.

```
 ======  Occupations ======
     1      0.00000000      0.00000000      0.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      0.00000000      0.00000000      0.00000000      0.00000000
     2      0.25000000      0.00000000      0.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      0.00000000      0.00000000      0.00000000      0.00000000
     3      0.50000000      0.00000000      0.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      1.00000000      1.00000000      1.00000000      1.00000000
      1.00000000      0.00000000      0.00000000      0.00000000      0.00000000
```

Occupations are usually between 0 and 1. When spin freedom is not considered, "`1.0`" means that two electrons occupy the state. Owing to system symmetries, reduction of ***k***-points may occur. In that case, occupations may vary depending on the reduction. This happens for bulk systems with many ***k***-points.

2.4.4.5 Elapsed time for each SCF calculation

If a "`printlevel`" tag in an input parameter file is set to 1 or more than 1, the elapsed time for that iteration is printed to the logfile as follows.

```
 << CPU Time Consumption -- TOP   9 Subroutines (    2) >>
  no  id           subroutine name          time(sec)  r(%)     count   no(2)
   1  20  evolve_WFs_in_subspace (davidson  115.74820 71.17       8       1
   2  13                 m_ES_Vnonlocal_W    10.78620  6.63       8       2
   3   8                    betar_dot_WFs     7.33490  4.51      14       3
   4  16                    m_CD_softpart     2.53880  1.56       1       4
   5   7               m_XC_cal_potential     0.97520  0.60       2       5
   6  17                    m_CD_hardpart     0.28100  0.17       1       6
   7  10            m_ES_Vlocal_in_Rspace     0.02990  0.02       1       7
   8  19                  m_CD_mix_pulay     0.00670  0.00       1       8
   9  18           m_CD_convergence_check     0.00230  0.00       1       9
 Total cputime of (    2 )-th iteration     162.64080 /   221.651 (sec.)
```

After "…`iteration`," the elapsed time for that iteration and the total elapsed time from the beginning appear. If the difference between the current iteration and the previous iteration is smaller than 5% of the elapsed time, this information is not displayed.

2.4.4.6 Progress situation of the calculation (jobstatus000)

In the file "jobstatus000," the progress situation of the calculation is recorded. The number "000" on file names depends on how many times the calculation has been executed in the directory. The record is as follows.

```
 status       =      FINISHED
 iteration    =           674
 iter_ionic   =            21
 iter_elec    =            23
 elapsed_time =     51648.7582
```

status	FINISHED (completion), ITERATIVE (in progress), START (initialization)
iteration	Number of total SCF calculation iterations
iter_ionic	Number of MD/optimization steps

iter_elec	Number of SCF iterations for the current MD/optimization step
elapsed_time	Total elapsed time

2.5 Analysis of calculation results and visualization

2.5.1 Total energy and force (recorded in nfefn.data)

In the file "nfefn.data" (or a file indicated by "F_ENF" in "file_names.data"), the total energy of the system and the maximum among forces acting on atoms at each MD/optimization step are recorded. In case of an MD calculation, the kinetic energy and the conserved quantity are also recorded.

The output content in "nfefn.data" for an MD simulation differs from that for structure optimization. In the following, both types of "nfefn.data" are shown separately.

2.5.1.1 Structure optimization

Output content of "nfefn.data" for an optimization calculation:

```
 iter_ion, iter_total, etotal, forcmx
     1      24     -108.4397629733        0.0086160410
     2      40     -108.4401764388        0.0076051917
     3      56     -108.4405310817        0.0068758156
     4      73     -108.4410640011        0.0065717365
     5      94     -108.4414256084        0.0099533097
     6     113     -108.4414317178        0.0094159378
                  ........
                  ........
                  ........
```

The meaning of each column is as follows.

iter_ion	Number of optimization steps
iter_total	Number of total SCF iterations.
etotal	Total energy in Hartree units.
forcmx	Maximum among the forces acting on all the atoms. The unit is hartree/bohr^3. The calculation continues until this value becomes smaller than the value for "max_force" set in the input parameter file.

2.5.1.2 MD simulation

Output content of "nfefn.data" for an MD simulation:

```
      iter_ion, iter_total, etotal, ekina, econst, forcmx
     1      18    -7.8953179624     0.0000000000    -7.8953179624     0.0186964345
     2      30    -7.8953851218     0.0000665502    -7.8953185716     0.0183575425
     3      43    -7.8955768901     0.0002565396    -7.8953203505     0.0173392067
                          ........
                          ........
                          ........
```

In addition to the structure optimization case, the following two columns are recorded.

ekina	Kinetic energy of atoms
econst	Conserved quantity. For an NVE calculation, this corresponds to the total energy, including the atomic kinetic energy. For an NVT calculation, this value corresponds to the total energy, including the energy of the heat bath.

2.5.2 Atomic geometry (recorded in nfdynm.data)

In the file "nfdynm.data" (or a file indicated by "F_DYNM" in "file_names.data"), coordinates, and forces for all the atoms at each MD/optimization step are recorded.

The content of "nfdynm.data" is as follows. Note that atomic units are used in this file regardless of the units specified in the input parameter file.

```
#
#   a_vector =         9.2863024980        0.0000000000        0.0000000000
#   b_vector =        -4.6431512490        8.0421738710        0.0000000000                (a)
#   c_vector =         0.0000000000        0.0000000000       10.2158587136
#   ntyp =        2 natm =        9                                                       (b)
# (natm->type)     1    1    1    1    1    1    2    2    2                            (c)
# (speciesname)     1 :   O                                                             (d)
# (speciesname)     2 :   Si
#
 cps and forc at (iter_ion, iter_total =     1      24 )                                 (e)
    1    3.161057370    1.169332082    1.214972077   -0.004058   -0.005565   -0.004966     (f)
    2    6.693102525    2.152889944    4.620258315    0.006945   -0.001028   -0.004994
    3    4.075293851    4.719951845    8.025544553   -0.002872    0.006394   -0.004796
    4   -1.482093879    6.872841789    5.595600399   -0.004362    0.005502    0.004993
    5   -0.567857398    3.322222026    9.000886637   -0.002792   -0.006296    0.004965
    6    2.049951276    5.889283925    2.190314161    0.006974    0.000708    0.004795
    7    4.921740324    0.000000000    3.405282833    0.001436    0.000122    0.000068
    8   -2.460870162    4.262352150    6.810569070   -0.000612    0.001305   -0.000066
    9    2.182281087    3.779821719   10.215855308   -0.000660   -0.001143    0.000001
 cps and forc at (iter_ion, iter_total =     2      40 )
    1    3.156999743    1.163767576    1.210005993   -0.002904   -0.005755   -0.003892
    2    6.700048015    2.151861938    4.615264365    0.006567    0.000186   -0.003832
    3    4.072421499    4.726345880    8.020748072   -0.003503    0.005487   -0.003829
    4   -1.486455954    6.878343743    5.600593135   -0.003122    0.005780    0.003831
    5   -0.570648922    3.315925959    9.005851266   -0.003532   -0.005392    0.003892
    6    2.056925355    5.889992076    2.195109289    0.006503   -0.000290    0.003828
    7    4.923176344    0.000121757    3.405351146    0.000397   -0.000013    0.000018
    8   -2.461482612    4.263656762    6.810503226   -0.000210    0.000337   -0.000017
    9    2.181621403    3.778679157   10.215856638   -0.000197   -0.000341    0.000000

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

(a) Lattice vectors.

(b) After "ntyp =," the number of elements is given. In this case, it is 2. After "natm =," the number of atoms is given. In this case, it is 9.

(c) After "(natom→type)," the correspondence between elements and atoms is shown. In this case, atoms 1–6 correspond to element "1," and atoms 7–9 correspond to element "2."

(d) After "(speciesname)," the list of elements is printed. In this case, "1" corresponds to "O" (oxygen) and "2" corresponds to "Si" (silicon).

(e) Header information for each step of an MD/optimization. In this case, "1" means the 1st step of an MD/optimization calculation, and "24" means the number of SCF iterations performed until this step.

(f) Coordinates and forces of atoms are listed. The first column is the ID of an atom. Columns 2–4 are its coordinates, and columns 5–7 are its force. If the "velocity" tag in the "printlevel" block is "2," the velocity of the atom is displayed in columns 8–10.

2.5.3 Charge density (recorded in nfchr.cube)

In the file "nfchr.cube," (or a file indicated by "F_CHR" in "file_names.data"), the charge density distribution in the Gaussian cube style is recorded. Only data from the last MD/optimization step are recorded.

Using the PHASE Viewer or other visualization software, users can view the atomic geometry and charge density distribution.

2.5.4 Density of states (recorded in dos.data)

In the file "dos.data" (or a file indicated by "F_DOS" in "file_names.data"), the DOS data are recorded.

To draw a graph of the DOS, a PHASE tool "dos.pl" is useful. The execution of this Perl script generates the file "density_of_states.eps." In the command below, "phase_v1100/bin/" is the directory in which PHASE is installed. The file "density_of_states.eps" can be viewed using ghost script or other tools. For more details, see the manual for PHASE tools.

```
% ../../phase_v1100/tools/bin/dos.pl dos.data -erange=-15,10 -with_fermi -color
```

dos.data	File containing the DOS data.
-erange	Energy range for DOS visualization; "-15,10" means a range from −15 Ht to 10 Ht.
-with_fermi	If this option is used, the Fermi level is indicated as shown in the figure below.
-color	Color output

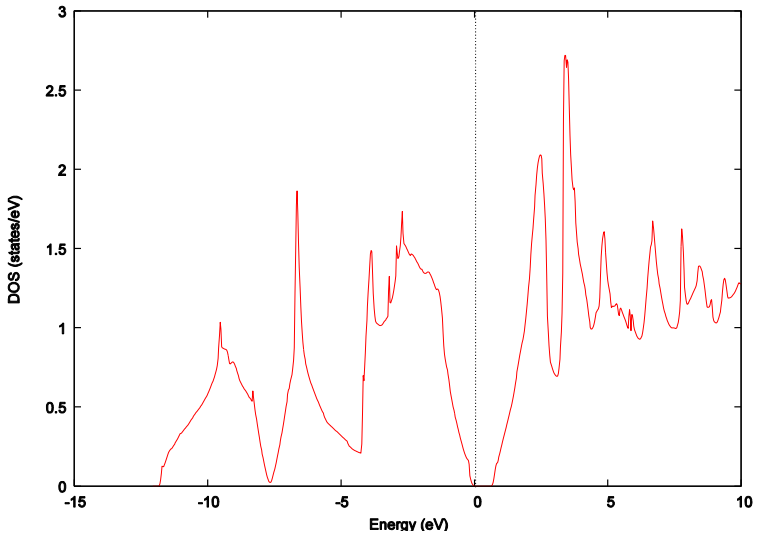


Figure 2.2 Example of the visualization of the DOS (diamond-structure Si)

2.5.5 Band structure (recorded in nfenergy.data)

In the file "nfenergy.data" (or a file indicated by "F_ENERG" in "file_names.data"), eigenvalues for all ***k***-points are recorded.

To draw a graph of the band structure, a PHASE tool "band.pl" is useful. The execution of this Perl script generates the file "band_structure.eps." In the command below, "phase_v1100/bin/" is the directory in which PHASE is installed. The file "band_structure.eps" can be viewed using ghost script or other tools. For more details, see the manual for PHASE tools.

```
% ../../phase_v1100/tools/bin/band.pl nfenergy.data bandkpt.in -erange=-15,10 -w
ith_fermi -color
```

nfenergy.data	File containing eigenvalue data.
bandkpt.in	File containing k-point data
-erange	Energy range for band structure visualization; "-15,10" means a range from −15 Ht to 10 Ht.
-with_fermi	If this option is used, the Fermi level is indicated as shown in the figure below.
-color	Color output

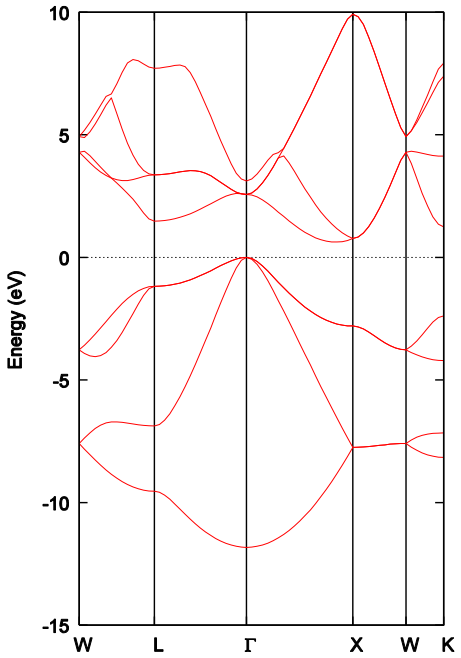


Figure 2.3 Example of the visualization of a band structure (diamond-structure Si)

2.6 References

[1] W.Kohn, L.J.Sham, Phys. Rev. A140, 1133, (1965)
[2] N. Troullier and J.L. Martins, Phys. Rev. B**43,** 1993 (1991)
[3] D. Vanderbilt, Phys. Rev. B**41** 7892 (1990)
[4] P.E. Blöchl, Phys. Rev. B 50, 17953 (1994)
[5] G. Kresse and D. Joubert, Phys. Rev. B59, 1758, (1999)
[6] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).
[7] J. P. Perdew and Y. Wang, Phys. Rev. B **45**, 13244 (1992).
[8] C.G. Broyden, Math. Comput. 19, 577, (1965)
[9] P. Pulay, Chem. Phys. Lett. 73, 393 (1980)

3. Input parameter file: nfinp.data (F_INP file)

3.1 Format of input parameter file

The input parameter file "nfinp.data" specifies a model structure (e.g., atomic positions) and calculation conditions. Although "nfinp.data" is the default name of this file, you can specify an arbitrary filename through the F_INP keyword. For example, you can set a name related to the target system.

3.1.1 Description of parameters

This section briefly describes how to write the input parameter file. In this file, input parameters are listed in hierarchical blocks, which are delimited by a block name and curly brackets {}, as shown below.

```
Upper_block{
  Lower_block{
    ...
    tag_keyword = value
  }
}
```

Each block specifies a crystal structure, calculation method, calculation accuracy, and other calculation conditions. Related parameters are listed together in one block. These blocks are defined in the format '`blockname{…}`'. Parameters are usually specified in the format '`tag_keyword = value`'. Further details for specifying parameters are described later.

In making an input file, note the following:

- Multiple blocks with the same name cannot be defined at the same hierarchical level.
- Block names are not case sensitive.
- If a block name is misspelled, the block will be ignored and default values will be employed for the variables in the block. Error messages will not be printed.
- Variables can be separated by commas as well as by line feeds.
- Double quotes are used to include spaces in a string variable;
  e.g., `title = "This is a title line for the bulk Si."`
- Two-byte characters cannot be used.

The following blocks are available at the top level.

control block	Sets options that control the entire calculation process.
accuracy block	Sets options related to calculation accuracy.
structure block	Sets an atomic structure.
wavefunction_solver block	Sets wavefunction solver.
charge_mixing block	Sets charge density mixing scheme.
structure_evolution block	Sets geometry optimization or molecular dynamics calculation.
postproccesing block	Sets post-processing.
printlevel block	Sets print level for the output file.

3.1.2 Specification of units

Although atomic units (e.g., bohr and hartree) are default units for input files, you can use other units as well. Table 3.1 lists available units in PHASE. Default units are shown in bold type.

Table 3.1 Available units in PHASE

Length	**bohr**, angstrom, nm
Energy	**hartree**, eV, rydberg
Time	**au_time**, fs, ps, ns, s, sec, min, hour, day
Velocity	**bohr/au_time**, bohr/fs, angstrom/fs, angstrom/au_time, nm/fs, nm/au_time
Force	**hartree/bohr**, hartree/angstrom, hartree/nm, eV/angstrom, eV/bohr, ev/nm, rydberg/bohr, rydberg/angstrom, rydberg/nm
Pressure	**hartree/bohr3**, hartree/angstrom3, hartree/nm3, eV/angstrom3, eV/bohr3, eV/nm3, rydberg/angstrom3, rydberg/bohr3, rydberg/nm3,
Mass	**au_mass**, atomic_mass,

A unit can be individually specified for each variable (e.g., `cpumax = 86400 sec`). Furthermore, you can specify units for an entire block. See the example below.

```
block{
  #units angstrom
  ...
  ...
}
```

In the above example, the unit of length is set to Ångstrom. When you specify multiple units, separate the units by spaces (e.g, `#units angstrom eV`).

3.1.3 Comment lines

All lines beginning with ! or // are considered to be comment lines. See the example below.

```
block{
!  comment
! tag_keyword = value1       comment
// tag_keyword = value2      comment
  tag_keyword = value3
}
```

3.1.4 Example of input parameter file

The following input file is an example for an electronic-state calculation of Si atoms (diamond structure; two Si atoms). In this example, typical calculation conditions are employed.

```
control{
  condition = initial
  cpumax = 86400 sec
  max_iteration = 10000
}

accuracy{
  cutoff_wf = 25.0 rydberg
  cutoff_cd = 100.0 rydberg
  num_bands = 8
  ksampling{
    method = monk
    mesh{
      nx = 10
      ny = 10
      nz = 10
    }
  }
  initial_wavefunctions = atomic_orbitals
  initial_charge_density = atomic_charge_density
  scf_convergence{
    delta_total_energy = 1e-10
    succession = 3
  }
  force_convergence{
    max_force = 0.001 hartree/bohr
  }
}

structure{
  element_list{
    #tag   element   atomicnumber
          Si    14
  }
  unit_cell{
    #units angstrom
    a_vector = 0 2.732299538 2.732299538
    b_vector = 2.732299538 0 2.732299538
    c_vector = 2.732299538 2.732299538 0
  }
  unit_cell_type = bravais
  atom_list{
    atoms{
      #tag   element   rx    ry    rz    imove
            Si    0.125  0.125  0.125     0
            Si   -0.125 -0.125 -0.125     0
    }
    coordinate_system = internal
  }
}

wavefunction_solver{
      solvers{
         #tag   sol   till n  prec cmix submat
```

```
                davidson    1    on   1    on
                rmm3       -1    on   1    on
        }
        rmm{
            edelta_change_to_rmm=5e-5
        }
}

charge_mixing{
        mixing_methods{
        #tag no   method    rmxs   rmxe   istr  prec nbmix
             1   pulay   0.40   0.40   3     on   15
        }
}

Postprocessing{
  dos{
    sw_dos = ON
    deltaE = 1.e-4 hartree
  }
  charge{
    sw_charge_rspace    = ON
    filetype = cube  !{cube|density_only}
    title  = "This is a title line for the bulk Si"
  }
}
```

3.2 List of tag keywords

Tag keywords for the input parameter file "nfinp.data" are listed in エラー! 参照元が見つかりません。. In this table, keywords are briefly described. Further details are described in later sections.

Table 3.2 List of tag keywords for the input parameter file "nfinp.data"

1st level block	2nd, 3rd level block	Tag keyword	Description
control			Block for specifying calculation conditions that control the entire calculation process
		condition	Specify the calculation condition. Options are: **preparation, −2**: only pre-processing is executed. **automatic, −1**: the option **initial** or **continuation** is automatically selected. **initial, 0**: the calculation is started from initial. **continuation, 1**: the previous calculation is continued. (The following options are used in EKCAL) **fixed_charge, 2**: the calculation with fixed charge density is started. **fixed_charge_continuation, 3**: the previous calculation by **fixed_charge** is continued. (defaults to **automatic**)
		cpumax	Upper limit of CPU time (defaults to 86400 sec). Units are {sec, min, hour, day}
		max_iteration max_total_scf_iteration	Maximum number of total SCF iterations (defaults to 10000)
		max_mdstep	Maximum number of total steps of an MD calculation (default: limitless)
		max_scf_iteration	Maximum number of SCF iterations in one MD step (default: limitless)
		nfstopcheck	A number written in the file "nfstop.data" that determines the number of steps to execute before stopping the process. This variable can be changed even when the calculation is running.
		sw_ekzaj	If this switch is set to ON, wavefunctions are stored in the wavefunction file F_ZAJ, which is used as an input for EKCAL. Set this switch to ON to read the file in EKCAL. Note that this is available only for the calculation at the Γ-point. (defaults to OFF)
accuracy			Block for controlling calculation accuracy
		cutoff_wf	Cutoff energy for wavefunctions
		cutoff_cd	Cutoff energy for charge density
		num_bands	Number of bands
	ksampling		Block for k-sampling
		method	Specify k-point sampling method. Options are **monk**: the Monkhorst–Pack method **mesh**: mesh generation **file**: read k-points from a file

			direct_in: Directly inputs k-points **gamma**: Γ-point only (Defaults to **monk**)
	mesh		Block for mesh generation
		nx, ny, nz	Number of mesh divisions in the X, Y, and Z directions. default value = (4,4,4) maximum value = (20,20,20)
	kshift		Block that specifies the shift of k-points. This block is enabled only for the Monkhorst–Pack method.
		k1, k2, k3	Specify the mesh displacement. Input range is [0.0, 0.5]. Default values are as follows: If the crystal system is hexagonal, k1 = k2 = 0, k3 = 0.5 Otherwise, k1 = k2 = k3 = 0.5 Here 0.5 indicates half the mesh width.
	kpoints		Block for weighting of k-points
		kx ky kz denom weight	$\vec{k}$ = (kx/denom, ky/denom, kz/denom) Coordinates and weighting of k-points
	smearing		Block for smearing of k-sampling
		method	Specify method used for smearing. Options are **parabolic**: parabolic method (default) **cold**: cold smearing method (This option is effective for metal systems.) **tetrahedron**: tetrahedron method **improved_tetrahedron**: improved tetrahedron method Note. **tetrahedron** and **improved_tetrahedron** are available only when the mesh method is selected for k-point sampling.
		width	Specify the smearing width (defaults to 0.001 hartree) This variable is used only when method = **parabolic** or **cold**.
	(no block name)		
		xctype	Specify a type of exchange-correlation energy. Options are LDA: **LDAPW91**, **PZ** GGA: **GGAPBE**, **REVPBE**
	scf_convergence		Block that specifies convergence criteria of the SCF calculation
		delta_total_energy	Convergence criterion for the total energy difference (default: 10^{-10} hartree)
		succession	The SCF iterations are terminated if the energy difference ΔE is less than the specified criterion $\boldsymbol{n}$-times in succession. The variable "succession" specifies the number $\boldsymbol{n}$. (defaults to 3)
	force_convergence		Block that specifies convergence criterion of atomic force
		max_force	Convergence criterion for the maximum force (default: 0.001 hartree/bohr)
	ek_convergence		Block that specifies convergence criteria for

			eigenvalues. This block is enabled only for EKCAL.
		num_extra_bands	Number of bands that are allowed to remain unconverged (default: 2)
		num_max_iteration	Maximum number of updates per k-point (defaults 300)
		sw_eval_eig_diff	Switch that specifies whether to evaluate eigenvalues. Options are **1**, **on**, **yes**: Evaluate (default) **0**, **off**, **no**: Do not evaluate
		delta_eigenvalue	Allowable error of eigenvalues (defaults to 10^{-15} hartree)
		succession	Number of iterations (defaults to 3)
	(no block name)		
		initial_wavefunctions	Initial guess for wave functions. Options are **random_numbers**: Initialize by random numbers **matrix_diagon**: Initialize by small matrix diagonalization **atomic_orbitals**: Initialize by atomic orbitals **file**: Input initial value from the file F_ZAJ
	matrix_diagon		Initial values of wavefunctions are given by small matrix diagonalization
		cutoff_wf	Cutoff of wavefunctions
	(no block name)		
		initial_charge_density	Initial value of charge density. Options are **Gauss**: Initialize by overlap of the Gaussian distribution function. **atomic_charge_density**: Initialize by overlap of electron density of atom. **file**: Input initial value from the file F_CHGT
	precalculation		Block for preconditioning of charge mixing
		nel_Ylm	Specify the highest order of spherical harmonics to be prepared in advance and stored in memory. (defaults to 9)
structure			Block for structure settings
		unit_cell_type	Type of unit cell. Options are **primitive** and **Bravais**.
	unit_cell	 a_vector b_vector c_vector a, b, c alpha, beta, gamma	Specify unit cell. The unit cell can be defined in the following two ways. (x,y,z) component for each lattice vector (default unit is Bohr) Lattice parameters: the a-, b-, c-axes; the angles formed by b–c, c–a, a–b axes (default unit of angle is degree)
	symmetry		
		method	Options are {manual, automatic}. The option **automatic** automatically determines symmetry.
		crystal_structure	Options are {diamond, hexagonal, fcc, bcc, simple cubic}
	tspace		Block for TSPACE.

			Details of TSPACE are described in "空間群のプログラム TSPACE (A program for space group TSPACE)" written by A.Yanase and the manual of ABCAP.
		lattice_system	Options are {rhombohedral,trigonal,r,t,-1}, {hexagonal,h,0}, {primitive,simple,p,s,1}, {facecentered,f,2}, {bodycentered,b,3}, {bottomcentered,basecentered,onefacecentered,bot,ba,o,4}
		num_generators	Number of generators (an integer value from one to three)
		generators	Generators
		af_generator	Generators for a magnetic space group
	(no block name)		
		sw_inversion	Switch for inversion symmetry
	(no block name)		
		magnetic_state	Options are {para, antiferro, ferro} **antiferro** can be abbreviated to **af**.
	atom_list		Atom list
		coordinate_system	Options are {cartesian, internal}
	atoms		Block of atoms (tabular form). This table contains the following columns.
		rx, ry, rz	xyz coordinates
		element	Element names
		mobile	Flags that specify whether the atoms can move or not during the calculation (Use {1,0}, {on,off}, or {yes, no})
		weight	If **sw_inversion=on** and **weight=2**, copied atoms are generated at positions of inversion symmetry.
	element_list		
		element	Element names. This parameter must match that specified in the **element** column of the **atoms** block.
		atomic_number	Atomic number
		mass	Mass
		zeta	Spin polarization: $\zeta = (n_\uparrow - n_\downarrow)/(n_\uparrow + n_\downarrow)$
		deviation	Deviations of Gaussian functions used to determine the initial guess of charge density distribution (σ of $\rho(r) = A\exp(-r^2/2\sigma^2)$, which determines deviation of electron density distribution). **dev** or **standard_deviation** can also be used as the tag name instead of **deviation**.
wavefunction_solvers	solver		Wave function solver
		sol	Specify the wavefunction solver. Options are **MatrixDiagon**: matrix diagonalization method **lm+MSD**: lm(line minimization) + MSD(modified steepest descent) method **RMM2P**, **RMM3**: The RMM method **MSD**: The modified steepest descent method **submat**: subspace rotation method

			Davidson: The Davidson method
		till_n	The wavefunction solver specified in the **sol** column is employed until the ***n***-th step. This column **till_n** sets the number ***n***.
		dts	Initial value of time step
		dte	Time step for the steps after the **itr**-th step. If only **dts** is specified, the same value is set to **dte**.
		itr	Time step is changed from **dts** to **dte** after the ***n***-th step. The variable **itr** sets the number ***n***.
		var	An interpolation method. Options are {**linear**, **tanh**}
		prec	Switch that specifies whether to execute preprocessing. Options are {on, off}
		cmix	Variables that specify which charge-mixing method is used for each wavefunction solver by the number corresponding to the order of the method listed in the **charge_mixing** block.
		submat	If this variable is **on**, subspace rotation as specified in the **subspace_rotation** variable is performed. Options are {on,off}
	line_minimization		Block for line minimization
		dt_lower_critical dt_upper_critical	Lower and upper limits of time steps for line minimization (default values are 0.005 and 2.0, respectively)
		delta_lmdenom	Factor of line minimization
	rmm		The residual minimization method
		imGSrmm	Specify how often Gram–Schmidt orthogonalization is applied to the wave functions updated by the RMM method. (default value is **imGSrmm=1**, which means orthogonalization is performed every step)
		rr_Critical_Value	Convergence criterion for each band. If the residual error of the band becomes less than the criterion, the updating of this band will stop.
		edelta_change_to_rmm	Threshold used to switch the wave function solver to the RMM method. If the difference in the total energy becomes less than the value specified by this variable, the wavefunctions solver is changed to the RMM method.
	subspace_rotation		Block for controlling subspace diagonalization
		subspace_matrix_size	Size of subspace matrix. Default value is equal to the number of bands (i.e., **num_bands**). If a specified value is larger than the **num_bands**, the value of **num_bands** is set to the keyword.
		damping_factor	A damping factor for off-diagonal elements. If a specified value exceeds the range of [0.0, 1.0], it is treated as 1.0.
		period	If the **submat** variable in the **solver** block is ON, subspace rotation is performed once per **period** times. If the **period = 3**, for example, the subspace rotation is performed when

			iteration = 1,4,7,10,… (Defaults to 1)
		critical_ratio	If the ratio between values of off-diagonal and diagonal elements becomes less than the **critical_ratio** once, then subspace rotation is not performed for the elements. (Defaults to 10^{-15})
charge_mixing			Block for the charge-mixing method
	mixing_methods		Mixing methods of charge density
		method	Method of charge mixing. Options are {simple,broyden2,pulay} (Defaults to simple)
		rmxs	Initial value of charge mixing ratio (Defaults to 0.5)
		rmxe	Charge mixing ratio for the steps after **itr**-th step. If only **rmxs** is specified, the same value is set to **rmxe**. (Defaults to 0.5)
		itr	Number of the steps taken to vary the charge mixing ratio form **rmxs** to **rmxe**.
		var	Method of varying the charge mixing ratio Options are {**linear**, **tanh**}
		prec	Switch that specifies whether to execute preprocessing. Options are {on,off}
		istr	When the specified method is not **simple**, this method is employed after the **istr**-th step.
		nbmix	Charge density of the previous n steps will be stored to arrays. The variable **nbmix** specifies the number n.
		update	Specify a way of renewing the arrays that store the charge density when the arrays are filled. Options are **anew**: discard all stored data and reallocate the arrays **renew**: replace the oldest data with the newest data
	charge_preconditioning		Block for preconditioning of charge mixing
		amix	Variable for preconditioning
		bmix	Variable for preconditioning
structure_evolution			Block for structure optimization and molecular dynamics
		method	Options are {sd, quench, gdiis, bfgs, cg, velocity_verlet}
		dt	Time step
	stress		Calculation of stress
		sw_stress	Switch that specifies whether to calculate stress. Options are {on, off}
	gdiis		Block for the GDIIS and BFGS methods
		initial_method	Initial method that is employed before switching the method to GDIIS or BFGS. Options are {quench, cg, sd}
		gdiis_box_size	Atomic coordinates of the previous n steps

			will be stored in arrays. The variable **gdiis_box_size** specifies the number n.
		gdiis_hownew	Method of renewing the arrays that store the atomic coordinates after the **gdiis_box_size** steps. Options are {anew, renew}
		c_forc2gdiis	Threshold used to switch the method to GDIIS or BFGS. Defaults to 0.0025 (hartree/bohr)
postprocessing			
	dos		Output of density of states
		sw_dos	Switch that specifies whether to output the density of states. Options are {on, off}
		method	Options are {tetrahedral, Gaussian}
		deltaE_dos	Energy accuracy for the density of states
		variance	Variance of the Gaussian function. This variable is enabled only when **method=Gaussian**.
		nwd_dos_window_width	Energy width ΔE is specified by the following equation ΔE =nwd_dos_window_width × deltaE_dos
	charge		Output of charge
		sw_charge_rspace	Switch that specifies whether to output charge density. Options are {on, off}
		filetype	File format of the charge density file Options are {cube, density_only}
		title	Title of the charge density file. This variable is enabled only when **fileytype=cube**.
printoutlevel			Print level of the standard output 0: no output 1: output normally 2: output extra information for debugs
		base	This variable becomes the default values for the other variables that specify the print level.
		pulay	Pulay charge-mixing method
		timing	Timing information
		solver	Electronic state solver
		evdff	Difference of eigenenergies
		rmm	The RMM method
		snl	Non-local potential
		gdiis	The GDIIS method
		eigenvalue	Eigenvalue
		spg	Space group
		kp	k-points
		matdiagon	Matrix diagonalization method
		vlhxcq	Local potential
		totalcharge	Electron density
		submat	Subspace matrix rotation method
		strcfctr	Structure factor
		parallel	Print level for the result of preprocessing of parallelization
		input_file	Output of analysis result of the input file F_INP
		parallel_debug	If this variable is set to 1, not only the 0-th

			process but also other processes will output the information to files such as output00x_xxx.
		jobstatus	If this variable is set to 1, progress of the calculation will also be printed in the file jobstatus00x.
	jobstatus_option		Output of the job status
		jobstatus_format	Options are tag, tag_line, and tabta. (Defaults to tag)
		jobstatus_series	ON or OFF

3.3 Control block

The **control** block contains variables that control the entire calculation process or specify general options. An example of the control block is shown below.

```
control{
    condition = initial
    cpumax = 1 day
    max_iteration = 1000000
}
```

The **control** block contains the following variables.

condition — Specifies whether the calculation starts from an initial condition or continues a previous calculation. Available options are as follows:

preparation: This option only executes pre-processing (e.g., printing size of allocated memory, generating k-points).

initial: The calculation starts from an initial condition (default).

continuation: The previous calculation is continued. Wave functions, charge density distribution, and other data are read from files generated by the previous calculation.

automatic: This option automatically chooses **initial** or **continuation**. The option **continuation** is chosen if the necessary files for **continuation** exist. (These files are automatically generated by the previous calculation). Otherwise, the option **initial** is selected.

(The following options are used in EKCAL.)

fixed_charge: Charge density distribution is read from files, and only wave functions are converged while the charge distribution is fixed. This option is employed for the purpose of only calculating, for example, band structure.

fixed_charge_continuation: Continuation of the **fixed_charge**.

Default value is **initial**. The keywords **initial**, **continuation**, **automatic**, **preparation**, **fixed_charge**, **fixed_charge_continuation** can be substituted by integer numbers 0, 1, −1, −2, 2, 3, -3, respectively.

cpumax — Specifies a time limit for the execution of PHASE. The value is specified in the format "real number + unit." Default value is 86400 s (i.e., one day). Available units are "sec," "s" (identical with "sec"), "min," "hour,"and "day." Unit cannot be omitted.

If the execution time exceeds the specified value, PHASE terminates the calculation and generates restart files even if the calculation has not yet converged.

If there is a possibility of exceeding the time limit given by a job scheduler, it is recommended to set a smaller value than the time limit.

(For example, if the job time limit is six hours and there is no post-processing, such as calculating density of states, "5.8 hour" may be appropriate.)

max_iteration
max_total_scf_iteration — Specifies the maximum number of total SCF iterations. If the number of total SCF iterations exceeds the specified value, the calculation halts, and restart files are generated. Default value is 10000.

max_scf_iteration — Specifies the maximum number of SCF iterations for one step of structure optimization or molecular dynamics. The first steps of structure optimization often require hundreds of SCF iterations to converge because of the instability of the initial structure. In such cases, it is recommended to truncate the SCF iterations early and calculate force to update atomic positions to a more stable structure. However, a very small value (e.g., 10) causes serious errors and makes the calculation slow. Do not use if an accurate force is essential.

3.4 Accuracy block

3.4.1 Cutoff energy

Cutoff energy is an important parameter that determines the accuracy of the calculation using a plane-wave basis set.

The cutoff energies are specified as follows:

```
accuracy{
  cutoff_wf = 25 Rydberg
  cutoff_cd = 225 Rydberg
}
```

cutoff_wf	Specifies cutoff energy for wave functions in units of energy.
cutoff_cd	Specifies cutoff energy for charge density in units of energy.

Although it should be examined whether the given cutoff energy can achieve sufficient accuracy, the following guides are also useful.

- 25 rydberg may be appropriate for **cutoff_wf**.
- If norm-conserving pseudopotentials are employed, four times **cutoff_wf** may be appropriate for **cutoff_cd**. Otherwise, nine times **cutoff_wf** may be appropriate.

3.4.2 Number of bands

The number of bands is specified by the **num_bands** variable in the **accuracy** block as follows.

```
accuracy{
  num_bands = 12
}
```

num_bands	Number of bands

The number of bands must be larger than half the number of valence electrons. Typically, **num_bands** is set to 1.2 times the minimum value. If the specified value is less than the lower limit, the value will be automatically increased. If no value is given to the keyword, the value is automatically set.

3.4.3 k-point sampling and smearing

In addition to the cutoff energy, k-point sampling is also an important factor that determines the reliability of the calculation. It is set up in the **ksampling** block of the **accuracy** block. An example is shown below.

```
accuracy{
  ksampling{
    method = monk
    mesh{
      nx=4
      ny=4
      nz=4
    }
  }
}
```

The ksampling block contains the following variables and blocks.

method — Specifies the method used for k-point sampling. Options are
monk: k-point sampling based on the Monkhorst–Pack method. Typically, this option is recommended. (default)
mesh: The reciprocal space is divided by a simple mesh. This option is specified when the tetrahedron method is employed to calculate the charge density distribution and the density of states.
file: k-points are read from a file. This option is used, for example, when user-defined k-points are specified to calculate the band structure.
gamma: Only the Γ-point is sampled. This option is specified when a sufficiently large unit cell is employed and the Γ-point is sufficient to obtain adequate accuracy.
In any method, if the Γ-point is included in k-point sampling and inversion symmetry is not applied for the system, calculation of wavefunctions at the Γ-point is executed about three times faster than that at other k-points by using symmetry of this point. (You can also apply the normal method to the Γ-point as well as other k-points as described later.)

mesh — Number of divisions along each reciprocal space direction, as specified by
nx: number of divisions along the first reciprocal lattice vector
ny: number of divisions along the second reciprocal lattice vector
nz: number of divisions along the third reciprocal lattice vector

Smearing is a manipulation that smears electrons over several orbitals within the range of the Fermi level. By this manipulation, high accuracy may be archived with few k-points for metal systems that have many states near the Fermi level. Smearing is specified in the block **smearing** of the **accuracy** block as follows.

```
accuracy{
  smearing{
    method = parabolic
    width = 0.001 hartree
  }
}
```

The **smearing** block contains the following variables.

method — Specifies a method of smearing. Options are
parabolic: Smearing of the electron distribution near the Fermi level.
tetrahedron or **improved_tetrahedron**: tetrahedron method. These options are selected when the tetrahedron method is employed to calculate the density of states.
cold: Cold smearing. This option is generally effective for metal systems.

width — Specifies the smearing width in units of energy. Default value is 0.001 hartee.
This variable is enabled only when **method=parabolic**.

3.4.4 Exchange-correlation energy

Exchange-correlation energies are classified into LDA or GGA. In PHASE, LDAPW91 and PZ are available for LDA, while GGAPBE and REVPBE are available for GGA.

```
accuracy{
  xctype = ggapbe
}
```

xctype — Exchange-correlation energy (LDA, GGA)
LDA: LDAPW91, PZ
GGA: GGAPBE, REVPBE

3.4.5 Convergence criteria

There are two types of convergence criteria: criterion for SCF calculation and criterion for structure optimization. These criteria are specified as follows:

```
accuracy{
  scf_convergence{
    delta_total_energy = 1.0E-8 Hartree
    succession = 3
  }
  force_convergence{
    max_force = 2.0E-4 Hartree/Bohr
  }
}
```

Blocks and variables related to convergence criteria are shown below.

scf_convergence — This block specifies convergence criteria for the SCF calculation.

delta_total_energy — Convergence criterion for the total energy difference. If the difference between the current total energy and the total energy of the previous step is smaller than the specified value, the convergence criterion is satisfied.
Default value is 1e−10 hartree.

succession — SCF iterations are terminated if the energy difference is smaller than the criterion **delta_total_energy** *n* times in succession. The variable **succession** specifies the number *n*. Default value is 3.

force_convergence — This block specifies the convergence criterion for structure optimization.

max_force — Convergence criterion for the maximum force in units of force.
Default value is 1e−3.

3.4.6 Initial wavefunctions and initial charge density

The SCF calculation can converge faster if initial wavefunctions and initial charge density are appropriately set up. Types of initial wave functions and initial charge density are specified as follows:

```
accuracy{
  initial_wavefunctions = atomic_orbitals
  intial_charge_density = atomic_charge_density
  matrix_diagon{
    cutoff_wf = 5 rydberg
  }
}
```

Blocks and variables related to initial wavefunctions and initial charge density are described below.

initial_wavefunctions — Specifies a method to initialize wavefunctions. Options are
random_numbers: Initial guess is obtained using random numbers.
matrix_diagon: Initial guess is obtained by matrix diagonalization. The cutoff energy for this procedure can be specified by the block **matrix_diagon** described later.
file: Initial guess is read from a wavefunctions file. If you have a data file of wavefunctions that is already almost converged, you can specify this file to read them.
atomic_orbitals: Wavefunctions are initialized by data of atomic orbitals that is saved in the pseudopotential files.
(Defaults to **random_numbers**)

initial_charge_density — Specifies a method used to initialize charge density. Options are
Gauss: Initial guess is obtained by simple atom-centered Gaussian functions.
file: Initial charge density is read from a file. If you have a data file of charge density

that is already almost converged, you can specify this file to read density.
atomic_charge_density: Charge density is initialized by the atomic orbitals saved in the pseudopotential files.
(Defaults to **Gauss**)

matrix_diagon — This block is for matrix diagonalization.
This block is enabled only when **initial_wavefunctions=matrix_diagon**.

cutoff_wf — Specifies the cutoff energy for initializing wavefunctions.
Default value is half the normal cutoff energy.

3.5 Structure block

A model of atomic structure is specified in the block **structure** as follows:

```
structure{
    unit_cell_type = Bravais
    unit_cell{
        #units angstrom
        a_vector = 4.914100000 0.000000000 0.000000000
        b_vector = -2.457050000 4.255735437 0.000000000
        c_vector = 0.000000000 0.000000000 5.406000000
    }
    atom_list{
        coordinate_system = Internal
        atoms{
            #units angstrom
            #tag element rx ry rz
             O 0.413100000054 0.145400000108 0.118930000000
             O 0.854599999943 0.267699999886 0.452263333333
             O 0.732300000003 0.586900000006 0.785596666667
             O 0.267699999946 0.854599999892 0.547736666667
             O 0.145399999997 0.413099999994 0.881070000000
             O 0.586899999939 0.732299999879 0.214403333333
             Si 0.530000000000 0.000000000000 0.333333000000
             Si -0.000000000072 0.529999999857 0.666666333333
             Si 0.469999999954 0.469999999908 0.999999666667
        }
    }
    element_list{
        #tag element atomicnumber mass zeta deviation
         O 8 29164.9435 * *
         Si 14 51196.4212 * *
    }
    symmetry{
          method = automatic
          sw_inversion = off
    }
}
```

3.5.1 Unit cell

unit_cell_type — Specifies a type of unit cell. Options are **primitive** or **bravais**. Default value is **bravais**.
To define a unit cell by lattice parameters, the variable **unit_cell_type** must be **bravais**.
In addition, when **unit_cell_type=bravais**, you can convert the lattice by using the variable **lattice_system** in the block **tspace** in the block **symmetry**. The details of the **lattice_system** variable are described latter.

unit_cell — This block specifies the unit cell. You can specify the unit cell by cell vectors or by lattice parameters. Specifying the unit cell by lattice parameters is enabled only when

unit_cell_type=bravais.

- Specifying by cell vectors

By using cell vectors, you can specify the unit cell as follows:

```
unit_cell{
    #units angstrom
    a_vector = a1 a2 a3
    b_vector = b1 b2 b3
    c_vector = c1 c2 c3
}
```

The a-axis, b-axis, and c-axis are specified by the variables **a_vector**, **b_vector**, and **c_vector,** respectively. The unit of length can be defined for the entire block, not for each variable. In this example, the unit of length is set to Ångstrom by the description "`#units angstrom`."

- Specifying by lattice parameters

By using lattice parameters, the unit cell can be defined as follows:

```
unit_cell{
   a = a0
   b = b0
   c = c0
   alpha = alpha0
   beta = beta0
   gamma = gamma0
}
```

The variables **a**, **b**, **c**, **alpha**, **beta**, and **gamma** specify the lattice parameters $a, b, c, \alpha, \beta,$ and γ, respectively. If a unit cell is specified using lattice parameters, cell vectors will be defined by the lower triangular matrix as follows:

```
a_vector = a1 0.0 0.0
b_vector = b1 b2  0.0
c_vector = c1 c2  c3
```

3.5.2 Atomic coordinates

atom_list		This block specifies atomic coordinates, etc.
coordinate_system		Specifies whether the atomic coordinates are defined using Cartesian coordinates or fractional coordinates. Options are **internal**: atomic positions are referred to lattice vectors. **cartesian:** atomic positions are given in Cartesian coordinates. Default value is **internal**.
atoms		This block specifies atomic coordinates and other properties of atoms. These properties are defined in a tabular form. Representative properties are list below.
	element	Element names. The element names used here need to be defined in the block **element_list** described latter.
	rx	x-coordinates
	ry	y-coordinates
	rz	z-coordinates
	mobile	Flag that specifies whether or not the atoms can move in geometry optimization or in molecular dynamics. Set this flag to on, to update the atomic coordinates. Default value is off.
	weight	If **sw_inversion=on** and **weight=2**, copied atoms are generated at positions

of inversion symmetry. Defaults to 1.

3.5.3 Atomic species

element_list		This block specifies elements and their properties. These properties are defined in tabular form. Representative properties are list below.
	element	Name of elements (required).
	atomicnumber	Atomic number (required).
	mass	Mass
	zeta	Initial value of spin polarization. This variable is enabled only when spin is considered. $\zeta = (n_\uparrow - n_\downarrow)/(n_\uparrow + n_\downarrow)$

The pseudopotential files are specified by the file pointer F_POT(n) in the file "file_names.data." Here *n* is an integer that corresponds to the order of elements in the **element_list** block. For example, if the **element_list** is defined as follows

```
structure{
  ...
  ...
  element_list{
    #tag element atomicnumber mass zeta deviation
    O 8 29164.9435 * *
    Si 14 51196.4212 * *
  }
}
```

and the corresponding pseudopotential files for Si and O atoms are Si_ggapbe_nc_01.pp and O_ggapbe_us_01.pp, respectively, you can specify these files as shown below.

```
&fnames
F_INP='./nfinp.data'
F_POT(1)='./Si_ggapbe_nc_01.pp'
F_POT(2)='./O_ggapbe_us_01.pp'
/
```

The pseudopotential files that can be downloaded from our website are generated either by GGA/PBE or by LDA/PW91.You can identify which functionals are used to generate the pseudopotential by their filenames because these filenames contain the string "ggapbe" or "ldapw91." Note that you cannot execute the hybrid calculation composed of ggapbe and ldapw91. Ultrasoft, PAW, and norm-conserving pseudopotentials should not be mixed. Atomic species can be listed up to 16 types

3.5.4 Symmetry

symmetry	This block specifies symmetry of the system. Using symmetry enables us to significantly reduce the calculation amount. The following blocks and variables are available.
method	This method determines symmetry. Options are **manual**: symmetry is directly specified by input (default). **automatic**: symmetry is automatically determined.
sw_inversion	Switch that specifies whether inversion symmetry is applied. The center of inversion symmetry is set to the origin (0,0,0). Note that this option is efficient for systems having inversion symmetry, but if this option is applied to a system that has no inversion symmetry, incorrect calculations will be carried out.
tspace	This block specifies the generator by using TSPACE
lattice_system	Specify the type of lattice. This variable is enabled only when **unit_cell_type=bravais**.

	Options are **facecentered**, **bodycentered**, **basecentered**, and **rhombohedral**. If this variable is specified, the input lattice is converted to the lattice specified. See Table 4.1 for more details on converting lattices. By using this variable, you can input the lattice using a Bravais lattice, which is easy to specify. However, the actual calculation is performed using a basic lattice, which is easy to calculate. By this option, only the unit cell is converted. Therefore, atomic potions need to be defined to fit the basic lattice. For instance, in the case of a face-centered cubic lattice, only the atom on the origin should be specified. k-point sampling should also be specified to fit the converted lattice.
generators	This table specifies the generators. The generators can be defined up to only three. See section 4.2 for more details on specifying generators.

3.6 Wavefunction_Solver block

3.6.1 Calculation flow of PHASE

Figure 3.1 shows calculation flow of PHASE.

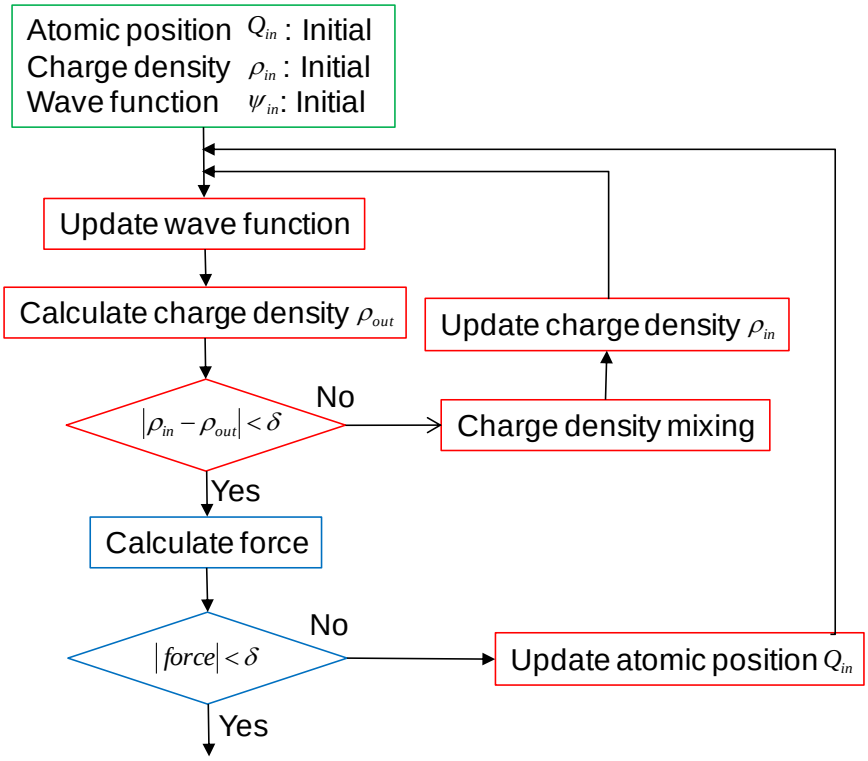


Figure 3.1 Calculation flow of PHASE

The Kohn-Sham equation

$$(H_{KS}(\rho_{inp}) - \epsilon_i)\psi_i = 0 \tag{1}$$

is solved during updating wavefunctions. Trial wavefunctions are given, and the equation

$$\Delta_i = (H_{KS}(\rho_{inp}) - \epsilon_i)\psi_i$$

is iteratively solved to obtain the solution to (1). Here Δ_i corresponds to a differential energy ϵ for wavefunctions ψ, and it approaches zero during optimization of the wavefunctions. During the process of generating the charge density in Fig. 3.1, by using the updated wavefunctions, a new charge density ρ is given by

$$\rho_{out} = 2\sum_{occ.} |\psi_i|^2$$

The inner loop in Fig. 3.1 is processed until self-consistency is achieved ($\rho_{inp} = \rho_{out}$). This calculation is called a self-consistent field (SCF) calculation. The outer loop in Fig. 3.1 is for structure optimization. Atomic positions are updated until the forces acting on atoms become almost zero.

3.6.2 Wavefunction solver

During the SCF calculation, wavefunctions are updated by the "wavefunction solver." The wavefunction solver is specified in the block **wavefunction_solver**.

```
wavefunction_solver{
    solvers{
    #tag sol      till_n prec cmix submat
         msd        1     on   1    on
         davidson   2     off  1    off
         rmm3      -1     on   1    on
    }
    davidson{
         max_subspace_size = 12
         ndavid = 4
    }
    rmm{
        edelta_change_to_rmm = 1e-3
    }
}
```

The following blocks and variables are available in the **wavefunction_solver** block.

solvers — This block specifies the wavefunction solver.

- sol — Specify an algorithm for the wavefunction solver. Options are
  - **MSD**: the modified steepest descent method. Although the computational cost for each step is the lowest, it is difficult to get convergence by using only this method. This method is mainly used for solving initial wavefunctions.
  - **lm+MSD**: the modified steepest descent method combined with line minimization. The computational cost for each step is lower than that of the other methods below, and this method can converge faster than the MSD method.
  - **CG**: the conjugate gradient method. Although the computational cost is higher than that of **lm+MSD**, convergence is normally better.
  - **Davidson**: the Davidson method. Although the computational cost for each step is higher than that of other methods, this method generally provides good accuracy.
  - **RMM3**: The RMM3 method has a lower computational cost than the Davidson method, while its accuracy is about the same. However, this method is not stable for random wavefunctions; thus, you should use another solver for the initial SCF steps before applying RMM3.
- till_n — The wavefunction solver specified in the **sol** column is applied until the n-th step. This **till_n** column specifies the number n.
  In the above example, the MSD method is employed for the first step, the Davidson method is applied until the second step (However, unless the criterion **edelta_charge_to_rmm** is satisfied, the Davidson method will be applied for later steps). If a negative value is set for **till_n**, the specified solver will be applied until the final step. Therefore, in the example, the RMM3 method will be employed until the last step.
- prec — Flag that specifies whether to execute preprocessing. Normally this should be "on." However, in case of the Davidson method, "off" may be appropriate.
- cmix — Specify the method used for charge mixing. Details of charge mixing are described later.
- submat — Specify whether to perform subspace matrix diagonalization. Options are ON and OFF. Normally, this should be "on," but it is not necessary to set to "on" for the Davidson method because subspace matrix diagonalization is already implemented in this method.

davidson — Block for the Davidson method. The following variables are available to control the Davidson method.

max_subspace_size — Specify the maximum size of the subspace used in the Davidson method. Default value is four times the number of bands.

ndavid — The Davidson method updates wavefunctions while gradually spreading the subspace. This variable **ndavid** specifies the number of steps to be used to spread the subspace. Defaults to 5.

rmm — Block for the RMM method

edelta_change_to_rmm — The RMM method is not stable for the totally unconverged wavefunctions in early steps. Therefore, another wavefunction solver is used for the initial steps before applying RMM. This variable **edelta_change_to_rmm** is a threshold used to switch the wave function solver to the RMM method. If the difference in the total energy becomes less than the value specified by this variable, the wavefunctions solver is changed to the RMM method.

line_minimization — Block for line minimization. Line minimization is performed in the lm+MSD method and CG method to obtain an appropriate step size.

dt_lower_critical — Specify lower limit for the step size of line minimization. (Defaults to 0.1)

dt_upper_critical — Specify upper limit for the step size of line minimization. (Defaults to 2.0.)

3.7 Charge_Mixing block

3.7.1 Charge mixing method

In the SCF iteration, the calculated charge density is mixed with the density obtained from the previous iteration to obtain the input density for the next iteration. This "charge mixing" is specified by the block **charge_mixing** as follows.

```
charge_mixing{
    mixing_methods{
        #tag method rmxs rmxe prec istr nbmix
             pulay 0.4 0.4 on 3 15
    }
    charge_preconditioning{
        amix = 0.9
        bmix = -1
    }
}
```

In the **charge_mixing** block, the methods used for charge mixing are specified by the following blocks and variables.

mixing_methods — This tabular form block specifies the methods used for charge mixing. Multiple methods can be defined in this table. These methods are specified by the **cmix** column of the **solvers** table mentioned above. The **cmix** column specifies which charge-mixing method is used for each wavefunction solver by the number corresponding to the order of the method list.

- method — Specify an algorithm used for charge mixing. Options are
  simple: simple mixing
  broyden2: improved Broyden method
  pulay: The RMM-DIIS method by Pulay
  Note. The **broyden2** and **pulay** are quasi-Newton methods.
- rmxs — Initial value of charge mixing ratio.

	rmxe	Final value of charge mixing ratio.
	prec	Specify whether to execute preprocessing. Normally, this should be "on".
	istr	Even if **broyden2** or **pulay** is selected, the simple method is employed for the first steps. This variable **istr** specifies the number of steps that use the simple method.
	nbmix	When **broyden2** or **pulay** is selected, this variable **nbmix** specifies the number of previous iterations whose charge density is stored as a record.
charge_preconditioning		Set the preconditioning factor. If preconditioning is enabled, the mixing ratio of G is determined by the following equation. $$\rho_{new}(G) \Leftarrow \big(1 - f(G)\big)\rho_{old}(G) + f(G)\rho_{new}(G),$$ $$f(G) = \frac{rmx * amix}{1 + \left(\frac{G_0^2}{G^2}\right)}$$ $$G_0 = \mathrm{bmix} * G_{\min}$$ Here $G_{\min}$ represents the minimum value of G (excluding the origin). Values for amix and bmix can be specified by the variables in the same block, but default values are recommended.
	amix	
	bmix	

3.7.2 Technics to accelerate the convergence

Here we introduce some techniques that are useful when the SCF calculation does not rapidly converge.

(1) Subspace diagonalization

By default, subspace diagonalization is not carried out. If subspace diagonalization is executed, although each step consumes more calculation time, convergence will be accelerated in most cases. To perform the diagonalization, define the **submat** column and set its value to "on."

```
wavefunction_solver{
        solvers{
        #tag    sol     till_n  dts  dte  itr  var     prec  cmix submat
                lmMSD     -1     0.2  1.0  40   linear  on    1    on
        }
}
```

The behavior of SCF convergence is affected by whether subspace diagonalization is applied before updating wavefunctions or after. This difference is especially significant when the RMM method is employed. By default, diagonalization is applied after updating wavefunctions. To apply it before updating wavefunctions, set the variable **before_renewal** to "on."

```
wavefunction_solver{
        solvers{
        #tag    sol     till_n  dts  dte  itr  var     prec  cmix submat
                lmMSD     -1     0.2  1.0  40   linear  on    1    on
        }
        submat{
              before_renewal=on
        }
}
```

Subspace diagonalization is more effective when many bands are involved. Because of this, although the computational cost generally increases with an increasing number of bands, the entire calculation time may sometimes be reduced when you increase the number of bands.

(2) Truncation of SCF iterations

When the initial atomic coordinates significantly differ from a stable structure, the first steps of structure optimization often require a large number of SCF iterations to converge. In this case, it is recommended to truncate the SCF iterations early and calculate the forces needed to update atomic positions to a more stable structure. This may reduce the entire calculation time. To truncate the SCF iterations, set the variable **max_scf_iteration** in the **control** block as follows.

```
control{
   ...
   max_scf_iteration = 50
}
```

In the above example, the SCF calculation will end after the 50-th iteration even if the SCF calculation has not converged, and the force is calculated to update atomic positions.

(3) Changing the mixing ratio of total and spin charge densities

When spin is considered, total charge densities and spin charge densities (i.e., difference between charge densities of up spin and down spin) are individually mixed, and different mixing ratios can be applied to these two charge mixings. To set a different charge-mixing ratio, define the **spin_density_maxfactor** variable as shown below.

```
charge_mixing{
        spin_density_mixfactor = 4
        mixing_methods{
        #tag   no  method    rmxs  rmxe  prec  istr  nbmix  update
               1   broyden2  0.1 0.1 on 3 15 renew
        }
}
```

In the above example, since the **spin_density_mixfactor** is set to four, the mixing ratio of spin density is set to 0.4 (= 0.1 × 4).

If you want to mix charge densities of up spin and down spins, set the **sw_recomposing** variable to "off":

```
charge_mixing{
    sw_recomposing = off
    ...
}
```

(4) Changing the algorithm used for spin charge mixing

You can force PHASE to employ the simple mixing method for spin charge density. This option can be specified by setting the variable **sw_force_simple_mixing** in the **spin_density** block to "on" as follows.

```
charge_mixing{
        sw_recomposing=on
        spin_density_mixfactor = 4
        mixing_methods{
        #tag   no  method    rmxs  rmxe  prec  istr  nbmix  update
               1   broyden2  0.1 0.1 on 3 15 renew
        }
        spin_density{
```

```
            sw_force_simple_mixing = on
        }
}
```

(5) Fixing spin

If you perform the SCF calculation with a fixed spin, convergence may be improved. This option is specified by the block **ferromagnetic_state** in the **structure** block as follows

```
structure{
    ...
    ferromagnetic_state{
        sw_fix_total_spin = on
        spin_fix_period = INITIALLY
        total_spin = 1.0
    }
    ...
}
```

The following variables can be specified in the **ferromagnetic_state** block.

sw_fix_total_spin	If "on," total spin is fixed.
spin_fix_period	Specify how to fix total spin. Options are **INITIALLY**: total spin is fixed to the value of the initial SCF iteration, and the constraint will be weakened step by step. **WHOLE**: total spin is fixed to the initial value until the final step. *any integer value:* total spin is fixed until the specified number of iterations. After that, a normal calculation is performed.
total_spin	Specify total spin (i.e., difference between up and down spins). Total spin for the entire unit cell is specified.

(6) Mixing of "deficit charge"

In the PAW method, mixing of "deficit charge" is performed. In the DFT+U method, mixing of occupied matrix is performed, but this is actually identical to mixing of "deficit charge." To apply the same algorithm as normal charge mixing to the mixing, set the switch **sw_mix_charge_hardpart** to "on."

```
charge_density{
    ...
    sw_mix_charge_hardpart = on
    ...
}
```

By this setting, convergence of the PAW method and the DFT+U method may be improved

3.8 Structure_evolution block

Parameters related to structure optimization or molecular dynamics are specified in the block **stucture_evolution**.

3.8.1 Structure optimization

Execution of structure optimization is specified in **structure_evolution** as follows.

```
...
structure_evolution{
    method = quench
    dt = 50
    ...
}
...
```

method	A method of structure relaxation. Options are **quench**: quenched MD method (default) **cg**: CG method **gdiis**: GDIIS method **bgfs**: BFGS method
dt	Time step for structure relaxation. Appropriately large values converge faster, but too large a value may make the calculation incorrect. Defaults to 100 au.

Because the GDIIS and BFGS methods are not stable when the force is large, the quenched MD or CG method is employed for earlier steps, and the method will be switched to GDIIS (or BFGS) after the force becomes sufficiently small.

The initial method for the earlier steps and the criterion for switching the method to GDIIS (or BFGS) are specified by the variables **initial_method** and **c_forc2gdiis**, respectively.

```
...
structure_evolution{
    method = gdiis
    dt = 50
    gdiis{
        initial_method = cg
        c_forc2gdiis = 0.0025 hartree/bohr
    }
}
...
```

The block **gdiis** is common to GDIIS and BFGS. Default values are **quench** for **initial_method** and 0.0025 hartree/bohr for **c_forc2gdiis**.

gdiis		Block for the GDIIS and BFGS methods.
	initial_method	Initial method for the earlier steps. Options are {quench, cg, sd}.
	gdiis_box_size	The atomic coordinates of the previous n steps will be stored in arrays. The variable **gdiis_box_size** specifies the number n.
	gdiis_hownew	Method of renewing the arrays that store the atomic coordinates after the **gdiis_box_size** steps. Options are {anew, renew}
	c_forc2gdiis	Criterion for switching the method to GDIIS or BFGS.

Defaults to 0.0025 (hartree/bohr).

3.8.2 Molecular dynamics

Parameters related to molecular dynamics are specified in the block **structure_evolution**.

```
structure_evolution{
    method = velocity_verlet
    dt = 100
}
```

method		Method for updating atomic coordinates. For molecular dynamics, the options are **velocity_verlet**: constant energy simulation. **temperature_control**: constant temperature simulation.
dt		Time step. Defaults to 100 au (about 2.4 fs).
thermostat		Block for specifying the thermostat.
	temp	Temperature.
	qmass	Thermostat parameter Q, corresponding to an effective mass. This parameter is required for constant-temperature simulations.

3.8.3 Stress tensor

Calculation of the stress tensor can be specified in the block stress in the **structure_evolution** block.

```
structure_evolution{
  stress{
    sw_stress=1
  }
}
```

stress		Calculation of stress tensor.
	sw_stress	Switch that specifies whether to calculate the stress tensor. Options are {on, off}.

3.9 Postproccesing

3.9.1 Density of states (DOS)

The density of states (DOS) can be calculated after the SCF iterations converge. Calculation of DOS is specified in the **dos** block in the **postprocessing** block as follows.

```
postprocessing{
    dos{
        sw_dos = on
        method = gaussian
        deltaE_dos = 1e-4 hartree
    }
}
```

The following variables are available in the **dos** block.

sw_dos	Switch that specifies whether to calculate the DOS. Options are **on** and **off**.
method	Method for calculating the density of states. Options are **gaussian**: simple Gaussian broadening. **tetrahedral**: accurate calculation based on the tetrahedral method. Note that the conditions in which the tetrahedral method is available are limited (See below).
deltaE_dos	Specifies the broadening used in the DOS calculation, in units of energy. Defaults to 1e−4 hartee.

The tetrahedral method is available when

- mesh method is employed for the k-sampling,

```
        accuracy{
            ksampling{
                method = mesh
            }
        }
```

- tetrahedral method is used for the smearing

```
        accuracy{
            smearing{
                method = tetrahedral
            }
        }
```

If the above conditions are not satisfied, the DOS is calculated by the Gaussian method.

3.9.2 Charge density

Charge density is calculated in reciprocal space during the SCF calculation, but you can convert the charge density to real space by inverse Fourier transformation. It can then be visualized using the PHASE-Viewer. The block **charge** in the block **postprocessing** is used to output the charge density to real space.

```
postprocessing{
    charge{
        sw_charge_rspace = on
        filetype = cube
    }
}
```

The following variables are set in the **charge** block.

sw_charge_rspace	Switch that specifies whether to generate the charge density in real space. Options are **on** or **off**.

filetype	Specifies the file format of the charge density data. Options are **density_only**: only the charge density is written to the file. (default) **cube**: charge density is stored in Gaussian cube format. Using the **cube** option is recommended.
title	Title of the Gaussian Cube file. Double quotes " " are used to include spaces in the title.

If **filetype=cube**, it is recommended to change the filename of the charge density file. The filename can be specified in the file "file_names.data" as shown below.

```
&fnames
...
F_CHR = './nfchr.cube'
/
```

The default name of the file is "nfchr.data." If spin polarization is considered and "nfchr.cube" is set to the filename, two cube files named "nfchr.up.cube" and "nfchr.down.cube," which correspond to densities of up and down spins, respectively, will be generated.

3.10 Print level

PHASE outputs a log file named "output000." The number "000" will increase as execution continues. The print level of the log file is specified in the block **printoutlevel**.

```
printoutlevel{
    base = 1
}
```

In the **printoutlevel** block, the variables that control the print level are listed. These variables are set to either 0, 1, or 2; a large value causes more detail to be printed. Default values of all these variables are 1. Representative variables are listed below.

base	Print level for the entire calculation. This variable becomes the default value for other variables that specify the print level.
timing	Print level of time profiles.
input	Print level of input.
solver	Print level of wavefunction solver.
spg	Print level of space group.

Note that **base=2** prints a large amount of output, making the log file hard to read. Because this additional information is mainly for debugging, it is not recommended for general users to use **base=2**.

4. Examples for basic functions

4.1 Total energy calculation

Calculation of total energy is one of the most basic functions of PHASE. By calculating total energy using several different lattice constants, the equilibrium lattice constant and bulk modulus can be obtained, and stability of crystals at absolute zero temperature can be evaluated.

4.1.1 Input parameters

Here we introduce an example total energy calculation. The target system, a silicon crystal (diamond structure) composed of eight silicon atoms, is shown in Figure 4.1 エラー! 参照元が見つかりません。 .

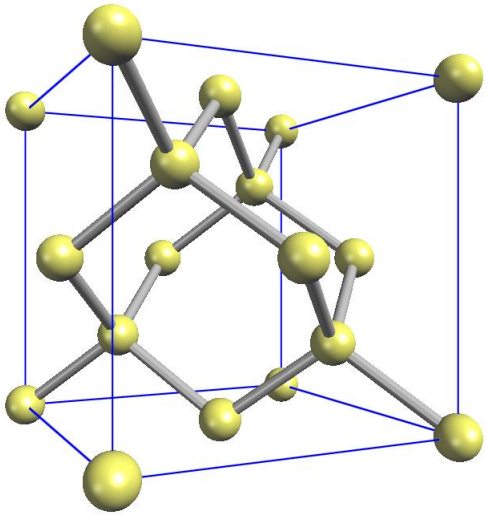


Figure 4.1 Diamond structure composed of silicon atoms

Input files of the calculation are specified in the file "file_names.data." These files are specified as follows.

```
&fnames
 F_INP     = './input_scf_Si8.data'
 F_POT(1)  = '../pp/Si_ldapw91_nc_01.pp'
  ...
 F_CHR     = './nfchr.cube'
&end
```

To execute PHASE, you need to specify a pseudopotential file and an input file to F_POT(1) and F_INP. "Si_ldapw91_nc_01.pp" is a pseudopotential file.

Here we describe the input parameter file "input_scf_Si8.data." The control block specifies calculation conditions for the entire calculation. For example, cpumax specifies the limit of calculation time.

```
Control{
        condition = initial
        cpumax = 3600 sec      ! {sec|min|hour|day}
}
```

The **accuracy** block contains parameters related to computational accuracy.

```
accuracy{
        cutoff_wf =   9.00  rydberg
        cutoff_cd =  36.00  rydberg
        num_bands = 20
        ksampling{
                method = mesh ! {mesh|file|directin|gamma}
                mesh{  nx = 4, ny =  4, nz =  4  }
        }
        ...
        xctype = ldapw91
        scf_convergence{
             delta_total_energy = 1.e-12 hartree
             succession = 3      !default value = 3
        }
        ...
}
```

Parameters **cutoff_wf** and **cutoff_cd** indicate that cutoff energies for wavefunctions and the charge density distribution are 9.0 Ry and 36.0 Ry, respectively. The parameter **num_bands** specifies the number of the energy level. Since this system has eight Si atoms and a Si atom has four valence electrons, the number of total occupied energy levels is obtained by $8 \times 4/2 = 16$. (This is divided by two because up- and down-spin electrons occupy the same energy level). Therefore, **number_bands** must be larger than 16. The **ksampling** block is used to specify a method of k-point sampling. In this example, a $4 \times 4 \times 4$ mesh was used for k-point sampling. **xctype = ldapw91** specifies using an LDA-type exchange-correlation energy. **scf_convergence** is used to specify a convergence criterion. In this example, SCF iterations were terminated when the difference in total energies was less than 10^{-12} Hartree, three times in succession.

The **structure** block is used to input the crystal structure. Default units are atomic units (The unit of length is Bohr).

```
structure{
        unit_cell_type = primitive
        unit_cell{
                a_vector =  10.26    0.00    0.00
                b_vector =   0.00   10.26    0.00
                c_vector =   0.00    0.00   10.26
        }
        atom_list{
                coordinate_system = internal ! {cartesian|internal}
                atoms{
                #default weight = 1, element = Si, mobile = 1
                #tag   rx       ry       rz
                      0.125    0.125    0.125
                     -0.125   -0.125   -0.125
                      0.125    0.625    0.625
                     -0.125   -0.625   -0.625
                      0.625    0.125    0.625
                     -0.625   -0.125   -0.625
                      0.625    0.625    0.125
                     -0.625   -0.625   -0.125
                }
        }
        element_list{ #tag element  atomicnumber
                              Si       14
        }
}
```

The **atom_list** specifies atomic species, internal coordinates, and whether the atomic position is fixed.
The **element_list** defines the element and its atomic number.

The **postprocessing** block is used to specify parameters related to post-processing.

```
postprocessing{
        ...
        charge{
                sw_charge_rspace    = ON
                filetype            = cube  !{cube|density_only}
                title  = "This is a title line for the bulk Si"
        }
}
```

The **charge** block is used to output charge density. The charge density is saved to the file specified by the F_CHR keyword in "file_names.data." **filetype = cube** specifies the Gaussian cube format. In this case, the file extension of the cube file must be *.cube. Gaussian cube files can be visualized using PHASE Viewer and other visualization software.

4.1.2 Execution of calculations

You execute PHASE as follows:

```
 % mpirun -np NP ../../bin/phase ne=NE nk=NK
```

Here NP is the number of processors used for the calculation, NE is the degree of parallelism for energy levels, and NK is the degree of parallelism for k-points. Note that NE times NK must be equal to NP (NP = NE×NK). If you use only one processor, you can execute PHASE as follows:

```
 % mpirun  ../../bin/phase
```

You can check the progress of the SCF calculation by checking the log file "Output000." Values of total energy can be extracted using the grep command as follows:

```
% grep TOTAL output000
```

The following results were obtained for the sample calculation of Si8.

```
 TOTAL ENERGY FOR     1 -TH ITER=    -30.851502112276   edel =  -0.308515D+02
 TOTAL ENERGY FOR     2 -TH ITER=    -31.428857832957   edel =  -0.577356D+00
 TOTAL ENERGY FOR     3 -TH ITER=    -31.547875271353   edel =  -0.119017D+00
 TOTAL ENERGY FOR     4 -TH ITER=    -31.575313743308   edel =  -0.274385D-01
 TOTAL ENERGY FOR     5 -TH ITER=    -31.582591031973   edel =  -0.727729D-02
 TOTAL ENERGY FOR     6 -TH ITER=    -31.585296287695   edel =  -0.270526D-02
 TOTAL ENERGY FOR     7 -TH ITER=    -31.586566551584   edel =  -0.127026D-02
 TOTAL ENERGY FOR     8 -TH ITER=    -31.587203940144   edel =  -0.637389D-03
 TOTAL ENERGY FOR     9 -TH ITER=    -31.587536187844   edel =  -0.332248D-03
 TOTAL ENERGY FOR    10 -TH ITER=    -31.587714367315   edel =  -0.178179D-03
 TOTAL ENERGY FOR    11 -TH ITER=    -31.587811775875   edel =  -0.974086D-04
 TOTAL ENERGY FOR    12 -TH ITER=    -31.587865777306   edel =  -0.540014D-04
 TOTAL ENERGY FOR    13 -TH ITER=    -31.587896135394   edel =  -0.303581D-04
 TOTAL ENERGY FOR    14 -TH ITER=    -31.587913347827   edel =  -0.172124D-04
 TOTAL ENERGY FOR    15 -TH ITER=    -31.587923218322   edel =  -0.987050D-05
 TOTAL ENERGY FOR    16 -TH ITER=    -31.587928921902   edel =  -0.570358D-05
 TOTAL ENERGY FOR    17 -TH ITER=    -31.587932250599   edel =  -0.332870D-05
 TOTAL ENERGY FOR    18 -TH ITER=    -31.587934208228   edel =  -0.195763D-05
 TOTAL ENERGY FOR    19 -TH ITER=    -31.587935369846   edel =  -0.116162D-05
 TOTAL ENERGY FOR    20 -TH ITER=    -31.587936064369   edel =  -0.694523D-06
 TOTAL ENERGY FOR    21 -TH ITER=    -31.587937128483   edel =  -0.106411D-05
 TOTAL ENERGY FOR    22 -TH ITER=    -31.587937146269   edel =  -0.177857D-07
 TOTAL ENERGY FOR    23 -TH ITER=    -31.587937147223   edel =  -0.953783D-09
 TOTAL ENERGY FOR    24 -TH ITER=    -31.587937147361   edel =  -0.138854D-09
 TOTAL ENERGY FOR    25 -TH ITER=    -31.587937147369   edel =  -0.733991D-11
 TOTAL ENERGY FOR    26 -TH ITER=    -31.587937147369   edel =  -0.358824D-12
 TOTAL ENERGY FOR    27 -TH ITER=    -31.587937147369   edel =  -0.117240D-12
```

The above results indicate that the total energy is converging.

4.1.3 Output of calculation results

The calculated total energy is printed to the F_ENF file. In the example Si8 calculation, the results were printed to the F_ENF file (nfefn.data) as follows:

```
iter_ion, iter_total, etotal, forcmx
1 12 -31.587937147369 0.0000004495
```

After the calculation is finished, the charge density file "nfchr.cube" is generated. The charge density distribution is shown in Figure 4.2. Note: You may need to fix the number of atoms written in the cube file.

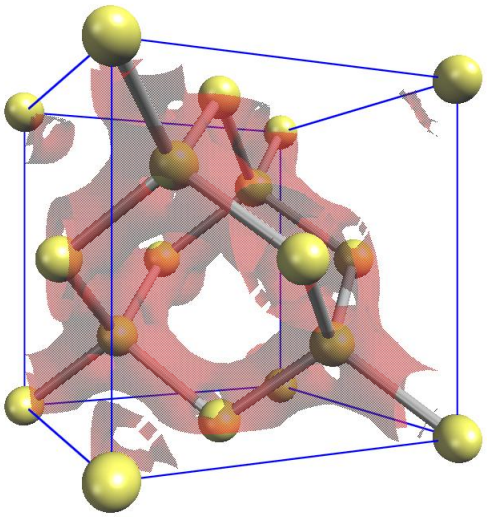


Figure 4.2 Visualized charge density distribution of a silicon crystal

4.2 Calculations using symmetry properties

PHASE can reduce its computational costs by using the symmetry of crystals. A type of symmetry can be automatically identified, or you can manually specify the generator. Atomic positions can be specified in either of two ways: basic lattice or Bravais lattice. To choose between these, you need to set the **unit_cell_type** to "primitive" or "Bravais."

4.2.1 Input parameters

4.2.1.1 Specifying the unit cell

(1) Specifying the unit cell by basic lattice

```
    unit_cell_type = primitive
    unit_cell{
         #units bohr
         a_vector =  0.00000  5.13000  5.13000
         b_vector =  5.13000  0.00000  5.13000
         c_vector =  5.13000  5.13000  0.00000
    }
```

This specification is available not only to **unit_cell_type=primitive** but also to **unit_cell_type =Bravais**.

(2) Specifying the unit cell by lattice parameters

```
  unit_cell_type = Bravais
  unit_cell{
       #units bohr
       a = 10.26, b = 10.26, c = 10.26
       alpha = 90, beta = 90, gamma = 90
  }
```

This specification is enabled only when **unit_cell_type=Bravais**. When the Bravais lattice is used, the basic lattice is automatically determined on the basis of a specification of symmetry. Note that the actual calculation will be carried out using the automatically determined basic lattice; thus, you need to specify atomic positions, number of k-points, symmetry of k-points, and other parameters that are suitable for the basic lattice.

If the **unit_cell_type** is "Bravais," you should input only the atom on a lattice point. For example, for a body-centered lattice, input only the atom on (0, 0, 0) and do not input the atom on (0.5, 0.5, 0.5). The type of lattice is specified by the **lattice_system** variable (See Table 4.1) For a rhombohedral crystal, you need to specify the lattice parameters for the corresponding hexagonal crystal. The relationship between the lattice vectors of rhombohedral and hexagonal crystals is shown in Figure 4.3.

Table 4.1 The Bravais lattices and crystal systems

crystal systems	lattice parameters	description of unit cell	type of lattice	keywords for the **lattic_system**
cubic (c)	a	a = a, b = a, c = a alpha = 90, beta = 90, gamma = 90	primitive (P) face-centered (F) body-centered	primitive face-centered bodycentered

			(I)	
tetragonal (t)	a, c	a = a, b = a, c = c alpha = 90, beta = 90, gamma = 90	primitive (P) body-centered (I)	primitive body-centered
orthorhombic (o)	a, b, c	a = a, b = b, c = c alpha = 90, beta = 90, gamma = 90	primitive (P) base-centered (C) face-centered (F) body-centered (I)	primitive base-centered face-centered body-centered
hexagonal (h)	a, c	a = a, b = a, c = c alpha = 90, beta = 90, gamma = 120	primitive (P)	hexagonal
trigonal (h) rhombohedral	a, c	a =a, b = a, c = c alpha = 90, beta = 90, gamma = 120	rhombohedral (R) primitive (P)	rhombohedral hexagonal
monoclinic (m)	a, b, c β	a = a, b = b, c = c alpha = 90, beta = β, gamma = 90	primitive (P) base-centered (C)	primitive base-centered
triclinic (a)	a, b, c α, β, γ	a = a, b = b, c = c alpha = α, beta = β, gamma = γ	primitive (P)	primitive

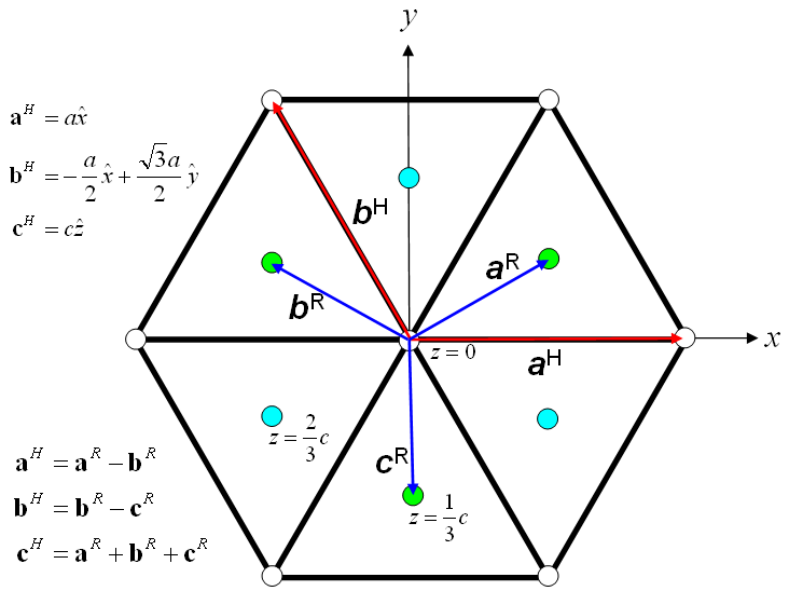


Figure 4.3 Relationship between the lattice vectors of rhombohedral and hexagonal crystals (view along the c-axis of hexagonal). a^H, b^H, c^H are the lattice vectors for hexagonal, and a^R, b^R, c^R are the lattice vectors for rhombohedral crystals

Table 4.2 Primitive translation vectors for Bravias lattice

the Bravais lattices	a	b	c
primitive cubic (cP)	$a\hat{x}$	$a\hat{y}$	$a\hat{z}$
face-centered cubic (cF)	$\frac{a}{2}(\hat{y} + \hat{z})$	$\frac{a}{2}(\hat{x} + \hat{z})$	$\frac{a}{2}(\hat{x} + \hat{y})$
body-centered	$\frac{a}{2}(-\hat{x} + \hat{y} + \hat{z})$	$\frac{a}{2}(\hat{x} - \hat{y} + \hat{z})$	$\frac{a}{2}(\hat{x} + \hat{y} - \hat{z})$

cubic (cI)			
primitive tetragonal (tP)	$a\hat{\boldsymbol{x}}$	$a\hat{\boldsymbol{y}}$	$c\hat{\boldsymbol{z}}$
body-centered tetragonal (tI)	$\frac{1}{2}(-a\hat{\boldsymbol{x}} + a\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} - a\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + a\hat{\boldsymbol{y}} - c\hat{\boldsymbol{z}})$
primitive orthorhombic (oP)	$a\hat{\boldsymbol{x}}$	$b\hat{\boldsymbol{y}}$	$c\hat{\boldsymbol{z}}$
base-centered orthorhombic (oC)	$\frac{1}{2}(a\hat{\boldsymbol{x}} - b\hat{\boldsymbol{y}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}})$	$c\hat{\boldsymbol{z}}$
face-centered orthorhombic (oF)	$\frac{1}{2}(b\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + c\hat{\boldsymbol{y}})$
body-centered orthorhombic (oI)	$\frac{1}{2}(-a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} - b\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}} - c\hat{\boldsymbol{z}})$
primitive hexagonal (hP)	$a\hat{\boldsymbol{x}}$	$a(-\frac{1}{2}\hat{\boldsymbol{x}} + \frac{\sqrt{3}}{2}\hat{\boldsymbol{y}})$	$c\hat{\boldsymbol{z}}$
primitive rhombohedral (hR)	$\frac{a}{2}\hat{\boldsymbol{x}} + \frac{a}{2\sqrt{3}}\hat{\boldsymbol{y}} + \frac{1}{3}c\hat{\boldsymbol{z}}$	$-\frac{a}{2}\hat{\boldsymbol{x}} + \frac{a}{2\sqrt{3}}\hat{\boldsymbol{y}} + \frac{1}{3}c\hat{\boldsymbol{z}}$	$-\frac{a}{\sqrt{3}}\hat{\boldsymbol{y}} + \frac{1}{3}c\hat{\boldsymbol{z}}$
primitive monoclinic (mP)	$a\hat{\boldsymbol{x}}$	$b\hat{\boldsymbol{y}}$	$c(\cos\beta\,\hat{\boldsymbol{x}} + \sin\beta\,\hat{\boldsymbol{z}})$
base-centered monoclinic (mC)	$\frac{1}{2}(a\hat{\boldsymbol{x}} - b\hat{\boldsymbol{y}})$	$\frac{1}{2}(a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}})$	$c(\cos\beta\,\hat{\boldsymbol{x}} + \sin\beta\,\hat{\boldsymbol{z}})$
primitive triclinic (aP)	$a\hat{\boldsymbol{x}}$	$b(\cos\gamma\,\hat{\boldsymbol{x}} + \sin\gamma\,\hat{\boldsymbol{y}})$	$c\left(\cos\beta\,\hat{\boldsymbol{x}} + \frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}\right)\hat{\boldsymbol{y}} + \sqrt{1 - \frac{\cos^2\alpha + \cos^2\beta - 2\cos\alpha\cos\beta\cos\gamma}{\sin^2\gamma}}\,\hat{\boldsymbol{z}}$

4.2.1.2 Specifying symmetry

Symmetry can be specified in three ways: the **crystal_structure** variable, automatic identification of the symmetry operation, and specification of the generator.

1. The crystal_structure variable

You can specify a type of crystal structure by the **crystal_structure** variable. Options are **diamond**, **hexagonal**, **fcc**, **bcc**, and **simple_cubic**. For the Si crystal, **diamond** is specified.

（1） Automatic identification of the symmetry operation

If the **method** variable is set to **automatic**, then symmetry is automatically identified. The **lattice_system** variable in the **tspace** block should be specified if the type of lattice is not primitive.

```
symmetry{
    method = automatic
    tspace{
       lattice_system = facecentered
   !{rhombohedral|hexagonal|primitive|facecentered|bodycentered|basecentered}
    }
}
```

（2） Specification of generator

The generator is specified by the **tspace** block. For the Si crystal, the **tspace** block is specified as follows:

```
 tspace{
    lattice_system = facecentered
!{rhombohedral|hexagonal|primitive|facecentered|bodycentered|basecentered}
    num_generators = 3
    generators{
    #tag rotation  tx   ty   tz
          IE      0    0    0
          C31+    0    0    0
          C4X+    1/4  1/2  3/4
    }
 }
```

The lattice_system=facecenterd indicates that the symmetry is face-centered, and num_generators=3 sets the number of generators to three. In the **generators** block, IE, C31+, and C4X+ are specified as the generators.

Here we describe how to specify the generators. The rotational operation is specified by the following codes. Each line corresponds to one rotation. The numbers in the first column or the symbol in the second column is used in the rotation column in the **generators** block to specify the symmetry operations. The third to fifth columns represent rotational operations. For trigonal and hexagonal lattices, W represents X-Y. The rotation can be specified either by the numbers in the first column or by the symbol in the second column.

For the trigonal and hexagonal,

```
 1   E     X  Y  Z          13  IE    -X -Y -Z
 2   C6+   W  X  Z          14  IC6+  -W -X -Z
 3   C3+  -Y  W  Z          15  IC3+   Y -W -Z
 4   C2   -X -Y  Z          16  IC2    X  Y -Z
 5   C3-  -W -X  Z          17  IC3-   W  X -Z
 6   C6-   Y -W  Z          18  IC6-  -Y  W -Z
 7   C211 -W  Y -Z          19  IC211  W -Y  Z
 8   C221  X  W -Z          20  IC221 -X -W  Z
 9   C231 -Y -X -Z          21  IC231  Y  X  Z
10   C212  W -Y -Z          22  IC212 -W  Y  Z
11   C222 -X -W -Z          23  IC222  X  W  Z
12   C232  Y  X -Z          24  IC232 -Y -X  Z
```

and for the others,

```
 1   E     X  Y  Z          25  IE    -X -Y -Z
 2   C2X   X -Y -Z          26  IC2X  -X  Y  Z
 3   C2Y  -X  Y -Z          27  IC2Y   X -Y  Z
 4   C2Z  -X -Y  Z          28  IC2Z   X  Y -Z
 5   C31+  Z  X  Y          29  IC31+ -Z -X -Y
 6   C32+ -Z  X -Y          30  IC32+  Z -X  Y
 7   C33+ -Z -X  Y          31  IC33+  Z  X -Y
 8   C34+  Z -X -Y          32  IC34+ -Z  X  Y
 9   C31-  Y  Z  X          33  IC31- -Y -Z -X
10   C32-  Y -Z -X          34  IC32- -Y  Z  X
11   C33- -Y  Z -X          35  IC33-  Y -Z  X
12   C34- -Y -Z  X          36  IC34-  Y  Z -X
13   C2A   Y  X -Z          37  IC2A  -Y -X  Z
14   C2B  -Y -X -Z          38  IC2B   Y  X  Z
15   C2C   Z -Y  X          39  IC2C  -Z  Y -X
16   C2D  -X  Z  Y          40  IC2D   X -Z -Y
17   C2E  -Z -Y -X          41  IC2E   Z  Y  X
18   C2F  -X -Z -Y          42  IC2F   X  Z  Y
19   C4X+  X -Z  Y          43  IC4X+ -X  Z -Y
```

```
              20   C4Y+  Z  Y -X                    44  IC4Y+ -Z -Y  X
              21   C4Z+ -Y  X  Z                    45  IC4Z+  Y -X -Z
              22   C4X-  X  Z -Y                    46  IC4X- -X -Z  Y
              23   C4Y- -Z  Y  X                    47  IC4Y-  Z -Y -X
              24   C4Z-  Y -X  Z                    48  IC4Z- -Y  X -Z
```

However, translation operations, which are associated with a rotational operation, are specified by the **tx**, **ty**, **tz** columns in the **generators** table. Fractional numbers refer to the lattice vectors being used.

4.2.1.3 Using inversion symmetry

If the system has inversion symmetry, this can be used to reduce the computational cost.

2. Using no inversion symmetry

If the variable **sw_inversion** is "off," inversion symmetry is not used. In this case, the **atom_list** block for the Si crystal is specified as below.

```
        atom_list{
             atoms{
                  !#tag  rx        ry        rz    element
                       0.125     0.125     0.125      Si
                      -0.125    -0.125    -0.125      Si
             }
        }
```

3. Using inversion symmetry

To use inversion symmetry, set the **sw_inversion** variable to "on." For example, the following atomic coordinates have inversion symmetry whose center is the origin; thus, the computational cost can be reduced using the **sw_inversion** variable.

```
     atom_list{
          coordinate_system = internal ! {cartesian|internal}
          atoms{
          #units  !{angstrom(cartesian) | bohr(cartesian)}
          #tag  rx        ry        rz      weight    element    mobile
               0.125     0.125     0.125       1          Si        1
              -0.125    -0.125    -0.125       1          Si        1
               }
     }
```

Set the **sw_inversion** to "on" in the **symmetry** block and specify the atomic coordinates as below.

```
     atom_list{
          coordinate_system = internal ! {cartesian|internal}
          atoms{
          #units  !{angstrom(cartesian) | bohr(cartesian)}
          #tag  rx        ry        rz      weight    element    mobile
               0.125     0.125     0.125       2          Si        1
               }
     }
     symmetry{
          sw_inversion = on
     }
```

In the above example, the **weight** column of the atom is set to 2. This indicates that the atoms will be copied by the inversion symmetry operation. It is recommended to use this option if the system has inversion symmetry. The origin is the center of the inversion symmetry operation. Note that if this option is applied to a system that has no inversion symmetry, an incorrect calculation will be carried out.

4.2.2 Example: Silicon crystal (Si2)

A diamond-structured silicon crystal (Figure 4.4) has two unique atoms in its unit cell. Here we introduce a sample input for the silicon crystal (hereafter referred to as Si2). This input file is in "sample/Si2/."

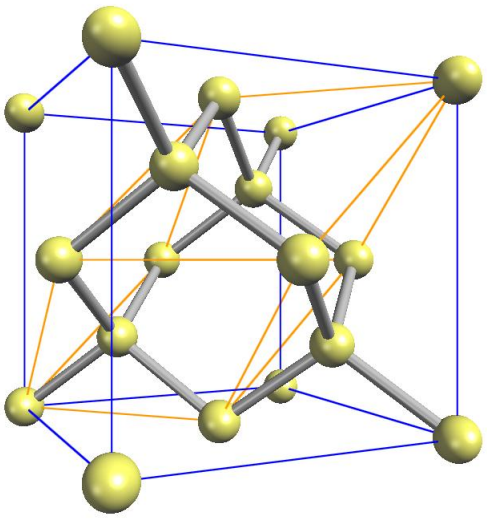


Figure 4.4 Atomic structure of Si_2. The orange lines represent the primitive lattice composed of two silicon atoms.

4. SCF calculation

First, perform the SCF calculation to obtain the charge density. The input file is "sample/Si2/scf."

In the file "file_names.data," the input parameter file and the pseudopotential files are specified.

```
F_INP    = './input_scf_Si.data'
F_POT(1) = '../../pp/Si_ldapw91_nc_01.pp'
F_CHGT   = '../scf/nfchgt.data'
   ...
```

In this input parameter file, "diamond" is specified for the **crystal_structure** variable.

```
accuracy{
        cutoff_wf =   9.00  rydberg
        cutoff_cd =  36.00  rydberg
        num_bands = 8
}

structure{
        unit_cell_type = Bravais
        unit_cell{
             a = 10.26, b = 10.26, c = 10.26
             alpha = 90, beta = 90, gamma = 90
        }
        symmetry{
             crystal_structure = diamond
        }
```

```
        atom_list{
             atoms{
                  #tag   rx         ry         rz     element
                       0.125     0.125     0.125       Si
                      -0.125    -0.125    -0.125       Si
             }
        }
}
```

The number of energy levels, **num_bands**, is set to 8 because the number of atoms is two.

Execute PHASE as below.

```
% mpirun  ../../../bin/phase
```

After the calculation is complete, the charge density is output to the file "nfchgt.data," as specified by the **F_CHGT** variable in file_names.data.

5. Density of states

To calculate the density of states (DOS), use an input file like sample/Si2/dos. Execute the calculation in a different directory from the previous directory, where the SCF calculation was done, to avoid overwriting output files.

In the **file_names.data** file, input and output files are specified as follows.

```
F_INP    = './input_dos_Si.data'
F_POT(1) = '../../pp/Si_ldapw91_nc_01.pp'
   ...           ...
F_CHGT   = '../scf/nfchgt.data'
   ...           ...
F_ENERG  = './nfenergy.data'
   ...           ...
```

The data file for charge density, specified by F_CHGT, is an output file created by the SCF calculation.

Input files are input_dos_Si.data and nfchgt.data. The following parameters are specified in the input file input_dos_Si.data.

```
Control{
       condition = fixed_charge
}

accuracy{
        cutoff_wf =   9.00  rydberg
        cutoff_cd =  36.00  rydberg
        num_bands = 8
        ksampling{
                method = mesh
                mesh{  nx = 4, ny =  4, nz =  4   }
        }
        smearing{
                method = tetrahedral
        }
        xctype = ldapw91
        initial_wavefunctions = matrix_diagon
            matrix_diagon{
               cutoff_wf =  9.00  rydberg
            }
        ek_convergence{
```

```
                num_max_iteration = 200
                sw_eval_eig_diff = on
                delta_eigenvalue = 1.e-8 hartree
                succession   = 2
        }
}

postprocessing{
        dos{
           sw_dos      = ON
           method      = tetrahedral   !{ tetrahedral | Gaussian }
           deltaE_dos  = 1.e-3 eV
           nwd_window_width = 10
        }
}
```

The first block in **Control** specifies that the charge density, obtained from the SCF calculation, is fixed. The parameter **ksampling** specifies that k-points are sampled by $4 \times 4 \times 4$ and that **tetrahedral** is employed for smearing. Convergence criteria are set by **ek_convergence**. The **postprocessing** block specifies parameters used for the calculation of DOS based on the tetrahedral method.

Calculate the DOS by using the program **ekcal** and these input files.

```
 % mpirun  ../../../bin/ekcal
```

After the calculation is done, the output file nfenergy.data is generated. In this file, energy eigenvalues are printed for every k-points in order of increasing energy. The header part of the file is shown below.

```
 num_kpoints =    141
 num_bands   =      8
 nspin       =      1
 Valence band max   =   0.233846

=== energy_eigen_values ===
 ik =    1 (  0.000000  0.500000  0.500000 )
     -0.0484324491     -0.0484324491      0.1258095002      0.1258095002
      0.2619554320      0.2619554320      0.6015285289      0.6015285289
=== energy_eigen_values ===
 ik =    2 (  0.000000  0.490000  0.490000 )
     -0.0540717117     -0.0427149546      0.1258687813      0.1258687813
      0.2607026827      0.2633829946      0.6006244013      0.6006244013
=== energy_eigen_values ===
 ik =    3 (  0.000000  0.480000  0.480000 )
     -0.0596299923     -0.0369220783      0.1260465996      0.1260465996
      0.2596226501      0.2649874134      0.5980547648      0.5980547648
=== energy_eigen_values ===
 ik =    4 (  0.000000  0.470000  0.470000 )
     -0.0651046420     -0.0310567694      0.1263428799      0.1263428799
      0.2587131916      0.2667706685      0.5941566835      0.5941566835
=== energy_eigen_values ===
 ik =    5 (  0.000000  0.460000  0.460000 )
     -0.0704931128     -0.0251220735      0.1267574962      0.1267574962
      0.2579721226      0.2687346642      0.5892968047      0.5892968047
```

T The first two lines give the number of k-points and the number of bands. The third line indicates that spin polarization was not considered in this calculation. The fourth line gives the highest valence band energy.

DOS can be plotted by the tool **dos.pl** as below. Here the option **-erange** is used to specify the energy range, where E1 and E2 are the minimum and maximum energy levels, respectively.

```
% dos.pl dos.data -erange=E1,E2
```

This command creates **density_of_states.eps**, a postscript format file for the DOS. If the command is executed with the –**with_fermi** option, a dashed line is drawn at the specified fermi level. Note that the dashed line is drawn at the level of the highest valence band for systems having a band gap.

```
% dos.pl dos.data -erange=-13,5 -with_fermi
```

Figure 4.5 shows the DOS for Si_2.

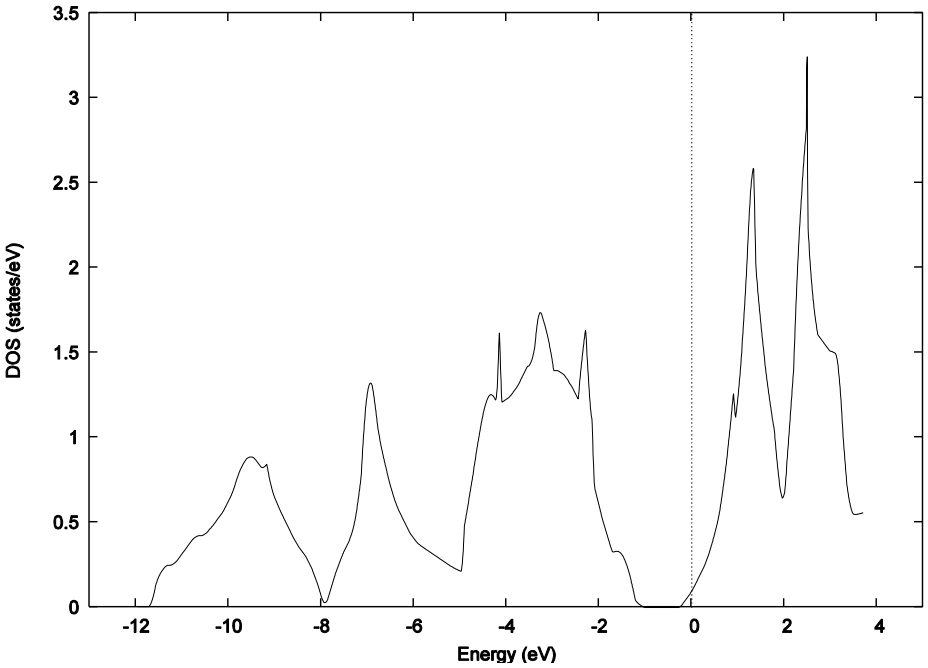


Figure 4.5 Density of states for Si2

6. Band structure

Band structure is calculated by the following procedure. The input file is **sample/Si2/band**.

In **file_names.data**, input and output files are specified as below.

```
F_INP    = './input_band_Si.data'
F_POT(1) = '../../pp/Si_ldapw91_nc_01.pp'
F_KPOINT = '../tools/kpoint.data'
F_CHGT   = '../scf/nfchgt.data'
  ...          ...
```

The above example specifies that the input file is **input_band_Si.data** and that k-points are to be read from **kpoint.data**. The **kpoint.data** file is generated using the tool **band_kpoint.pl** as below. Here **bandkpt_fcc_xglux.in** specifies parameters used to generate the k-points.

```
% band_kpoint.pl bandkpt_fcc_xglux.in
```

Execute the program **ekcal** with these input files.

```
% mpirun  ../../../bin/ekcal
```

By using the tool **band.pl**, band structures can be plotted from the output file **nfenergy.data**. The file **band_structure.eps**, a postscript format file for band structures, is generated by the following command.

```
% band.pl nfenergy.data bandkpt_fcc_xglux.in -erange=E1,E2 -with_fermi
```

In this example, the band structure is plotted in the energy range from E1 = −13 to E2 = 5; these are the same values as in the previous calculation.

```
% band.pl nfenergy.data bandkpt_fcc_xglux.in -erange=-13,5 -with_fermi
```

Figure 4.6 shows the band structure of Si_2.

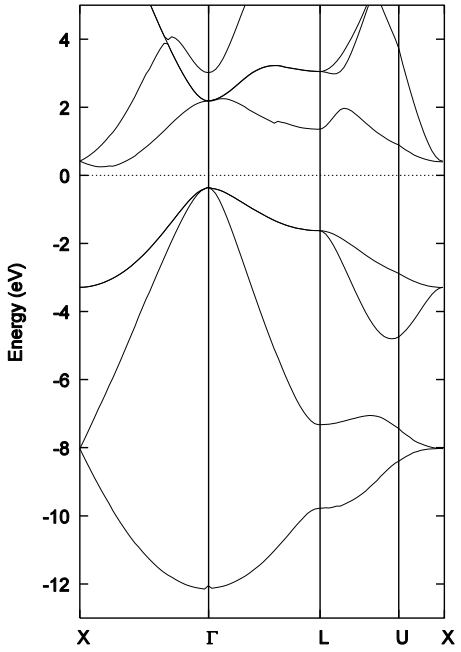


Figure 4.6 Band structure of Si2.

4.3 Spin-polarized calculation

Spin polarization needs to be considered to study ferromagnetic or antiferromagnetic substances. This section describes procedures for spin-polarized calculations. In these examples, body-centered cubic iron and body-centered cubic chrome are employed as examples for ferromagnetic and antiferromagnetic substances, respectively.

4.3.1 Calculations for a ferromagnetic substance

4.3.1.1 Input parameters

As an example of a ferromagnetic substance, the input file for body-centered cubic iron is shown below. This file is in **sample/bcc_Fe**.

```
Control{
        condition = initial
        cpumax = 3 hour
        max_iteration = 250
}

accuracy{
        cutoff_wf =   25  rydberg
        cutoff_cd =  225.00  rydberg
        num_bands =  20
        ksampling{
                method = mesh
                mesh{ nx = 10, ny = 10, nz = 10 }
        }
        smearing{
                method = tetrahedral
        }
        xctype = ggapbe
        scf_convergence{
                delta_total_energy = 1.e-10  hartree
                succession = 3
        }
}

structure{
        unit_cell_type = Bravais
        unit_cell{
                 #units angstrom
                 a = 2.845, b = 2.845, c = 2.845
                 alpha = 90, beta = 90, gamma = 90
        }

        symmetry{
             crystal_structure = bcc
        }

        magnetic_state = ferro

        atom_list{
             atoms{
                !#tag  rx       ry         rz      element
                     0.000     0.000     0.000          Fe
             }
```

```
        }
        element_list{ !#tag element  atomicnumber     zeta  dev
                                 Fe            26    0.275  1.5 }

}

Postprocessing{
        dos{
                sw_dos = ON
                method = tetrahedral
                deltaE = 1.e-4 hartree
                nwd_dos_window_width = 10
        }
        charge{
                sw_charge_rspace    = OFF
                filetype            = cube
                title  = "This is a title line for FM bcc Fe"
        }
}

printlevel{
        base = 1
}
```

7. Specifying the crystal structure

The **crystal_structure** variable is set to "bcc," which means that the crystal structure is body-centered cubic. The unit cell is defined by the Bravais lattice, and only one atom is listed in the **atom_list** block. Note that the atom at body center is not listed. Since the **crystal_structure** is "bcc," PHASE converts the specified lattice to the basic lattice.

8. Specifying spin freedom

To calculate for a ferromagnetic substance, set the variable **magnetic_state** to "ferro."

```
structure{
        magnetic_state =  ferro   !{para|antiferro|ferro}
}
```

In addition, you need to set an initial value of spin polarization for each atom. In the following input file,

```
        element_list{ #tag element  atomicnumber     zeta  dev
                                Fe          26         0.275  1.5
        }
```

the variable **zeta** specifies the initial value of spin polarization $\zeta = (n_\uparrow - n_\downarrow)/(n_\uparrow + n_\downarrow)$, which is the difference between up- and down-spin densities.

4.3.1.2 Output

Transition of spin polarization is printed to the log file **output000**. You can check the change by using the grep command as below.

```
% grep charge output000 | grep NEW | more
```

```
 !*--- input-file style = NEW
 !NEW total charge (UP, DOWN, SUM) =     4.91749982 (+)    3.08250018 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.75677803 (+)    3.24322197 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.64472738 (+)    3.35527262 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.55104317 (+)    3.44895683 (=)    8.00000000
```

```
 !NEW total charge (UP, DOWN, SUM) =     4.47221206 (+)    3.52778794 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.46057861 (+)    3.53942139 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.48476557 (+)    3.51523443 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.52141098 (+)    3.47858902 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.56555794 (+)    3.43444206 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.61364243 (+)    3.38635757 (=)    8.00000000
   ..............................................................
   ..............................................................
   ..............................................................

 !NEW total charge (UP, DOWN, SUM) =     5.11286684 (+)    2.88713316 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     5.11285665 (+)    2.88714335 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     5.11284790 (+)    2.88715210 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     5.11284030 (+)    2.88715970 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     5.11283035 (+)    2.88716965 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     5.11282059 (+)    2.88717941 (=)    8.00000000
```

By the definition of spin polarization $\zeta = (n_\uparrow - n_\downarrow)/(n_\uparrow + n_\downarrow)$, the above result indicates that the spin polarization has converged to $\zeta = 0.2782$.

You can check the change in values between before and after updating spin by using the grep command as below.

```
 % grep charge output000 | more
```

```
 F_CHGT     = ./nfcharge.data                                              opened = false
 !** --- charge preconditioning ---
 !** sw_charge_rspace   =      0
 !** charge_filetype    =      1
 !** charge_title    =
 !** deviation( 1) of the Gauss. distrib. func. for the initial charge construction =    1.50000
 F_CHGT     = ./nfcharge.data
 F_CHGT     = ./nfcharge.data
 !! total_charge =        8.000000 (m_CD_initial_CD_by_Gauss_func)
 !OLD total charge (UP, DOWN, SUM) =     5.10000000 (+)    2.90000000 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.91749982 (+)    3.08250018 (=)    8.00000000
 !OLD total charge (UP, DOWN, SUM) =     4.91749982 (+)    3.08250018 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.75677803 (+)    3.24322197 (=)    8.00000000
 !OLD total charge (UP, DOWN, SUM) =     4.75677803 (+)    3.24322197 (=)    8.00000000
 !NEW total charge (UP, DOWN, SUM) =     4.64472738 (+)    3.35527262 (=)    8.00000000
   ..............................................................
   ..............................................................
   ..............................................................
```

4.3.2 Calculation for an antiferromagnetic substance

To reproduce an antiferromagnetic state, the initial spin must be set to an antiferromagnetic spin configuration. Otherwise, the configuration is more likely to converge to a ferromagnetic state, which is metastable. As described in the section for ferromagnetic substances, the initial value of spin polarization can be specified only for each element. Therefore, to assign different initial spins to these elements, you need to prepare two elements whose pseudopotentials are the same.

4.3.2.1 Input parameters

As an example of an antiferromagnetic substance, the input file for a body-centered cubic chrome is shown below.

The Cr atom is specified as "Cr1" and "Cr2" in the **element_list**.

```
    element_list{
        #tag element  atomicnumber zeta
          Cr1           24     0.3
```

```
        Cr2          24    -0.3
      }
    }
```

The two different elements, Cr1 and Cr2, are defined with different values of spin polarization, zeta = 0.3 and zeta = −0.3, assigned to each. Atomic coordinates are specified as below.

```
    atom_list{
      atoms{
      #tag  rx       ry         rz      element
          0.000    0.000     0.000         Cr1
          0.500    0.500     0.500         Cr2
      }
    }
```

The atom at the origin is defined as Cr1, and the atom at the body center is defined as Cr2.

The **magnetic_state** is set to "ferro" as the spin freedom.

```
      magnetic_state =  ferro   !{para|antiferro|ferro}
```

In the **file_names.data** file, the pseudopotentials are specified as below.

```
&fnames
 F_INP    = './nfinp.data'
 F_POT(1) = '../../Cr_ggapbe_paw_002.gncpp2'
 F_POT(2) = '../../Cr_ggapbe_paw_002.gncpp2'
/
```

By this specification, the same pseudopotential is used for both elements Cr1 and Cr2.

Similarly, you can perform calculations for systems having more complex magnetic states.

4.4 Geometry optimization

PHASE can perform geometry optimizations by calculating the atomic force. This section describes how to use geometry optimization.

4.4.1 Input parameter

An example of an input file for geometry optimization is shown below. Convergence criteria of geometry optimization are specified in the **accuracy** block. The parameter **max_force** specifies the maximum value of atomic force.

```
...
accuracy{
    ...
    max_force = 1.0e-3 hartree/bohr
    ...
}
...
```

The default value of the **max_force** is 10^{-3} hartree/bohr.

The **mobile** attribute is defined in the **atom_list** block to specify whether the atom is optimized. If the flag is 1, the position of the atom is optimized. Set the flag 0 or * to fix the position of the atom.

```
...
structure{
    ...
    atom_list{
    !#tag element  rx     ry     rz     mobile
          Ba       0.0000 0.5000 0.05   0
          O        0.5000 0.0000 0.05   1
          Ba       0.5000 0.0000 0.15   1
          O        0.0000 0.5000 0.15   1
          ...
    }
}
...
```

In the above example, the first Ba atom is fixed, and the remaining atoms are optimized.

The parameters used for geometry optimization are specified in the **structure_evolution** block.

```
...
structure_evolution{
    method = quench
    dt = 50
    ...
}
...
```

method	A method of structure relaxation. Options are: **quench**: quenched MD method (default) **cg**: CG method **gdiis**: GDIIS method **bgfs**: BFGS method
dt	Time step for the structure relaxation. Appropriately large value get the iterations converged faster, but too much large value may make the calculation incorrect. Defaults to 100 au.

Because the GDIIS or BFGS method does not work stably when atomic forces are large, the quenched MD or CG method is employed for earlier steps, and the method will be switched to the GDIIS (or BFGS) after the force become small enough.

The initial method for the earlier steps and the criterion for switching the method to GDIIS (or BFGS) are specified by the variables **initial_method** and **c_forc2gdiis**, respectively.

```
...
structure_evolution{
    method = gdiis
    dt = 50
    gdiis{
        initial_method = cg
        c_forc2gdiis = 0.0025 hartree/bohr
    }
}
...
```

The block **gdiis** is common to GDIIS and BFGS. Default values are **quench** for **initial_method** and 0.0025 hartree/bohr for **c_forc2gdiis**.

4.4.2 Output

If geometry optimization is performed, changes in the energy and maximum atomic force are printed to the **F_ENF file** (default name: **nfefn.data**), and trajectories of atomic positions are stored in the **F_DYNM file** (default name: **nfdynm.data**).

4.4.3 Example: geometry optimization of a silicon crystal

Here we introduce an example of geometry optimization for a silicon crystal. In this example, atomic positions are manually displaced from their stable positions, and then geometry optimization is performed. The input file is in **sample/Si2/relax**.

9. Input files

In the **file_names.data** file, **input_relx_Si.data** is an input file, and **nfdynm.data** is an output file in which atomic positions and the atomic force are stored.

```
 F_INP    = './input_relax_Si.data'
  ...
 F_DYNM   = './nfdynm.data'
  ...
```

In the input file **input_relax_Si.data,** atomic positions are displaced from their stable positions by changing the interval from 0.125 to 0.130. To optimize the atomic positions, the **mobile** variables are set to "yes."

```
structure{
          ...
        atom_list{
             atoms{
                #tag     rx         ry         rz     element mobile
                       0.130      0.130      0.130      Si    yes
                      -0.130     -0.130     -0.130      Si    yes
             }
       }
```

```
}
```

The **accuracy** block specifies the convergence criterion for the maximum atomic force.

```
accuracy{
        force_convergence{
                max_force = 1.0e-3
        }
}
```

10. Calculation results

The output file for geometry optimization, nfdynm.data, is shown below.

```
#
#   a_vector =         0.0000000000        5.1300000000        5.1300000000
#   b_vector =         5.1300000000        0.0000000000        5.1300000000
#   c_vector =         5.1300000000        5.1300000000        0.0000000000
#   ntyp =        1 natm =        2
# (natm->type)     1    1
# (speciesname)     1 :   Si
#
 cps and forc at (iter_ion, iter_total =     1      34 )
    1    1.333800000    1.333800000    1.333800000   -0.010794   -0.010794   -0.010794
    2   -1.333800000   -1.333800000   -1.333800000    0.010794    0.010794    0.010794
 cps and forc at (iter_ion, iter_total =     2      53 )
    1    1.331707297    1.331707297    1.331707297   -0.010402   -0.010402   -0.010402
    2   -1.331707297   -1.331707297   -1.331707297    0.010402    0.010402    0.010402
 cps and forc at (iter_ion, iter_total =     3      75 )
    1    1.327597870    1.327597870    1.327597870   -0.009614   -0.009614   -0.009614
    2   -1.327597870   -1.327597870   -1.327597870    0.009614    0.009614    0.009614
 cps and forc at (iter_ion, iter_total =     4     100 )
    1    1.321624355    1.321624355    1.321624355   -0.008433   -0.008433   -0.008433
    2   -1.321624355   -1.321624355   -1.321624355    0.008433    0.008433    0.008433
 cps and forc at (iter_ion, iter_total =     5     127 )
    1    1.314015753    1.314015753    1.314015753   -0.006865   -0.006865   -0.006865
    2   -1.314015753   -1.314015753   -1.314015753    0.006865    0.006865    0.006865
 cps and forc at (iter_ion, iter_total =     6     155 )
    1    1.305076108    1.305076108    1.305076108   -0.004930   -0.004930   -0.004930
    2   -1.305076108   -1.305076108   -1.305076108    0.004930    0.004930    0.004930
 cps and forc at (iter_ion, iter_total =     7     184 )
    1    1.295180554    1.295180554    1.295180554   -0.002671   -0.002671   -0.002671
    2   -1.295180554   -1.295180554   -1.295180554    0.002671    0.002671    0.002671
 cps and forc at (iter_ion, iter_total =     8     213 )
    1    1.284767108    1.284767108    1.284767108   -0.000159   -0.000159   -0.000159
    2   -1.284767108   -1.284767108   -1.284767108    0.000159    0.000159    0.000159
```

The first lines beginning with # contain a part of the input data. The next line gives the number of the optimization cycles and the total number of SCF iterations. Therefore, the above output shows that the wavefunctions were updated 34 times in the first optimization cycle. Convergence criteria for the SCF calculation are specified in the same manner as described in Section 3.

The next two lines contain the atomic number, atomic position (x,y,z coordinates, in units of bohr), and atomic forces (x,y,z components, in units of hartree/bohr). The above results show that the atomic forces drastically decreased. In the final step, all atomic forces are below the specified threshold, and the optimization is terminated.

4.5 Calculation of surface

4.5.1 How to calculate surface

Strictly speaking, PHASE cannot treat finite systems, including surfaces, because periodic boundary conditions must be applied. However, you can treat a system as a surface by creating a "vacuum layer" between slabs. The vacuum layer should be sufficiently large to avoid interactions between the surface and the bottom of the slab. Typically, a vacuum layer with a thickness of 10Å is employed.

Here we introduce an example calculation for a hydrogen-terminated silicon surface. Figure 4.7 shows the slab model for the silicon surface. The Si atoms on the bottom are terminated by artificial hydrogen atoms.

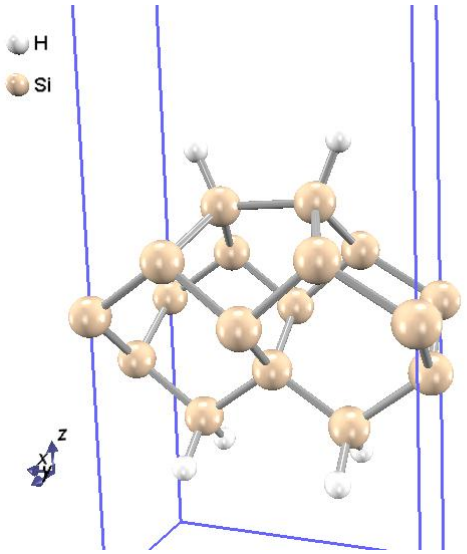


Figure 4.7 Atomic structure for the hydrogen terminated Si(001)-p(2×1) surface

The following shows the **file_names.data** of this example.

```
&fnames
 F_INP    = './input_SiH2x1.data'
 F_POT(1) = '../pp/Si_ldapw91_nc_01.pp'
 F_POT(2) = '../pp/H_ldapw91_nc_01.pp'
  ..............................
&end
```

Pseudopotentials for Si and H atoms are specified by **F_POT(1)** and **F_POT(2).**

Input parameters are described below.

The parameters for k-point sampling are specified as follows:

```
accuracy{
        cutoff_wf =  15.00  rydberg
        cutoff_cd =  60.00  rydberg
        num_bands =  25
        ksampling{
                method = monk  ! {mesh|file|directin|gamma}
                mesh{  nx = 2, ny =  4, nz =  1  }
                kshift{ k1 = 0.5, k2 = 0.5, k3 = 0.0 }
```

```
        }

        ........................
}
```

Since this example is a slab model, only one k-point is sampled in the k_z direction.

```
structure{
        unit_cell_type = primitive
        unit_cell{
                a_vector =  14.512      0.000      0.000
                b_vector =   0.000      7.256      0.000
                c_vector =   0.000      0.000     30.784
        }
        symmetry{}

        magnetic_state = para   !{para|af|ferro}

        atom_list{
            coordinate_system = internal
            atoms{
                #default weight = 1, element = Si, mobile = 0
                #tag   rx       ry       rz         element
                       0.26177  0.50000  0.65651       H
                       0.73823  0.50000  0.65643       H
                       0.34138  0.50000  0.56971
                       0.65858  0.50000  0.56966
                       0.26229  0.00000  0.49388
                       0.73763  0.00000  0.49385
                       0.00000  0.00000  0.41498
                       0.50000  0.00000  0.40298
                       0.00000  0.50000  0.32769
                       0.50000  0.50000  0.32150
                       0.25000  0.50000  0.24167
                       0.75000  0.50000  0.24167
                       0.25000  0.20000  0.18269       H
                       0.25000  0.80000  0.18269       H
                       0.75000  0.20000  0.18269       H
                       0.75000  0.80000  0.18269       H
            }
       }
}

postprocessing{
    charge{
        sw_charge_rspace    = ON
        filetype            = cube  !{cube|density_only}
        title  = "Si(001) p(2x1) surface terminated by H atoms"
    }
}
```

In the **atoms** block, the default element is set to Si, so the atoms not specified as H by the element attribute are treated as Si atoms. Since **mobile = 0** is set by default, the positions of all atoms are fixed.

```
 % grep TOTAL output000
```

By the above command, the convergence progress of the total energy can be checked as follows.

```
TOTAL ENERGY FOR     1 -TH ITER=    -41.206501960258  edel =  -0.412065D+02 : SOLVER = MATDIAGON
 TOTAL ENERGY FOR     2 -TH ITER=    -42.928541839902  edel =  -0.172204D+01 : SOLVER = DAVIDSON
 TOTAL ENERGY FOR     3 -TH ITER=    -42.956734520103  edel =  -0.281927D-01 : SOLVER = DAVIDSON
 TOTAL ENERGY FOR     4 -TH ITER=    -42.960659333525  edel =  -0.392481D-02 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR     5 -TH ITER=    -42.961623666220  edel =  -0.964333D-03 : SOLVER = SUBMAT + RMM3
```

```
 TOTAL ENERGY FOR     6 -TH ITER=    -42.962559338199  edel =  -0.935672D-03 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR     7 -TH ITER=    -42.964136746929  edel =  -0.157741D-02 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR     8 -TH ITER=    -42.964791285123  edel =  -0.654538D-03 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR     9 -TH ITER=    -42.964953052183  edel =  -0.161767D-03 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    10 -TH ITER=    -42.965045860995  edel =  -0.928088D-04 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    11 -TH ITER=    -42.965076083146  edel =  -0.302222D-04 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    12 -TH ITER=    -42.965088896548  edel =  -0.128134D-04 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    13 -TH ITER=    -42.965091550789  edel =  -0.265424D-05 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    14 -TH ITER=    -42.965092402734  edel =  -0.851945D-06 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    15 -TH ITER=    -42.965092972980  edel =  -0.570245D-06 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    16 -TH ITER=    -42.965093291397  edel =  -0.318417D-06 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    17 -TH ITER=    -42.965093454357  edel =  -0.162961D-06 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    18 -TH ITER=    -42.965093580068  edel =  -0.125710D-06 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    19 -TH ITER=    -42.965093601039  edel =  -0.209711D-07 : SOLVER = SUBMAT + RMM3
 TOTAL ENERGY FOR    20 -TH ITER=    -42.965093604435  edel =  -0.339656D-08 : SOLVER = SUBMAT + RMM3
```

Although the above example is only for the energy calculation, you can perform structure relaxations of a surface, too. To execute a relaxation calculation, you need fix the bottom artificial hydrogen atoms and the atoms connected to them. In this example, the value of the mobile variable is set to 1, and the mobile attribute of the fixed atom is set to 0.

```
        atoms{
                #default weight = 1, element = Si, mobile = 1
                #tag   rx       ry       rz         element  mobile
                       0.26177  0.50000  0.65651       H
                       0.73823  0.50000  0.65643       H
                       0.34138  0.50000  0.56971
                       0.65858  0.50000  0.56966
                       0.26229  0.00000  0.49388
                       0.73763  0.00000  0.49385
                       0.00000  0.00000  0.41498
                       0.50000  0.00000  0.40298
                       0.00000  0.50000  0.32769
                       0.50000  0.50000  0.32150
                       0.25000  0.50000  0.24167       *        0
                       0.75000  0.50000  0.24167       *        0
                       0.25000  0.20000  0.18269       H        0
                       0.25000  0.80000  0.18269       H        0
                       0.75000  0.20000  0.18269       H        0
                       0.75000  0.80000  0.18269       H        0
         }
```

Note that the stable structure of the buckled dimer for a Si(001) surface is c(4 × 2), not p(2 × 1). To reproduce this structure, you need to add one more Si dimer so that the number of Si dimers on the surface is even.

4.5.2 Surface calculation using inversion symmetry

Some surfaces have inversion symmetry. By taking advantage of this, you can reduce the computational cost by half. The following input is for a Pt(111) surface. In this example, the **structure** block is specified as below.

```
structure{
    element_list{
        #tag element atomicnumber mass zeta deviation
         Pt 78 355606.909 0.0 1.83
    }
    atom_list{
        coordinate_system = cartesian
        atoms{
            #units angstrom
            #tag element rx ry rz mobile weight
Pt 0.00 0.00 0.00
```

```
Pt 1.4142135624 2.4494897428 0.00
Pt 2.8284271248 0.00 0.00
Pt 4.2426406871 2.4494897428 0.00
Pt 5.6568542497 3.2659863239 2.30940111
Pt 4.2426406874 0.8164965811 2.30940111
Pt 2.828427125 3.2659863239 2.30940111
Pt 1.4142135626 0.8164965811 2.30940111
Pt 2.8284271245 1.6329931617 4.618802187
Pt 4.2426406868 4.0824829045 4.618802187
Pt 5.6568542492 1.6329931617 4.618802187
Pt 7.0710678116 4.0824829045 4.618802187
Pt 5.6568543525 0.0000002214 6.928203264
Pt 1.4142137683 2.4494897428 6.928203264
Pt 2.8284271248 0.00 6.928203264
Pt 4.2426406871 2.4494897428 6.928203264
Pt 5.6568542497 3.2659863239 9.237604341
Pt 4.2426406874 0.8164965811 9.237604341
Pt 2.828427125 3.2659863239 9.237604341
Pt 1.4142135626 0.8164965811 9.237604341
Pt 2.8284271245 1.6329931617 -2.30940111
Pt 4.2426406868 4.0824829045 -2.30940111
Pt 5.6568542492 1.6329931617 -2.30940111
Pt 7.0710678116 4.0824829045 -2.30940111
Pt 5.6568542497 3.2659863239 -4.618802187
Pt 4.2426406874 0.8164965811 -4.618802187
Pt 2.828427125 3.2659863239 -4.618802187
Pt 1.4142135626 0.8164965811 -4.618802187
Pt 2.8284270217 4.8989792642 -6.928203264
Pt 7.0710676059 2.4494897428 -6.928203264
Pt 2.8284271248 0.00 -6.928203264
Pt 4.2426406871 2.4494897428 -6.928203264
Pt 2.8284271245 1.6329931617 -9.237604341
Pt 4.2426406868 4.0824829045 -9.237604341
Pt 5.6568542492 1.6329931617 -9.237604341
Pt 7.0710678116 4.0824829045 -9.237604341
        }
    }
    unit_cell{
        #units angstrom
        a_vector = 5.6568542495 0.00 0.00
        b_vector = 2.8284271247 4.8989794856 0.00
        c_vector = 0.00 0.00 30.00
    }
    symmetry{
        method = automatic
        tspace{
            lattice_system = primitive
        }
        sw_inversion = on
    }
}
```

This structure has inversion symmetry whose center is on the origin. To utilize this symmetry, set the variable **sw_inversion** to "on." The structure is shown in Figure 4.8.

In this example, the surface has inversion symmetry along the thickness direction. In such cases, you can reduce the computational cost to half by setting **sw_inversion** to "on." Calculations in which molecules or atoms are adsorbed on the surface can also be executed by arranging the adsorbent on both sides to preserve

the inversion symmetry.

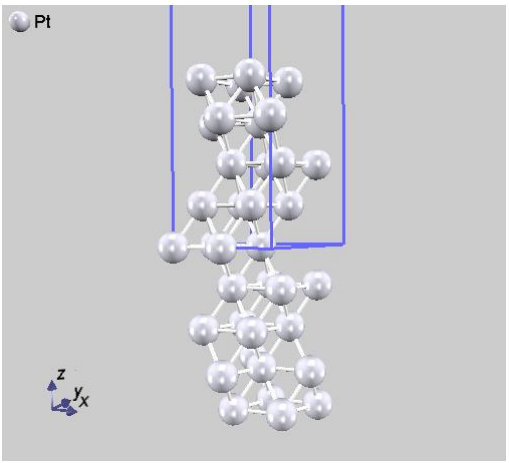


Figure 4.8 Atomic positions of Pt(111) surface.
This structure has an inversion symmetry whose center is at the origin.

4.5.3 Example: generation energy of metallic surfaces

Generation energy of a surface at 0 K can be estimated by the equation

$$\gamma = (E_s - E_b)/2A$$

Here γ is the surface generation energy, E_s is the total energy of the surface, E_b is the total energy of the buck structure, and A is the surface area. The above equation is divided by 2A because two surfaces appear in the calculation. Note that the total energy of the buckled structure is scaled to fit the number of atoms in the surface model.

Here we introduce an example for calculating the generation energy of a Pt surface.

Pt(111) surface	nine-layered (111) surface, total 36 atoms lattice parameters are $a = b = 5.657\text{Å}, c = 30\text{Å}, \alpha = \beta = 90°, \gamma = 120°$ See Fig. 4.8.
Pt(110) MR surface	Fifteen-layered missing-row (MR) (110) surface, total 28 atoms MR surface means that the surfaces in which the atoms that compose a "row" of surface are missing every second row. lattice parameters are $a = 4\text{Å}, b = 2.828427125\text{Å}, c = 30\text{Å}, \alpha = \beta = \gamma = 90°$ See Fig. 4.10 (This figure is displayed in super cell.)
Pt(110) surface	Fifteen-layered (110) surface, total 15 atoms lattice parameters are $a = 8\text{Å}, b = 2.8284271248\text{Å}, c = 30\text{Å}, \alpha = \beta = \gamma = 90°$ See Fig. 4.9 (This figure is displayed in super cell.)

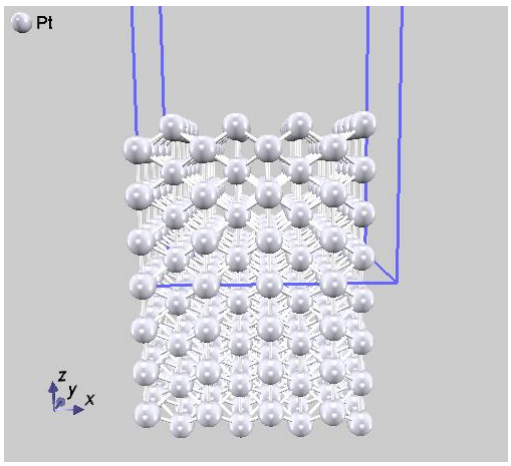


Figure 4.9 Pt(110) ideal surface (viewed in super cell)

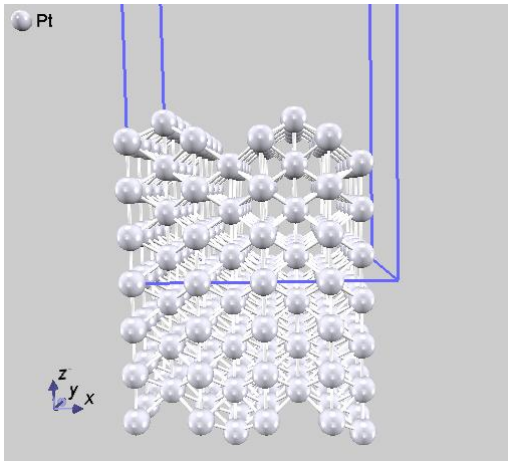


Figure 4.10 Pt(110) missing-row surface model (viewed in super cell)

Generally speaking, the (111) surface is the most stable Pt surface, and surface reconstruction occurs in the (110) surface to generate a mission-row surface. Here we confirm that the calculated surface generation energy can explain these results.

- all models employ inversion symmetry
- cutoff energy is 25 Rydberg
- k-point samplings are 6×6×1, 6×8×1, and 3×8×1 for (111), (110), and (110) MR surface, respectively
- geometry optimization is performed by the BGFS method; the convergence criteria is 2×10^{-4} hartree/bohr
- four layers from the surface are optimized

Table 4.3 lists the surface generation energy obtained from the above conditions. These results indicate that the (111) surface has the lowest energy, followed by (110) MR and by (110) surface.

Table 4.3 Generation energies for the platinum surface
The (111) surface has the lowest energy, followed by (110) MR and (110) surfaces.

	(111)	(110) MR	(110)
generation energy (eV/Å^2)	0.089	0.099	0.108

4.6 Calculation of atoms and molecules

Isolated atoms and molecules can also be calculated by creating a vacuum layer. In such cases, the vacuum layer needs to be created in all directions to negate the effects of periodic boundary conditions. Normally, k-point is sampled only at the Γ point.

4.6.1 Input parameters

To calculate isolated atoms or molecules, a large unit cell is defined.

```
        unit_cell{
              a_vector =  15.0              0.0              0.0
              b_vector =   0.0             15.0              0.0
              c_vector =   0.0              0.0             15.0
        }
```

The following example is for the calculation of a water molecule. The unit cell is large compared to the molecule.

```
Control{
        condition = initial
        cpumax = 1 day ! maximum cpu time
        max_iteration = 6000
}

accuracy{
        cutoff_wf =   25.00  rydberg
        cutoff_cd =  225.00  rydberg
        num_bands = 8
        xctype = ggapbe
        initial_wavefunctions = matrix_diagon
        matrix_diagon {
          cutoff_wf = 5.0 rydberg
        }
        ksampling{
          method = gamma
        }
        scf_convergence{
          delta_total_energy = 1.e-10
          succession = 3
          num_max_iteration = 300
        }
        force_convergence{
          delta_force = 1.e-4
        }
        initial_charge_density = Gauss
}

structure{
        unit_cell_type = primitive
        unit_cell{
              a_vector =  15.0              0.0              0.0
              b_vector =   0.0             15.0              0.0
              c_vector =   0.0              0.0             15.0
        }
        symmetry{
             tspace{
                lattice_system = primitive
```

```
                generators{
                   #tag rotation tx ty tz
                        C2z      0  0  0
                        IC2x     0  0  0
                }
             }
        }

        atom_list{
             coordinate_system = cartesian
             atoms{
                !#default mobile=on
                !#tag  rx             ry       rz          element
                      -1.45           0.000    1.123       H
                       1.45           0.000    1.123       H
                       0.0            0.0      0.0         O
             }
        }
        element_list{  #units atomic_mass
                       #tag element  atomicnumber zeta  dev
                            H             1       1.00  0.5
                            O             8       0.17  1.0    }
}

wf_solver{
        solvers {
        !#tag sol    till_n dts dte itr  var    prec cmix submat
              msd      5    0.1 0.1   1    tanh on   1    on
              lm+msd  10    0.1 0.4  50    tanh on   1    on
              rmm2p   -1    0.4 0.4   1    tanh on   2    on
        }
        rmm {
          edelta_change_to_rmm = 1.d-6
        }
        lineminimization {
          dt_lower_critical = 0.1
          dt_upper_critical = 3.0
        }
}

charge_mixing{
        mixing_methods {
        !#tag id method   rmxs rmxe itr var    prec istr nbxmix update
              1  broyden2 0.3  0.3  1   linear on   5    10     RENEW
              2  simple   0.2  0.5  100 linear on   *    *      *
        }
}
```

4.7 Output of charge density

Although the charge density is treated in reciprocal space during an SCF calculation, the converged charge density can be converted to real space and outputted to a file. PHASE-Viewer and other viewers can then be used to visualize the charge density. To output the charge density to real space, define the **postprocessing** block at the head of the input file and specify the charge block in it.

```
postprocessing{
    charge{
        sw_charge_rspace = on
        filetype = cube
    }
}
```

The **charge** block contains the following variables.

sw_charge_rspace	Switch that specifies whether to generate the charge density in real space. Options are **on** or **off**.
filetype	Specifies the file format for the charge density data. Options are **density_only**: only the charge density is written to the file. (default) **cube**: charge density is stored in the Gaussian cube format. Using the **cube** option is recommended.
title	Title of the Gaussian Cube file. Double quotes " " are used to include spaces in the title.

If **filetype=cube**, it is recommended to change the file name of the charge density file. The filename can be specified in the file "file_names.data" as shown below.

```
&fnames
...
F_CHR = './nfchr.cube'
/
```

The default name of the file is "nfchr.data." If spin polarization is considered and "nfchr.cube" is set to the filename, two cube files "nfchr.up.cube" and "nfchr.down.cube," which are the respective densities of up and down spins, will be generated.

As an example, Figure 4.11 shows the charge densities of minority and majority spins, visualized by PHASE-Viewer. In addition, PHASE can extract and output the charge density within a specific energy range. This function is described later in the section on advanced functions.

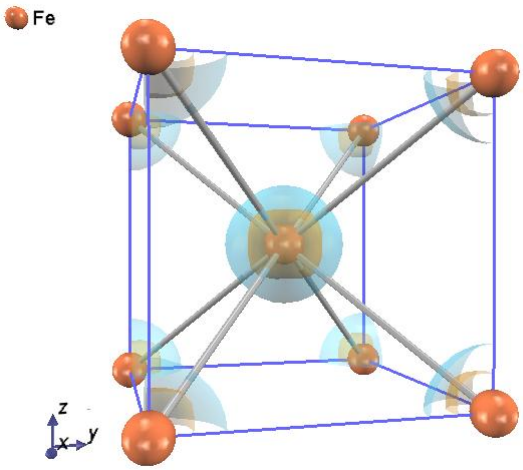


Figure 4.11 Charge density distribution of Fe. Blue and orange surfaces denote the respective isosurfaces for the charge densities of minority and majority spins generated by spontaneous magnetization

4.8 Density of states

The DOS can be calculated after SCF iterations converge. Calculation of DOS is specified in the **dos** block in the **postprocessing** block as follows.

```
postprocessing{
    dos{
        sw_dos = on
        method = gaussian
        deltaE_dos = 1e-4 hartree
    }
}
```

The following variables are available in the **dos** block.

sw_dos	Switch that specifies whether to calculate the DOS. Options are **on** and **off**.
method	Method for calculating the DOS. Options are **gaussian**: simple Gaussian broadening **tetrahedral**: accurate calculation based on the tetrahedral method Note that the tetrahedral method is available only under limited conditions (See below).
deltaE_dos	Specifies the broadening used in the DOS calculation, in units of energy. Defaults to 1e−4 hartee.

The tetrahedral method is available when

- mesh method is employed for k-sampling

```
accuracy{
    ksampling{
        method = mesh
    }
}
```

- tetrahedral method is used for smearing

```
accuracy{
    smearing{
        method = tetrahedral
    }
}
```

If the above conditions are not satisfied, the DOS is calculated by the Gaussian method.

Figure 4.12 and Figure 4.13 show the DOS for body-centered cubic iron that are calculated by the Gaussian method and the tetraheral method, respectively. Both calculations employed a 10 × 10 × 10 k-point mesh. These figures indicate that the tetrahedral method can calculate the DOS more sharply and accurately than the Gaussian method.

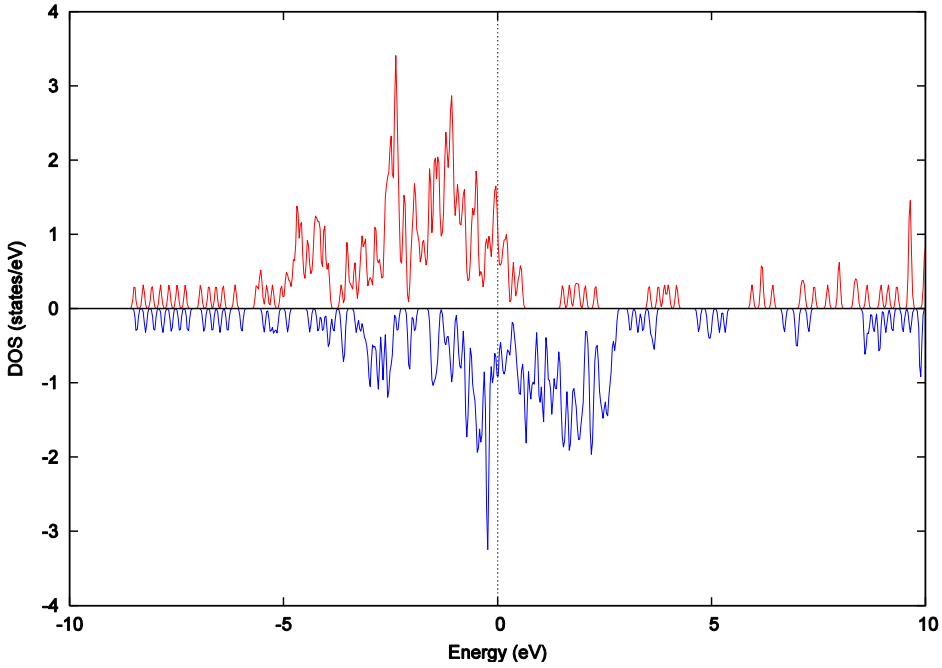


Figure 4.12 Density of states of bcc Fe calculated by the Gaussian method

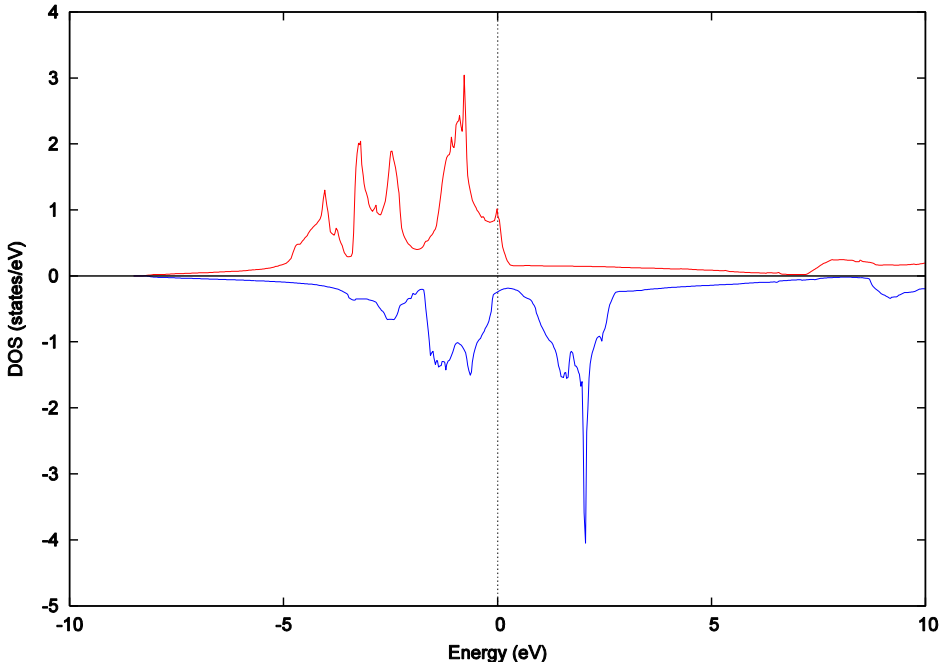


Figure 4.13 Density of states of bcc Fe calculated by the tetrahedral method

PHASE can also calculate a "partial DOS;" that is a DOS for specific atoms, layers, etc. This function is described later in the section on advanced functions.

4.9 Calculation of band structure

4.9.1 Generating k-point data

To obtain band structures, we need k-point data to calculate band dispersion. The k-point data are generated by the script band_kpoint.pl. First, you need to prepare an input file for the band_kpoint.pl. The format of the input file is shown below.

```
dkv
b1x b2x b3x
b1y b2y b3y
b1z b2z b3z
n1 n2 n3 nd # Symbol
...
```

The dkv of the first line is the interval between k-points. In the second line, b1x, b1y, and b1z represent the x, y, and z components of the reciprocal lattice vector b_1, respectively. The third and fourth lines are the x, y, and z components of reciprocal lattice vectors b_2,b_3. In the fifth line, special k-points and their symbols are specified. The specifications of these symbols are not required. However, if the symbols are specified, they are used to output a band structure figure.

The vectors of these k-points k are specified by the integers n_1, n_2, n_3, n_d as

$$k = \frac{n_1}{n_d} b_1 + \frac{n_2}{n_d} b_2 + \frac{n_3}{n_d} b_3$$

The symbols are written after the #. An example for a face-centered cubic lattice is shown below.

```
0.02                        <---- interval of k-points
-1.0  1.0  1.0
1.0 -1.0  1.0               <---- reciprocal lattice vector
1.0  1.0 -1.0
0 1 1 2 # X                 <---- n1 n2 n3 nd # Symbol
0 0 0 1 # {/Symbol G}
1 1 1 2 # L
5 2 5 8 # U
1 0 1 2 # X
```

After preparing the input file, a file kpoint.data can be generated using the script band_kpoint.pl as shown below.

```
% band_kpoint.pl bandkpt.in
```

The following is an example of kpoint.data.

```
141 141          (a)
0 50 50 100 1    (b)
0 49 49 100 1
0 48 48 100 1
0 47 47 100 1
0 46 46 100 1
0 45 45 100 1
0 44 44 100 1
0 43 43 100 1
   ......
   ......
   ......
```

The content of this file is as follows:

(a) The first line gives the number of k-points. This example used 141 k-points.

(b) The remaining lines contain five integers: n_1, n_2, n_3, n_d, w. These are used in

$$\vec{k} = w \times \left(\frac{n_1}{n_d}\overrightarrow{b_1} + \frac{n_1}{n_d}\overrightarrow{b_2} + \frac{n_3}{n_d}\overrightarrow{b_3}\right)$$

where $\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}$ are reciprocal lattice vectors.

4.9.2 Calculation with fixed charge

The program **ekcal** is used to calculate the DOS or band structure with fixed charges as obtained from a previous SCF calculation. Although you can execute this calculation in the same directory for the SCF calculation, it is recommended that you create a new directory and execute the calculation in it to avoid overwriting other output files, such as those containing wavefunction data.

4.9.2.1 Input parameters

11. file_names.data

In case of ekcal, you need to specify the charge density file created by the SCF calculation. The name of this file is specified by the F_CHGT keyword in **filenames.data** of the previous SCF calculation; its default name is **nfchgt.data**. For example, if the calculation is executed in the directory created in the SCF directory, the charge density file is specified in the **file_names.data** as follows. The k-points data file **kpoint.data**, which is used to plot the band structure, can be identified by the F_KPOINT keyword.

```
&fnames
...
F_CHGT = '../nfchgt.data'
F_KPOINT = 'kpoint.data'
...
/
```

If the PAW method is employed, in addition to the F_CHGT keyword, the F_CNTN_BIN_PAW keyword must also be specified to the file created by the SCF calculation. If the DFT+U method is employed, the occupied matrix data file must be specified by the F_OCCMAT keyword. See the example below.

```
&fnames
...
F_CHGT = '../nfchgt.data'
F_OCCMAT = '../occmat.data'                          <--- necessary for DFT+U
F_CNTN_BIN_PAW = '../continue_bin_paw.data'          <--- necessary for PAW
...
/
```

12. Input parameter file

Here we explain how to create the input file for calculations with fixed charge. Basically, it is easier to make the input file from the input file used in the previous SCF calculation. However, note the following.

- Atomic coordinates

If geometry optimization was performed in the previous calculation, the ekcal calculation must be executed with the optimized structure. In this case, use the final atomic coordinates printed in the output file specified by the F_DYNM keyword.

- Calculation condition

Set the **condition** variable to fixed_charge.

```
Control{
    ...
    condition = fixed_charge
    ...
}
...
```

The fixed_charge calculation can also be restarted. To restart the calculation, set the **condition** variable to "fixed_charge_continuation."

- k-point sampling

Set the **method** variable in the **ksampling** block to "file" to read the generated **kpoint.data.**

```
accuracy{
    ...
    ksampling{
        method = file
    }
    ...
}
```

- ek_convergence block

The **ek_convergence** block in the **accuracy** block specifies convergence criteria. Set the block as follows.

```
accuracy{
    ...
    ek_convergence{
        num_max_iteration = 500
        delta_eigenvalue = 1.e-5
        succession = 2
    }
    ...
}
```

- The **ek_convergence** block contains the following variables.

num_max_iteration	Specifies the maximum number of iterations
delta_eigenvalue	Specifies the convergence criterion for the energy difference. The default value, 1.e−15 hartree, is very small. Use about 1.e−4 rydberg for insulator/semiconductor materials and about 1.e−6 rydberg for metals.
succession	Iterations are terminated when the energy difference is smaller than the criterion **delta_eigenvalue** *n* times in succession. The variable **succession** specifies the number *n*. Default is 3.

- Solver

The default solver used in the ekcal is the steepest descent method. Since this simple method requires a large number of iterations, use one of the other solvers, such as **lm+msd**, **davidson**, **rmm3**.

4.9.3 Plotting band structure

As an output of the calculation, eigenenergies of bands for all k-points are printed to the file nfenergy.data.

```
 num_kpoints =    117                                                    (a)
 num_bands   =      8                                                    (b)
 nspin       =      1                                                    (c)
 Valence band max   =   0.233846                                         (d)

 nk_converged =      117                                                 (e)
 ik =    1 (  0.500000  0.500000  0.000000 )
 ik =    2 (  0.487805  0.487805  0.000000 )
 ik =    3 (  0.475610  0.475610  0.000000 )
 ik =    4 (  0.463415  0.463415  0.000000 )
 ik =    5 (  0.451220  0.451220  0.000000 )
 ik =    6 (  0.439024  0.439024  0.000000 )
...
...
...

=== energy_eigen_values ===
 ik =    1 (  0.000000  0.500000  0.500000 )                            (f)
     -0.0484324576     -0.0484324576      0.1258094928      0.1258094928  (g)
      0.2619554301      0.2619554301      0.6015285208      0.6015285208
=== energy_eigen_values ===
 ik =    2 (  0.000000  0.490000  0.490000 )
     -0.0540717201     -0.0427149632      0.1258687739      0.1258687739
      0.2607026807      0.2633829927      0.6006243932      0.6006243932
                           ......
                           ......
                           ......
```

The above items are

(a) Number of k-points. This example has 117 k-points.

(b) Number of bands. This example has eight bands.

(c) Spin degree of freedom, 1 or 2. In this example, the value is 1, which means that spin polarization was not considered in the calculation.

(d) Fermi energy. For semiconductor/insulator materials, the energy of the valence-band edge is printed. The unit is hartree.

(e) Calculated k-points.

(f) Eigenvalues are printed here. This first line identifies the k-point to which the eigenvalues apply. In this example, the first k-point corresponds to the (0,0.5,0.5) reciprocal lattice vector.

(g) Eigenvalues for all bands are printed. The unit is hartree.

If spin polarization is considered, the output of eigenenergies is almost same, but "UP" or "DOWN" is added next to item (f). Eigenvalues corresponding to the major and minor spin are printed.

```
                           ......
                           ......
                           ......
=== energy_eigen_values ===
 ik =    1 (  0.000000  0.000000  0.000000)    UP
     -0.1998699758      0.0267639589      0.0267639589      0.0267639589
      0.0725171077      0.0725171077      1.0289118953      1.0289118953
      1.0289118953      1.1650173104      1.1650173104      1.1650173104
```

```
      1.2129026022      1.2129026022      1.2994754011      1.2994754011
      1.2994754011      1.6365336765      2.2629596795      2.2629596795
=== energy_eigen_values ===
 ik =    2 (  0.000000  0.000000  0.000000)  DOWN
     -0.1960420390      0.1062941746      0.1062941746      0.1062941746
      0.1799862148      0.1799862148      1.0183970612      1.0183970612
      1.0183970612      1.2174266166      1.2174266166      1.2192701193
      1.2192701193      1.2192701193      1.3289165100      1.3289165100
      1.3289165100      1.6910264603      2.2876818717      2.2876818717
                           ......
                           ......
                           ......
```

To plot the band structure from this data, a useful Perl script band.pl, which is contained in PHASE, is available. The script band.pl is executed as shown below.

```
% band.pl nfenergy.data bandkpt.in -erange=-10,10 -color -with_fermi
```

As an example, band structure of body-centered cubic iron is shown in Figure 4.14.

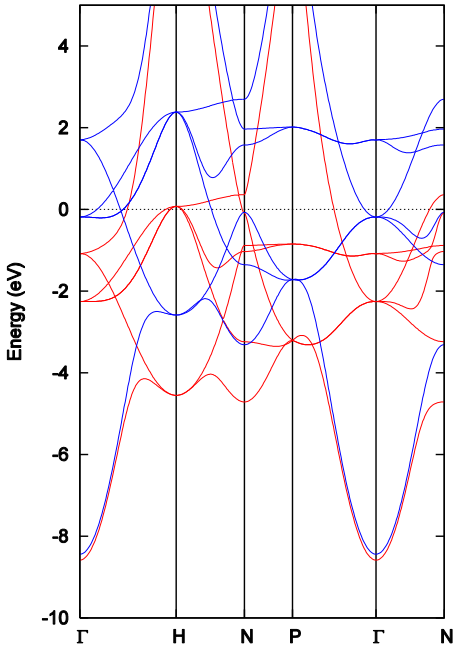


Figure 4.14 Band structure of body-centered cubic iron

4.10 Lattice constant

4.10.1 Calculation method

The equilibrium lattice constant can be obtained from total energies that are calculated for several different lattice constants. In addition, if the lattice is cubic, the bulk modulus can also be determined by fitting the Murnaghan equation of state,

$$E_{\mathrm{tot}}(V) = \frac{BV}{B'\left(B'-1\right)}\left[B'\left(1-\frac{V_0}{V}\right)+\left(\frac{V_0}{V}\right)^{B'}-1\right]+E_{\mathrm{tot}}(V_0)$$

Here $E_{\mathrm{tot}}(V)$ is the total energy with the lattice constant whose unit cell volume is V, B is the bulk modulus, B' is the bulk modulus pressure derivative, and V_0 is the unit cell volume for the equilibrium lattice constant. The four variables $B, B', V_0, E_{\mathrm{tot}}(V_0)$ are fitting parameters.

4.10.2 Example: Si crystal

Here we describe an example for calculating the equilibrium lattice constant of a Si crystal. This example is stored in the directory sample/Si_lat. In this directory, there are several subdirectories named volxxx. Each subdirectory contains input data for a unit cell having volume xxx. For example, the calculation model in the directory vol1200 is specified as shown below.

```
structure{
  element_list{
    #tag    element    atomicnumber
            Si    14
  }
  atom_list{
    atoms{
      #units angstrom
      #tag    element    rx    ry    rz
              Si    0.125  0.125  0.125
              Si   -0.125 -0.125 -0.125
    }
    coordinate_system = internal
  }
  unit_cell{
    a_vector = 10.62658569182611066038 0 0
    b_vector = 0 10.62658569182611066038 0
    c_vector = 0 0 10.62658569182611066038
  }
  symmetry{
    method = automatic
    tspace{
        lattice_system = facecentered
    }
    sw_inversion = on
  }
  unit_cell_type = bravais
}
```

Atomic coordinates are given in fractional coordinates (atomic positions are referred to the lattice vectors). Fractional coordinates are more appropriate than Cartesian coordinates because Cartesian coordinates need to be modified every time the lattice constant is changed.

In this example, the **unit_cell_type** is set to "bravais," and the **lattice_system** is set to "facecentered." By using this variable, you can input the lattice using a bravais lattice, which is easy to specify. However, the actual calculations are performed using a basic lattice, which is easy to calculate. Note that if the volume of Bravais lattice is employed, you need to scale the results. In this example, the bulk modulus is quadruplicated because the volume of the Bravais lattice is four times larger than that of the basic lattice.

Figure 4.15 shows the energy–volume curve fitted to the Murnaghan equation of state, and Table 4.4 lists the equilibrium lattice constant and bulk modulus obtained from the fit. Cohesive energy is also separately calculated and is listed in Table 4.4. The cohesive energy is the difference between the average energy of the atoms of a crystal and that of the free atoms. It can be obtained by $E_{\mathrm{isolated}} - E_{\mathrm{solid}}/N_{\mathrm{atom}}$, where E_{isolated} is the total energy of a free atom, E_{solid} is the total energy of a crystal at equilibrium lattice constant, and N_{atom} is the number of atoms in the crystal.

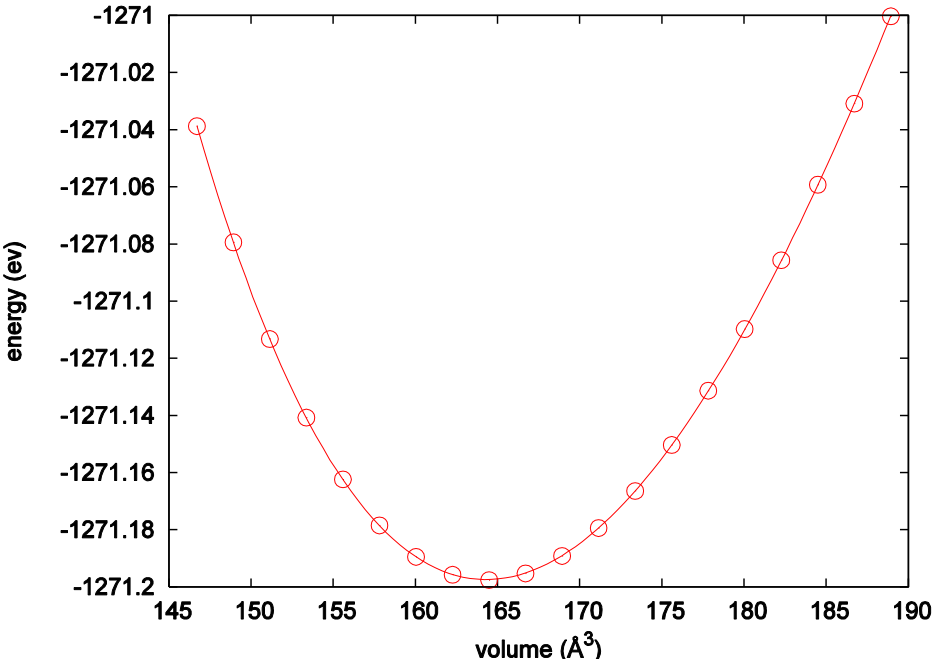


Figure 4.15 Energy-volume curve for a Si crystal. The white circles represent calculated values, and the solid line represents the result from the fit.

Table 4.4 Resulting equilibrium lattice constant and bulk modulus

	PHASE	Experimental data
a (Å)	5.48	5.43
B (GPa)	87.5	98.8
Ecoh (eV/atom)	4.60	4.63

5. Advanced functions

5.1 Analysis functions

5.1.1 Stress tensor

5.1.1.1 Overview

PHASE has a function to calculate the stress tensor. By calculating it, the bulk modulus can be estimated.

5.1.1.2 Input parameters

To calculate the stress tensor, you need to define **sw_stress=1** in the **stress** block in the **structure_evolution** block. The following example is an input parameter file for Si (cubic). This file is in **sample/stress/**.

```
Control{
  cpumax = 24 hour
}

accuracy{
  cutoff_wf =  20.25  rydberg
  cutoff_cd =  81.00  rydberg
  num_bands = 20
  xctype = ggapbe
  ksampling{
    method = mesh
    mesh{ nx = 8, ny = 8, nz = 8 }
  }
  smearing{
    method = tetrahedral
  }
  scf_convergence{
    delta_total_energy = 1.0e-10  hartree
    succession = 3
  }
  force_convergence{
    delta_force = 1.0e-4
  }
  initial_wavefunctions = matrix_diagon
  matrix_diagon{
    cutoff_wf = 5.00  rydberg
  }
  initial_charge_density = Gauss
}

structure{
  unit_cell_type = primitive
  unit_cell{
    #units angstrom  ! Unit of LENGTH changes to Angstrom.
    a_vector =   0.0000000000   2.7296850000   2.7296850000
    b_vector =   2.7296850000   0.0000000000   2.7296850000
    c_vector =   2.7296850000   2.7296850000   0.0000000000
  }

  symmetry{
    crystal_structure = diamond
  }
```

```
  atom_list{
    coordinate_system = internal
    atoms{
      #tag    rx      ry      rz   element  mobile  weight
           0.125   0.125   0.125        Si     yes       1
          -0.125  -0.125  -0.125        Si     yes       1
    }
  }
  element_list{ #tag   element   atomicnumber   dev
                           Si            14    1.2
  }
}

structure_evolution{
  stress{
    sw_stress=1
  }
}
```

Execute PHASE as usual.

```
% mpirun PATH_TO_PHASE
```

Check results after the calculation is completed. The calculated stress tensor can be extracted from the output file by the following command.

```
% grep -A3 'STRESS TENSOR$' OUTPUT_FILE
```

```
  STRESS TENSOR
        0.0000003475        0.0000000000        0.0000000000
        0.0000000000        0.0000003475        0.0000000000
        0.0000000000        0.0000000000        0.0000003475
```

The stress tensor is printed in the matrix form below:

$$\begin{pmatrix} X_x & X_y & X_z \\ Y_x & Y_y & Y_z \\ Z_x & Z_y & Z_z \end{pmatrix}$$

The unit is [Hartree/Bohr^3]. Because slightly smaller values were given to the lattice constants in the above example, positive values were obtained for the diagonal elements X_x, Y_y, Z_z. These values become 0 if the lattice constants are the equilibrium ones. Here the following Hooke's law holds:

$$\left.\begin{aligned} X_x &= c_{11}e_{xx} + c_{12}e_{yy} + c_{12}e_{zz} \\ Y_y &= c_{12}e_{xx} + c_{11}e_{yy} + c_{12}e_{zz} \\ Z_z &= c_{12}e_{xx} + c_{12}e_{yy} + c_{11}e_{zz} \\ X_y(=Y_x) &= c_{44}e_{xy} \\ Y_z(=Z_y) &= c_{44}e_{yz} \\ Z_x(=X_z) &= c_{44}e_{zx} \end{aligned}\right\}$$

where e represents a lattice deformation from the equilibrium constants, and c represents the stiffness constant.

5.1.1.3 Elastic constant

The elastic constant can be obtained from the calculated stress tensor. Here we calculate the elastic constant of a Si (cubic) crystal from its stress tensor. First, the equilibrium lattice constant, in which the stress tensor is 0, needs to be calculated. An accurate value for the lattice constant is necessary to calculate an accurate

stress tensor. In this example, the following lattice vectors, in which the stress tensor is almost 0, are employed.

```
    a_vector =   0.0000000000   2.7297895000   2.7297895000
    b_vector =   2.7297895000   0.0000000000   2.7297895000
    c_vector =   2.7297895000   2.7297895000   0.0000000000
```

The following stress tensor is obtained after the calculation is completed.

```
% grep -A3 'STRESS TENSOR$' OUTPUT_FILE
```

```
  STRESS TENSOR
        0.0000000000        0.0000000000        0.0000000000
        0.0000000000        0.0000000000        0.0000000000
        0.0000000000        0.0000000000        0.0000000000
```

Make sure that all elements are zero or sufficiently small. Next, you need to deform the unit cell; e.g., make the unit cell 0.01 angstrom larger in the x-direction. Modify the lattice vector as follows. Note that the **symmetry** block in the input file must be removed or commented out.

```
    a_vector =   0.0000000000   2.7296850000   2.7296850000
    b_vector =   2.7296850000   0.0000000000   2.7296850000
    c_vector =   2.7296850000   2.7296850000   0.0000000000
```

The following stress tensor is obtained for the modified lattice.

```
% grep -A3 'STRESS TENSOR$' OUTPUT_FILE
```

```
  STRESS TENSOR
       -0.0000093954        0.0000000063        0.0000000016
        0.0000000063       -0.0000033142        0.0000000000
        0.0000000016        0.0000000000       -0.0000033163
```

In this example, the elastic constant is obtained from the diagonal elements of the stress tensor. Y_y and Z_z are supposed to be equivalent because of symmetry; thus, their mean value −0.00000331525 is used for both Y_y and Z_z. Since a twist or shearing strain is not given in this example, the off-diagonal elements become zero in theory. Some off-diagonal elements are not exactly zero because of numerical error.

By using a deformation from the equilibrium constants (0.01 angstrom in the x-direction) and the calculated stress tensor, the stiffness constants c_{11}, c_{12} can be obtained as follows:

$$\left.\begin{aligned} X_x &= c_{11}e_{xx} + c_{12}e_{yy} + c_{12}e_{zz} \\ Y_y &= c_{12}e_{xx} + c_{11}e_{yy} + c_{12}e_{zz} \\ Z_z &= c_{12}e_{xx} + c_{12}e_{yy} + c_{11}e_{zz} \\ X_y(=Y_x) &= c_{44}e_{xy} \\ Y_z(=Z_y) &= c_{44}e_{yz} \\ Z_x(=X_z) &= c_{44}e_{zx} \end{aligned}\right\}$$

In this example, the stiffness constants c_{11}, c_{12} are calculated as below (units are $[10^{12}\mathrm{dyn/cm^2}]$):

$$\left.\begin{aligned} c_{11} &= 1.5091525 \\ c_{12} &= 0.5325178 \end{aligned}\right\}$$

Further, elastic constants, Young's modulus($\equiv Y$), Poisson's ratio($\equiv P$), and the bulk modulus($\equiv V$) can be obtained by from the following equations.

$$\left.\begin{aligned} Y &= \frac{c_{11}^2 + c_{11}c_{12} - 2c_{12}^2}{c_{11} + c_{12}} \\ P &= \left|\frac{c_{12}}{c_{11} + c_{12}}\right| \\ V &= \frac{c_{11} + 2c_{12}}{3} \end{aligned}\right\}$$

The modulus of rigidity is given by $Y/(2 + 2P)$. By substituting the stiffness constants c_{11}, c_{12} into the above equations, the elastic constants of Si are obtained:

$$\left.\begin{aligned} Y &\approx 1.231[10^{12}\text{dyn/cm2}] = 123.1[\text{GPa}] \\ P &\approx 0.261(16) \\ V &\approx 0.858[10^{12}\text{dyn/cm2}] = 85.8[\text{GPa}] \end{aligned}\right\}$$

To accurately calculate elastic constants, **cutoff_wf** and **cutoff_cd** must be sufficiently large to get well-converged wave functions. However, this calculation is time consuming.

5.1.2 Local density of states and energy-dependent charge density

5.1.2.1 General features

To analyze electronic states, the local density of states (DOS) and energy-dependent charge density are very useful. With the atom-divided local DOS, bonding states become clear. For a laminated structure or an interface between two materials, the layer-divided local DOS is a powerful tool that enables users to identify layer-dependent electronic states or a change in electronic states around an interface. An energy-dependent charge density provides the charge density over a limited range of energies. This enables users to identify the atoms that contribute to the states in that energy range.

In the following, both functions are described using the interface between BaO/Si(001) as an example. For convenience, the lattice constant of BaO is taken to be the same as that of Si (5.43 Å). The left part of Figure 5.1 shows the structural model, which consists of five layers of Si and six layers of BaO with the connecting atom being O. The sample files for this calculation are in the directory "sample/BaO_Si001."

The "structure" block of an input parameter file is as follows.

```
structure{
        unit_cell_type=bravais
        unit_cell{
          !! a_Si=5.43 A, c-axis=5*a_Si
          !! (c.f. a_BaO=5.52 A)
          !#units angstrom degree
          a = 3.83958982184, b= 3.83958982184, c= 27.15
          alpha=90.0, beta=90.0, gamma=90.0
        }

        symmetry{
           tspace{
              system = primitive
              generators {
                 !#tag rotation tx  ty  tz
                       E         0   0   0
                       C2z       0   0   0
              }
           }
           sw_inversion = off
        }
        magnetic_state = para  !{para|af|ferro}

        atom_list{
             coordinate_system = internal ! {cartesian|internal}
             atoms{
             !#default mobile=no
!#tag element  rx     ry     rz     num_layer
      Ba       0.0000 0.5000 0.05   1
      O        0.5000 0.0000 0.05   1
      Ba       0.5000 0.0000 0.15   2
      O        0.0000 0.5000 0.15   2
      Ba       0.0000 0.5000 0.25   3
      O        0.5000 0.0000 0.25   3
      O        0.0000 0.5000 0.35   4
      Si       0.0000 0.0000 0.40   5
      Si       0.5000 0.0000 0.45   6
      Si       0.5000 0.5000 0.50   7
      Si       0.0000 0.5000 0.55   8
      Si       0.0000 0.0000 0.60   9
```

```
      O        0.5000 0.0000 0.65  10
      Ba       0.5000 0.0000 0.75  11
      O        0.0000 0.5000 0.75  11
      Ba       0.0000 0.5000 0.85  12
      O        0.5000 0.0000 0.85  12
      Ba       0.5000 0.0000 0.95  13
      O        0.0000 0.5000 0.95  13
            }
        }
        element_list{ !#tag element  atomicnumber  zeta  dev
                                Si            14  0.00  1.5
                                Ba            56  0.00  1.5
                                O              8  0.00  1.5
        }
}
```

In this example, "mobile" is set as "no," because the optimization calculation takes a significant amount of time.

5.1.2.2 Atom-divided local density of states

To calculate an atom-divided local density of states (ALDOS), edit the "postprocessing" block in an input parameter file as follows. Add "dos" and "ldos" sub-blocks in the "postprocessing" block. Set the "sw_dos" tag in the "dos" sub-block and the "sw_aldos" tag in the "ldos" sub-block to be "ON."

```
Postprocessing{
   dos{
       sw_dos = ON
       method = g
   }
   ldos{
      sw_aldos = ON
      aldos{
         crtdst = 6.0 bohr
         naldos_from = 1
         naldos_to   = 19
      }
   }
}
```

The "crtdst" tag specifies the length at which the Voronoi polyhedrons are cut. Regions that are farther than this value from any atom are treated as vacuum. The local DOS for vacuum is output as atom "number of atoms + 1" in the file "dos.data." Tags "naldos_from" and "naldos_to" are used to indicate atoms for which ALDOS are calculated. In the example, ALDOS are calculated for atoms from 1 to 19 that appear in the atom list of an input parameter file. If these tags are not specified, ALDOS are calculated for all atoms in the list. To calculate ALDOS, the column "aldos" of the atom list in an input parameter file is also available. If this column is "off," the DOS for that atom is not calculated. The tags "naldos_from" and "naldos_to" are superior to "aldos."

Calculation results are output to "dos.dat." To draw a graph of ALDOS, a PHASE tool "dos.pl" is useful. Execute this Perl script as follows, and files "dos_a001.eps," "dos_a002.eps,"…, "dos_axxx.eps" are generated.

```
 % ../../../tools/bin/dos.pl dos.data -erange=-30,5 -dosrange=0,12 -mode=atom
```

Calculated ALDOS for the BaO/Si(001) interface are shown on the right in Figure 5.1. This figure clearly shows the characteristics of Si, Ba, and O atoms in the interface.

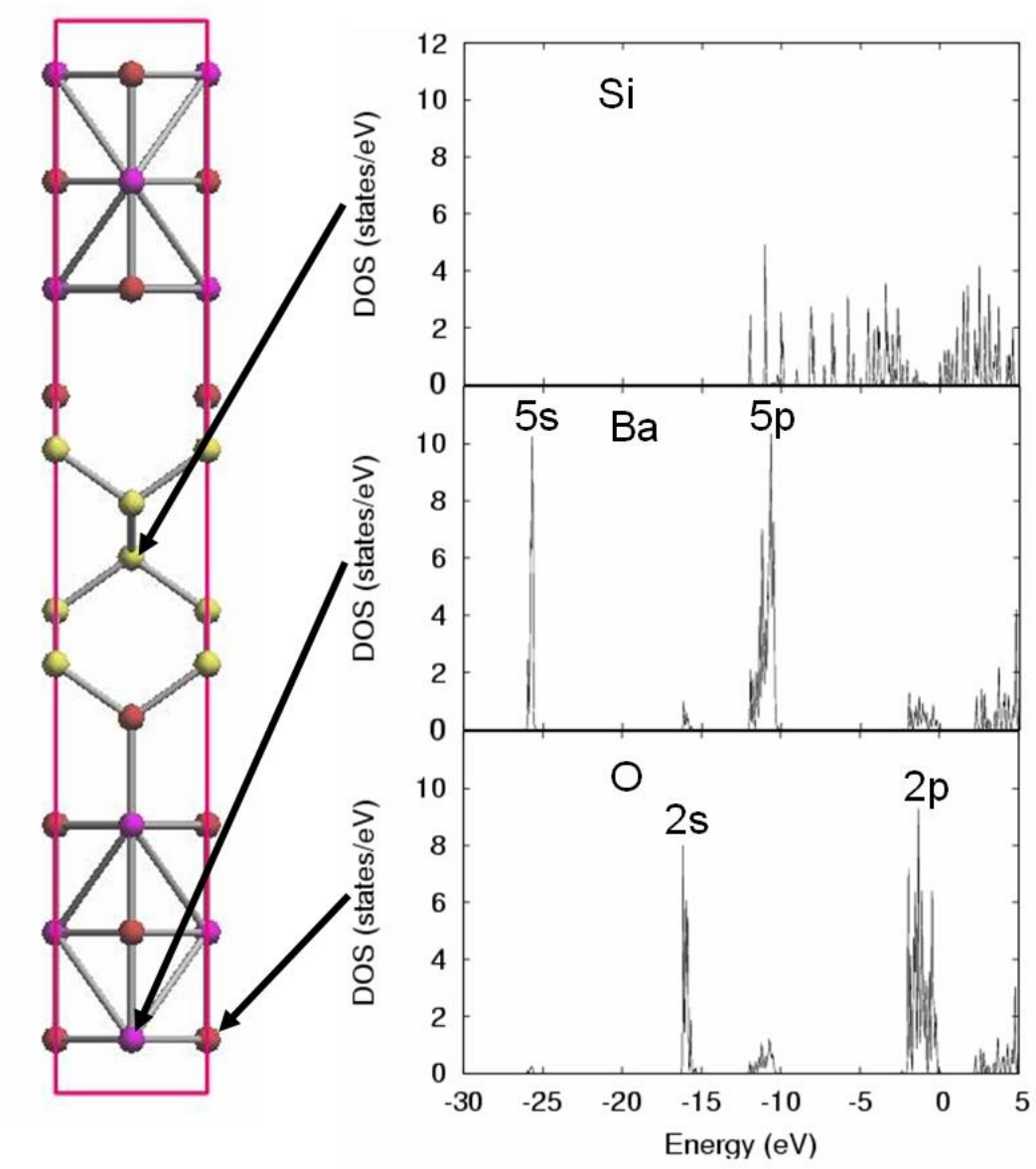


Figure 5.1 Atom-divided local density of states for a BaO/Si(001) interface. On the right, the upper panel is for a Si atom at the center of Si layers, and the middle panel is for a Ba atom at the center of BaO layers, and the bottom one is for an O atom at the center of BaO layers.

5.1.2.3 Layer-divided local density of states

To calculate layer-divided local density of states (LayerDOS), edit the "`postprocessing`" block in an input parameter file as follows. Add "`dos`" and "`ldos`" sub-blocks in the "`postprocessing`" block. Set the "`sw_dos`" tag in the "`dos`" sub-block and the "`sw_layerdos`" tag in the "`ldos`" sub-block to be "`ON`."

```
dos{
     sw_dos = ON
     method = g
}
ldos{
   sw_layerdos = ON
   layerdos{
      slicing_way = by_atomic_positions !{regular_intervals|by_atomic_positions
```

```
}
         deltaz = 1.0 angstrom
         normal_axis = 3crtdst
         crtdst = 3.5 bohr
      }
   }
```

The "normal_axis" tag specifies the direction normal to the divided layers; "1" means the direction is along the "a_vector," "2" means along the "b_vector," and "3" means along the "c_vector." If "by_atomic_positions" is set to the "slicing_way" tag, LayerDOS output depends on an atomic coordinate of the defined axis. In this case, the "num_layer" column in the atom list of an input parameter file specifies which atoms are classified into which layer. In the example input parameter file shown above, atoms are classified into 13 layers. If "regular_intervals" is set to the "slicing_way" tag, the unit cell is divided into layers of widths "deltaz." The "crtdst" tag specifies the distance from the outermost atoms to which LayerDOS are calculated. This tag is meaningless if a slab model is not used in the calculation.

Information about the range of each divided layer is output to a logfile "output000" as follows.

```
 !!ldos     no,         min,            max
 !!ldos     1          0.00000000          5.13060607
 !!ldos     2          5.13060607         10.26121214
 !!ldos     3         10.26121214         15.39181821
 !!ldos     4         15.39181821         19.23977276
 !!ldos     5         19.23977276         21.80507579
 !!ldos     6         21.80507579         24.37037883
 !!ldos     7         24.37037883         26.93568186
 !!ldos     8         26.93568186         29.50098489
 !!ldos     9         29.50098489         32.06628793
 !!ldos    10         32.06628793         35.91424248
 !!ldos    11         35.91424248         41.04484855
 !!ldos    12         41.04484855         46.17545462
 !!ldos    13         46.17545462         51.30606069
 !!ldos    14          0.00000000          0.00000000
```

Here "no" means a layer number, while "min" and "max" mean the lower and upper limits of a layer in atomic units, respectively. The last line in the list corresponds to the sum of the other areas.
The calculation results are output to file "dos.data." To draw a graph of DOS, "dos.pl" is available. Execute it as follows, and files "dos_l001.eps," "dos_l002.eps,"…," "dos_lxxx.eps" are created.

```
 % ../../../tools/bin/dos.pl dos.data -erange=-20,5 -dosrange=0,20 -mode=layer
```

The calculated LayerDOS for the BaO/Si(001) interface are shown in Figure 5.2.

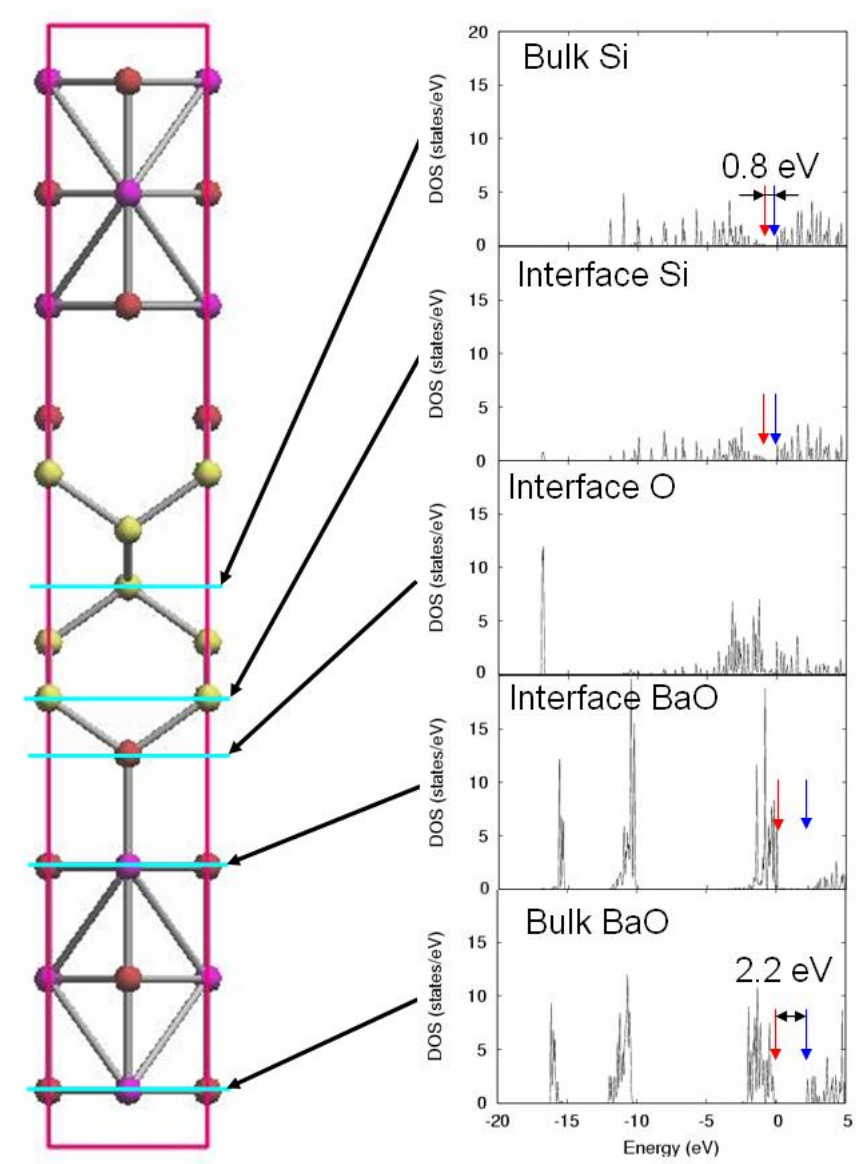


Figure 5.2 Layer-divided local density of states for a BaO/Si(001) interface. On the right, the upper panel is for the center layer of a Si slab, the second panel is for a Si layer at the interface, the third panel is for an O layer at the interface, the fourth panel is for a BaO layer at the interface, and the bottom panel is for a center layer of the BaO slab.

5.1.2.4 Energy-dependent charge density

To calculate an energy-dependent charge density, edit the "`postprocessing`" block in an input parameter file as follows. Add a "`charge`" sub-block in the "`postprocessing`" block and in the "`charge`" sub-block add a "`partial_charge`" sub-block. In it, set the "`sw_partial_charge`" tag to be "ON." The tags "`Erange_max`" and "`Erange_min`" mean the maximum and minimum of the energy range, respectively, for which the user wants to calculate the energy-dependent charge density. For these two tags, energy values are based on the Fermi energy for metals or on the top of the valance band for insulators. The tag "`Erange_delta`" means the width of energy windows; then the number of energy windows is calculated by (`Erange_max`–`Erange_min`) / `Erange_delta`. Note that two additional energy windows are calculated and output: one is just above `Erange_max` and the other is just below `Erange_min`.

```
    dos{
```

```
     sw_dos = ON
    method = g
  }
  charge{
    sw_charge_rspace    = On
    filetype            = cube !{cube|density_only}
    title = "a BaO/Si(001) interface"
    partial_charge{
       sw_partial_charge      = On
       Erange_min            = -0.50    eV
       Erange_max            =  0.50 eV
       Erange_delta           = 0.05 eV
       partial_charge_filetype = individual
    }
}
```

Information about the energy window for each energy-dependent charge density is output to the logfile "output000" as follows.

```
 !pc nEwindows =   20, nvb_windows =   10, ncb_windows =   10 <<m_ESoc_set_nEwindows_pc>>
 !pc    iw  if_elec_state              erange(hartree)                        erange(eV)
 !pc                              (asis)                 (shifted)              (shifted)
 !pc     1       1          (  0.094537  0.096374 ) ( -0.018375 -0.016537 ) ( -0.500000 -0.450000 )
 !pc     2       1          (  0.096374  0.098211 ) ( -0.016537 -0.014700 ) ( -0.450000 -0.400000 )
 !pc     3       1          (  0.098211  0.100049 ) ( -0.014700 -0.012862 ) ( -0.400000 -0.350000 )
 !pc     4       1          (  0.100049  0.101886 ) ( -0.012862 -0.011025 ) ( -0.350000 -0.300000 )
 !pc     5       0          (  0.101886  0.103724 ) ( -0.011025 -0.009187 ) ( -0.300000 -0.250000 )
 !pc     6       1          (  0.103724  0.105561 ) ( -0.009187 -0.007350 ) ( -0.250000 -0.200000 )
 !pc     7       1          (  0.105561  0.107399 ) ( -0.007350 -0.005512 ) ( -0.200000 -0.150000 )
 !pc     8       0          (  0.107399  0.109236 ) ( -0.005512 -0.003675 ) ( -0.150000 -0.100000 )
 !pc     9       0          (  0.109236  0.111074 ) ( -0.003675 -0.001837 ) ( -0.100000 -0.050000 )
 !pc    10       1          (  0.111074  0.112911 ) ( -0.001837  0.000000 ) ( -0.050000  0.000000 )
 !pc    11       1          (  0.112911  0.114749 ) (  0.000000  0.001837 ) (  0.000000  0.050000 )
 !pc    12       0          (  0.114749  0.116586 ) (  0.001837  0.003675 ) (  0.050000  0.100000 )
 !pc    13       0          (  0.116586  0.118424 ) (  0.003675  0.005512 ) (  0.100000  0.150000 )
 !pc    14       0          (  0.118424  0.120261 ) (  0.005512  0.007350 ) (  0.150000  0.200000 )
 !pc    15       0          (  0.120261  0.122099 ) (  0.007350  0.009187 ) (  0.200000  0.250000 )
 !pc    16       1          (  0.122099  0.123936 ) (  0.009187  0.011025 ) (  0.250000  0.300000 )
 !pc    17       1          (  0.123936  0.125773 ) (  0.011025  0.012862 ) (  0.300000  0.350000 )
 !pc    18       0          (  0.125773  0.127611 ) (  0.012862  0.014700 ) (  0.350000  0.400000 )
 !pc    19       0          (  0.127611  0.129448 ) (  0.014700  0.016537 ) (  0.400000  0.450000 )
 !pc    20       0          (  0.129448  0.131286 ) (  0.016537  0.018375 ) (  0.450000  0.500000 )
```

Here "nEwindows" means the number of energy windows; "nvb_windows" is the number of energy windows for valence-band states, and "ncb_windows" is that for conduction-band states. The quantity "iw" is a window number. The parameter "if_elec_state" indicates whether there are electronic states in the corresponding energy window: "0" means there are no electronic states in the energy window, while "1" means that one or more electronic states exist in the energy window. The energy-window range in atomic units is "asis," while "shifted" gives the energy-window range based on the Fermi level.

When the "partial_charge_filetype" tag is set to "individual" or "separate," each charge density file is output separately with its file name being "nfchr.00xx.cube," where "nfchr" comes from file_names.data and "xx" comes from "iw." If spin freedom is considered, two files are created: one is "nfchr.up.00xx.cube," and the other is "nfchr.down.00xx.cube." When "if_elec_state" is "0," the corresponding charge density file is not created. When the "partial_charge_filetype" tag is set to "integrated," the charge density data for all energy windows are output to one file in which "PARTIALCHARGE" is written above each set of charge density data, and "END" is written below the data.

The calculated energy-dependent charge densities for the BaO/Si(001) interface are shown in Figure 5.3.

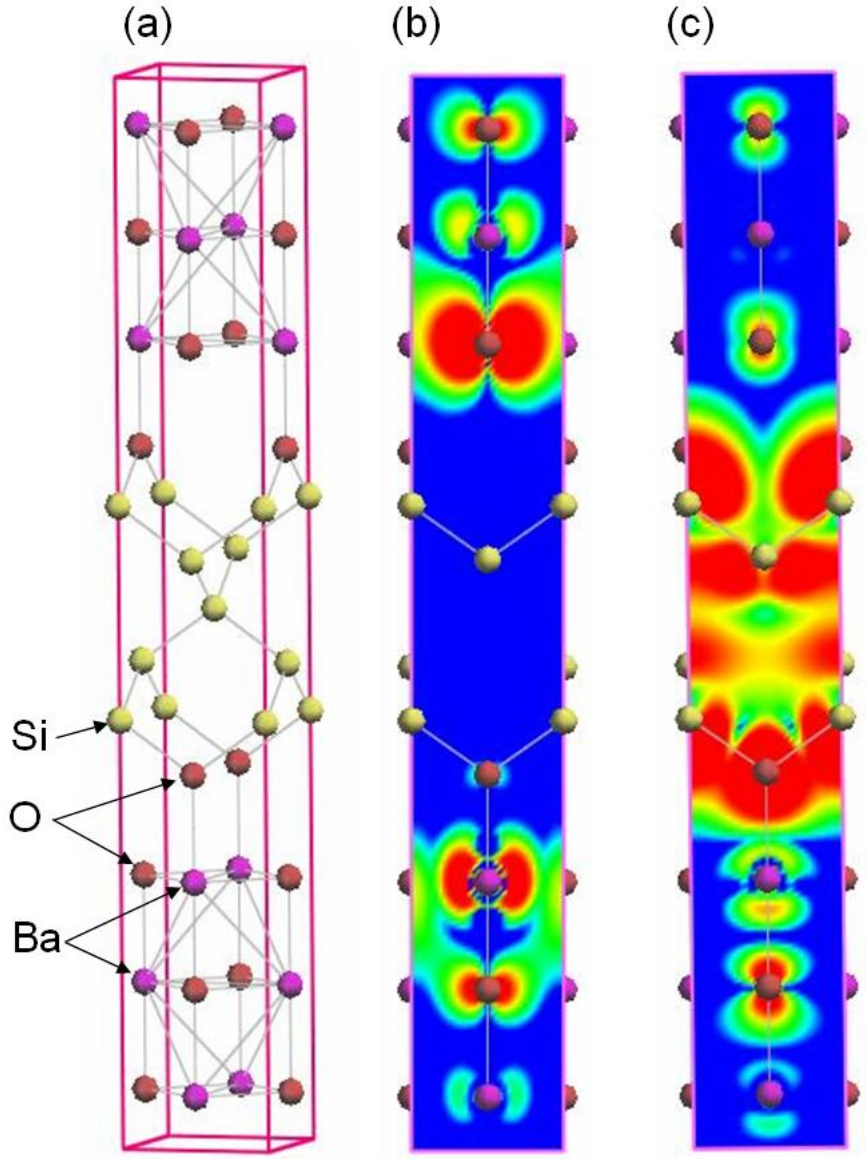


Figure 5.3 Energy-dependent charge density distributions for a BaO/Si(001) interface. (a) Structural model of a BaO/Si(001) interface. (b) Charge density for the energy range from −0.05 eV to 0 eV (Fermi energy). (c) Charge density for the energy range from 0 eV to 0.05 eV. Blue indicates less charge, and red indicates more charge.

5.1.3 Projected density of states

PHASE has a function to calculate the projected density of states (PDOS). This section describes how to calculate PDOS.

5.1.3.1 Input parameters

To calculate PDOS, the **projector_list** block is defined to specify the orbitals projected.

```
accuraccy{
  ...
  projector_list{
    projectors{
      #tag no group radius l t
      1 1 1.0 0 1
      2 1 1.0 1 1
      3 2 1.5 0 2
      4 2 1.5 1 2
      5 2 1.5 2 2
    }
  }
}
```

Here the column labeled **no** contains identification numbers for orbitals. This can be omitted. The **group** specifies "orbital group." Give the same numbers to the orbitals that you want to treat as the same group. The **radius** indicates the orbital radius in units of Bohr. Half of the atomic distance may be appropriate; the default is 1 Bohr. The column labeled **l** contains the orbital angular momentum. The values 0, 1, 2, and 3 correspond to orbitals s, p, d, and f, respectively. The column **t** contains the principal quantum numbers. However, this principal quantum number is counted from the pseudopotential and is 1 in most cases. Some pseudopotential files contain two orbitals whose angular momenta are the same. In such cases, the orbital with a higher energy should be used when the variable **t** is set to 2.

Next, assign the above-defined projectors to atoms. These projectors can be assigned by adding the **proj_group** attribute to the **atom_list** block as follows:

```
structure{
  atom_list{
    atoms{
      #tag element rx ry rz mobile proj_group
      Fe1 0.0 0.0 0.14783 on 1
      Fe2 0.0 0.0 0.35217 on 2
      Fe1 0.0 0.0 0.85217 on 1
      Fe2 0.0 0.0 0.64783 on 2
      ...
      ...
    }
  }
}
```

Here is a correspondence table between magnetic quantum numbers and orbital characteristics:

index	$l=0$	$l=1$	$l=2$	$l=3$
1	s	x	$3z^2-r^2$	$z(5z^2-3r^2)$
2		y	x^2-y^2	$x(5z^2-3r^2)$
3		z	xy	$y(5z^2-3r^2)$

4			yz	$z(x^2-y^2)$
5			zx	xyz
6				$x(x^2-3y^2)$
7				$y(3x^2-y^2)$

In this example, group 1 and group 2 are defined as orbital groups for Fe1 and Fe2, respectively. Different groups must be assigned to different elements.

Set the **sw_pdos** switch in the **postprocessing** block to "on."

```
postprocessing{
  ...
  pdos{
    sw_pdos = on
  }
}
```

The PDOS is calculated by the same methods as normal DOS.

5.1.3.2 Output

```
PDOS: ia= 2 l= 1 m= 1 t= 1
No. E(hr.) dos(hr.) E(eV) dos(eV) sum
6 -1.95781 0.0000000000 -56.762838 0.0000000000 0.0000000000
16 -1.95681 0.0000000000 -56.735626 0.0000000000 0.0000000000
26 -1.95581 0.0000000000 -56.708415 0.0000000000 0.0000000000
36 -1.95481 0.0000000000 -56.681204 0.0000000000 0.0000000000
46 -1.95381 0.0085366260 -56.653992 0.0003137151 0.0000002437
56 -1.95281 0.0176460501 -56.626781 0.0006484801 0.0000254127
```

The first line beginning with "PDOS" indicates the beginning of PDOS data. The variables **ia**, **l**, **m**, and **t** indicate the atom ID, angular momentum, magnetic quantum number, and principal quantum number of the projected orbitals. The next lines contain the PDOS, which are printed in the same data format as the normal DOS. The relationships between the magnetic quantum numbers and orbital characteristics are shown in the previous table.

The generated PDOS file, **dos.data,** is processed by the script **dos.pl** with –**mode=projected** option.

```
% dos.pl dos.data -mode=projected -color -with_fermi
```

After execution, an EPS format file **dos_aAAAlLmMtT.eps** is written. In this filename, AAA indicates the ID of atoms, L indicates the orbital angular momentum, M indicates the magnetic quantum number, and T indicates the principal quantum number. If the –**data=yes** option is given, DOS data files are provided for each orbital. In that case, the filename becomes **dos_aAAAlLmMtT.data**.

5.1.3.3 Example: PDOS of BaTiO3 crystal

Here we introduce a calculation example of PDOS for a $BaTiO_3$ crystal. The $BaTiO_3$ crystal forms a perovskite structure. Strictly speaking, this crystal structure is tetragonal, but it is very similar to cubic. In this example, this crystal was treated as cubic, as shown below.

```
structure{
  atom_list{
    atoms{
      #units angstrom
      #tag element rx ry rz proj_group
      Ba 0.00 0.00 0.00
```

```
    O 0.50 0.50 0.00 2
    O 0.50 0.00 0.50 2
    O 0.00 0.50 0.50 2
    Ti 0.50 0.50 0.50 1
   }
  }
  unit_cell{
   #units angstrom
   a_vector = 4 0.00 0.00
   b_vector = 0.00 4 0.00
   c_vector = 0.00 0.00 4
  }
}
```

The projector block is defined as follows:

```
accuracy{
  projector_list{
    projectors{
      #tag no group radius l
      1 1 1.0 2
      2 2 1.0 1
    }
  }
}
```

In the above example, group 1, in which l is 2 (i.e., d-orbital), is assigned to the Ti atom, and group 2, in which l is 1 (i.e., p-orbital) is assigned to an O atom.

The **sw_pdos** switch in the **postprocessing** block is set to "on" to calculate PDOS.

```
postprocessing{
  dos{
    sw_dos = on
    method = tetrahedral
  }
  pdos{
    sw_pdos = on
  }
}
```

In this example, DOS is calculated by the tetrahedral method. Therefore, k-sampling must be performed by the mesh method, and the tetrahedral method needs to be employed for smearing.

エラー! 参照元が見つかりません。 shows the total DOS for the $BaTiO_3$ crystal, and **エラー! 参照元が見つかりません。** shows the PDOS for d-orbitals of the Ti atom.

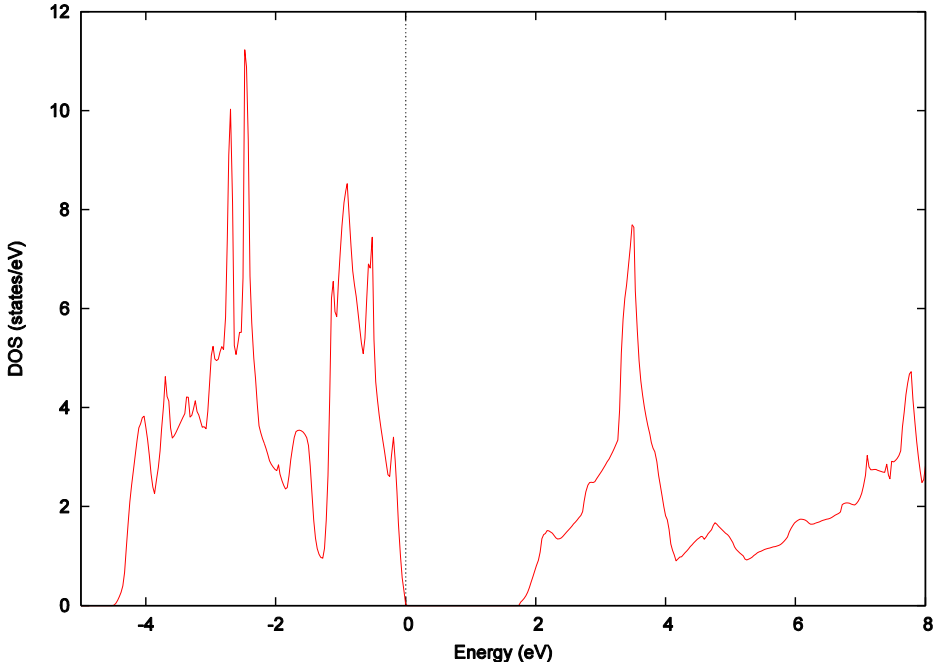


Figure 5.4 Total DOS of a $BaTiO_3$ crystal

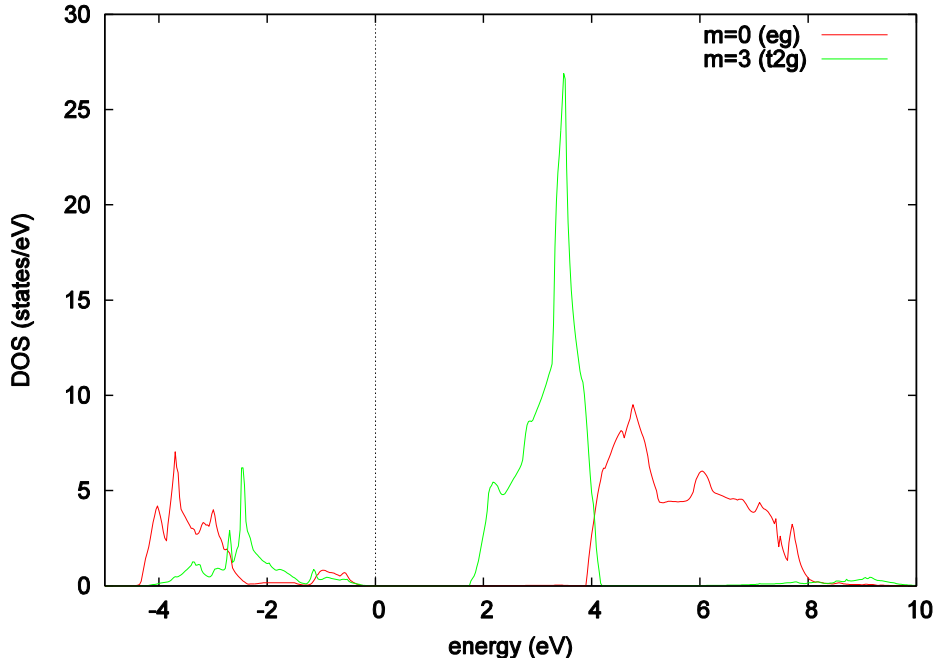


Figure 5.5 PDOS for a d-orbital of a Ti atom

5.1.4 Positron lifetime

5.1.4.1 Functions

Since a positron is an antiparticle of the electron, it has the same mass as an electron but has a positive charge. The positron annihilates an electron, resulting in the emission of a γ-ray. This annihilation can be used to study material defects, and in general, the quality of materials. To obtain useful information from positron annihilation experiments, it is necessary to compare the experimental results with first-principles calculations. PHASE has a function for predicting positron lifetimes by the following procedure.

(A)	First, electronic-structure calculations (band calculations) are carried out. The calculations are based on the pseudopotential and the plane-wave method that have been implemented in PHASE. From a band calculation, the electron density of valence electrons ρ_v can be obtained. The electron density of all electrons is given by $\rho_e = \rho_v + \rho_c$, (1) where ρ_c denotes the density of core electrons. The published pseudopotential data file, which is created by CIAO, contains information about electron densities of core electrons in free atoms. We read this data to evaluate equation (1).
(B)	The positron wave function ψ_+ is given by the following equation (in atomic units), $\left[-\frac{1}{2}\Delta - \int d\overrightarrow{r'} \frac{\rho_e(\overrightarrow{r'}) - \rho_n(\overrightarrow{r'})}{\left\vert\vec{r} - \overrightarrow{r'}\right\vert} + \mu_c(\rho(\vec{r}))\right]\psi_+(\vec{r}) = \varepsilon\psi_+(cr)$, (2) where μ_c denotes the potential energy derived from the electron–positron correlation, and ρ_n represents the point charge of the nucleus. Now, since there is only one positron assumed in the solid, it is sufficient to only calculate the most stable eigenstate. Therefore, eigenstates of the positron belong to the Γ point in the Brillouin zone. This wave function can be expanded by plane waves, $\psi_+ = \sum_{\vec{G}} C_{\vec{G}} \exp\left(\mathrm{i}\vec{G}\cdot\vec{r}\right)$. (3) Here to suppress the finite summation over the reciprocal lattice periodic vector $\vec{G}$, we need to set an upper limit on the kinetic energy of the plane wave.
(C)	The electron density of the positron is obtained from $\rho_p(\vec{r}) = \vert\psi_+\vert^2$.
(D)	Using electron and positron charge densities, the positron lifetime is evaluated by $\frac{1}{\tau} = \pi r_e^2 c \int d\,r \rho_e(r)\rho_p(r)\Gamma(\rho_e)$, (4) where r_e is the classical radius of an electron, and c is the speed of light. The quantity Γ is an enhancement factor caused by electron–positron correlations. In PHASE, evaluation of the above equation is performed under the following approximation, $\rho_e\Gamma(\rho_e) \cong \rho_v\Gamma(\rho_v) + \rho_c\Gamma(\rho_c)$. (5) For this approximation to hold, the overlap of the distributions of valence electrons and core electrons should be small.

In the calculation of the correlation of electron and positron, the local density approximation is used. In other words, based on calculation results, when there is a single positron in a homogeneous electron gas, the correlation potential and the enhancement factor are given as functions of electron density. The following equation has been proposed for the enhancement factor [Puska95],

$$\Gamma = 1 + 1.23r_s + 0.9889r_s^{3/2} - 1.482r_s^2 + 0.3956_s^{5/2} + r_s^3/6$$

where $\frac{4\pi}{3}r_s^3 = 1/\rho_e$. In addition, in systems with a gap (dielectric), since the screening effect of electrons is much smaller than that of the metal, it is recommended that this expression be corrected for Γ as follows [Puska91], [Nakamoto07].

$$\Gamma = 1 + 1.23r_s + 0.9889r_s^{3/2} - 1.482r_s^2 + 0.3956_s^{5/2} + (1 - 1/\varepsilon_{ele})r_s^3/6$$

where ε_{ele} is the dielectric constant of the electron system. If the dielectric constant has not been determined by experiment, it can be evaluated using UVSOR, which is based on density functional theory. For details on the calculation method, please refer to the literature [akamoto07].

5.1.4.2 Input file

Here is an example calculation for a Si crystal. In the sample of PHASE, there is a folder named positron Si, in which there are folders named "input" and "output." In the folder named "input," there is the input file for the calculation of positron lifetimes in Si crystals using PHASE.

The file that contains input parameters for the calculation of positron lifetimes is samples/positron Si/input/nfposnew.data. Here we explain only those parts of the file that relate to the calculation of positron lifetimes.

- Use the control tag to enable calculation of a positron lifetime,

```
Control{
  positron = BULK
}
```

Declaring positron = BULK causes the electronic-structure calculation (band calculation) to be done first followed by calculation of the positron lifetime.

- Use the accuracy tag to specify options for the positron lifetime calculation,

```
accuracy{
    cutoff_pwf = 50.00 rydberg
    positron_convergence{
      num_extra_bands = 8
      delta_eigenvalue = 1.d-8 rydberg
      succession = 6
      num_max_iteration = 32000
      dtim = 0.01
      epsilon_ele = 12}
}
```

cutoff_pwf = 50.00 rydberg	Cutoff energy for expanded positron wave functions [See equation (3)]
positron_convergence {}	Positron wave functions are obtained by an iterative calculation; this tag specifies options for identifying a converged solution when solving equation (2).
num_extra_bands = 8	For the eigenstate of the positron, it is sufficient to only calculate the ground state. However, for the converged solution obtained by the iterative calculation, wave functions having higher energies than the ground state should also be calculated. This tag specifies the number of those extra wave functions. Note that the resulting wave functions all belong to points in the Brillouin zone.
delta_eigenvalue = 1.d-8 rydberg	Refer to the explanation of line 5.
succession = 6	In the iterative calculation, if physical quantities from the previous and current iterations (refer to line 7) are consistent within a range given by line

4, and if they are continuous over the times specified by line 5, the calculation is considered to have converged.

num_max_iteration = 32000 — If the calculation has not converged by this number of iterations, the calculation will stop.

dtim = 0.01 — In the iterative calculation, dtim measures the extent of change permitted from one wave function to the next. When dtim is large, the calculation will converge faster. However, if it is very large, no converged solution will be obtained. In contrast, when dtim is small, the calculation will be more stable, but the calculation time will increase. Thus, depending to the system being studied, it is recommended that users seek an optimum value for dtim.

epsilon_ele = 12 — This tag is used when the system has a gap; hence, a correction is needed that involves the dielectric constant of the electron system of LDA. In this example, the tag is set to "12," which is the dielectric constant for Si. If the system has no gap (e.g., if it is a metal), then no value should be assigned to this parameter, and line 8 should be deleted.

5.1.4.3 Output file

After performing the calculation for a positron lifetime, an output file and three cube files will be generated. They are placed in the directory /samples/positron Si/output/.

13. Log outputfile, output000

The first part of this file contains information related to the calculation of the electronic bands of Si. After the calculation of electronic bands, the charge density of electrons will be obtained, and then the calculation of the positron lifetime will be performed.

In the output, the part related to positron calculations starts at

"--- initial positron energy eigen values ---"

The positron wave function is determined by the iterative calculation. In the following output, at the first iteration, there is an eigenvalue 14.6379 eV. There are also extra bands (14.9628460558–15.0292289699) that are higher than the positron eigenvalue. In the second iteration, the eigenvalue becomes 0.0021898139 eV.

```
--- initial positron energy eigen values ---
 === positron eigen values ===
     14.6378982055
 -- extra_bands --
     14.9628460558     14.6842242625     14.9879179620     15.2755174303
     14.8070539395     14.6061318397     14.8086346971     15.0292289699

 === positron eigen values ===
      0.0021898139
 -- extra_bands --
      0.0892687578      0.1056325893      0.2037689630      0.2140559068
      0.3115605599      0.3359746459      0.3540270556      0.4738130045
```

Then, the file below contains the following output:.

```
************************************************
 positron lifetime(ps)    220.184723312044
```

```
 core rate    3.79328791767622       %
 ***********************************************
```

This means that, after the iterative calculation converges, the calculated positron lifetime is 220 ps. Here the core rate is the percentage of the annihilation rate of core electrons relative to the total annihilation rate.

14. Cube file

After the calculation, the following files are created: electron.cube, positron.cube, and ep_pair.cube. These cube files contain the charge distribution of electrons, the charge distribution of the positron, and the distribution of the electron–positron pair, respectively. The latter can be visualized using the Biostation viewer. (This software is not part of PHASE, but it can be downloaded from the web.)

Figure 5.6 shows distributions computed for the example Si crystal. This figure shows that the valence electron mainly exists in the bond region, while the positron exists in the interstitial region. While the positron wave function is energetically favorable when the kinetic energy of spreading is low, the positron commonly tends to be in the interstitial region. The distribution of the electron–positron pair in Fig. 5.7(c) indicates that when the distribution is high, annihilation of the positron occurs at a high rate.

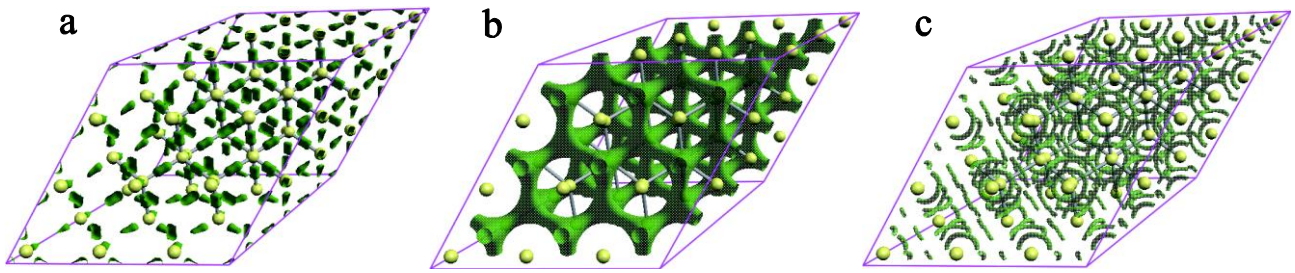


Figure 5.6 Distributions of (a) electron, (b) positron, and (c) electron–positron pair computed for a Si crystal.

5.1.4.4 Notes on calculation of positron lifetimes

Summary of the notes upon the calculation of positron lifetime.

- Selection of pseudopotential

There may be a semi-core state that depends on the chemical element.
A semi-core state arises when the overlap between the core and valence electrons cannot be neglevted.. The semi-core electrons need to be classified into valence electrons in making pseudopotentials, If no such published pseudopotential is available for a chemical element, a pseudopotential of this element can be created by using CIAO .

- Selection of cutoff energy

In the band calculation for the Si crystal, the input file contains the following,

```
accuracy{
    cutoff_wf =  50.00  rydberg  ! cke_wf
    cutoff_cd =  200.00  rydberg  ! cke_cd
    cutoff_pwf = 50.00 rydberg
```

These set the cutoff energies for electron wave function, charge density, and positron wave function. To confirm that the positron-lifetime calculation has sufficiently converged, change these cutoff values and

repeat the calculation.

- output

The positron wave function and the electron wave function are provided by the iterative calculation. For each iteration, the output000 file, which is the output for the last iteration, contains the following:

```
=== positron eigen values ===
     -0.5674635596
 -- extra_bands --
     -0.0490686179     -0.0460091253     -0.0446118499     -0.0275856742
     -0.0102856694      0.0069403602      0.0274419414      0.2284487012
lifetime:    220.180365487100        220.179503204077
```

This output, near the end of the calculation, confirms that the positron eigenvalue is sufficiently converged. In this output (output000) from the sample calculation, there are the eigenvalues

```
-0.5674635596
-0.5674635638
```

etc., suggesting that the calculation is sufficiently converged. In addition, the successive lifetime values 220.180365487100 ps and 220.179503204077 ps suggest that a converged value for the lifetime is being approached. If the electronic band calculation is converged, and it can be observed the 3.4.4, it is considered that this calculation is sufficient. It is recommended that you first do a calculation for a relatively simple system and then confirm the calculation results by comparing with experimental data. After that, you can perform calculations on the system of interest. It is fortunate that these calculations can be helpful in the analysis of positron annihilation experiments in various systems.

5.2 Atomic dynamics

5.2.2 Molecular dynamics simulation

5.2.2.1 Overview

By calculating the forces acting on atoms, molecular dynamics (MD) simulations are carried out. PHASE can perform constant-energy and constant-temperature MD simulations.

5.2.2.2 Input parameters

The following table lists blocks and variables related to MD simulations.

1st level block	2nd, 3rd level block	Tag keyword	Description
structure_evolution			Block for specifying a method for updating atomic coordinates.
		method	Specify a method for updating atomic coordinates. Options are either velocity_verlet (constant-energy MD simulation) OR temperature_control (constant-temperature MD simulation).
		dt	Specify the time step. Defaults to 100 au (nearly equals 2.4 fs).
	thermostat		Tabular block that defines thermostat.
		temp	Specify the target temperature.
		qmass	Specify the mass Q. This parameter must be given if a constant-temperature simulation is performed.
structure	atom_list		
	atoms		Tabular block that defines atomic positions.
		thermo group	This column is used to assign thermostats to atoms. This column must be defined even if only one thermostat is defined.
	element_list		Tabular block that defines elements
		mass	Specify mass of atoms. Unit is atomic unit.
printlevel			
		iprivelocity	If this variable is set to 2, velocity is also printed into the **F_DYNM** file.

5.2.2.3 Output

Atomic coordinates at each step are printed to the **F_DYNM** file. Its format is the same as that for geometry optimization.

- Atomic coordinates

Atomic coordinates are written into the **F_DYNM** file (default name is **nfdynm.data**) defined in **file_names.data**.

A Perl script, **animate.pl,** can convert the format of this file so it can be read by the PHASE viewer.

The velocities of atoms are also printed to this file, if the **iprivelocity** variable in the **printoutlevel** block is set to more than 2. The velocities are printed in atomic units after printing the atomic forces.

- Total energy

The total energy at each step is dumped into a file designated by the **F_ENF** keyword in the **file_name.data** (default filename is **nfefn.data**). The following shows an example of this file.

```
  iter_ion, iter_total, etotal, ekina, econst, forcmx
     1      18    -7.8953179624     0.0000042358    -7.8953179624     0.0186964345
     2      30    -7.8953851218     0.0000665502    -7.8953185716     0.0183575424
     3      43    -7.8955768901     0.0002565396    -7.8953203505     0.0173392067
     4      56    -7.8958649874     0.0005418445    -7.8953231430     0.0156398790
     5      69    -7.8962052587     0.0008785990    -7.8953266596     0.0132645441
     6      83    -7.8965425397     0.0012120826    -7.8953304571     0.0102355854
     7      97    -7.8968179539     0.0014840140    -7.8953339398     0.0066063151
     8     111    -7.8969784478     0.0016420281    -7.8953364197     0.0024736141
     9     125    -7.8969875377     0.0016502900    -7.8953372478     0.0020111576
    10     139    -7.8968352058     0.0014992046    -7.8953360011     0.0066379641
    11     153    -7.8965440599     0.0012113794    -7.8953326806     0.0111430822
                           ..............................
                           ..............................
                           ..............................
```

The first column indicates the number of MD steps, the second column indicates the number of total SCF calculations, the third column indicates the total potential energy, the fourth column indicates the kinetic energy of the system, the fifth column is the sum of the total potential energy and kinetic energy. The fifth column contains the conserved quantity in constant-energy MD simulations.

5.2.2.4 Usage: constant-energy MD simulation

The following is an example of the input parameters for a constant-energy MD simulation. This sample file is in **sample/molecular_dynamics/NVE**.

```
accuracy{
    cutoff_wf = 9.00 rydberg
    cutoff_cd = 36.00 rydberg
    num_bands = 8
    xctype = ldapw91
    force_convergence{
        max_force = 1.0e-8 Hartree/Bohr
    }
    initial_wavefunctions = matrix_diagon
    ksampling{
        mesh{
            nx = 4
            ny = 4
            nz = 4
        }
    }
    scf_convergence{
        delta_total_energy = 1e-12 Hartree
        succession = 3
    }
}
```

```
...
...
structure{
    unit_cell_type = primitive
    unit_cell{
        a_vector = 0.0000000000        5.1300000000        5.1300000000
        b_vector = 5.1300000000        0.0000000000        5.1300000000
        c_vector = 5.1300000000        5.1300000000        0.0000000000
    }
    atom_list{
        atoms{
            #tag element rx ry rz mobile
             Si 0.130 0.130 0.130 yes
             Si -0.130 -0.130 -0.130 yes
        }
    }
    element_list{
        #tag element atomicnumber
         Si 14
    }
}
...
...
structure_evolution{
    method = velocity_verlet
    dt = 100
}
...
...
```

In the **atoms** block, the **mobile** attribute is set to "yes." If "no" or "0" is given, the atom is fixed during the MD simulation. In this example, intentionally unstable atomic coordinates are given. To be more specific, the two silicon atoms are slightly shifted to separate one from the other in the (111) direction. The **structure_evolution** block identifies the method as "velocity_verlet." By using this method, a microcanonical ensemble MD simulation is carried out. The **dt** variable sets the time step of each cycle to "100" in atomic units. As mentioned before, this value is equivalent to 2.418×10^{-15} s. In the above example, the initial velocities of all atoms are set to "0." To give initial velocities to the atoms, the following input needs to be prepared.

```
structure_evolution{
    method = velocity_verlet
    dt  = 100
    temperature_control{
        thermostat{
            #tag temp
                 300
        }
    }
}
```

Here the **temp** variable gives the initial temperature in Kelvin. Initial velocities, given by normalized random numbers, correspond to this temperature such that the total momentum is 0. One can set different initial temperatures to each atomic species. In such cases, several target temperatures are defined in the **thermostat** block as follows:

```
structure_evolution{
    method = velocity_verlet
```

```
    dt  = 100
    temperature_control{
        thermostat{!#tag temp
                        300
                        500
                        700
        }
    }
}
```

Next, define the **thermo_group** attribute in the **atoms** block.

```
structure{
     ...
    atom_list{
        atoms{
        !#tag rx ry rz  element mobile weight  thermo_group
            0.1159672611      0.1235205209      0.1215156388    Si   1   1  1
           -0.1329067626     -0.1264216714     -0.1225370484    Si   1   1  2
            0.1273740089      0.6305999369      0.6247606249    Si   1   1  3
            ...
            ...
        }
    }
    ...
}
```

The above example indicates that the initial velocities of the first, second, and third atoms are assigned to reproduce the temperatures 300 K, 500 K, and 700 K, respectively.

Figure 5.7 shows the potential energy, kinetic energy, and total energy of this sample simulation.

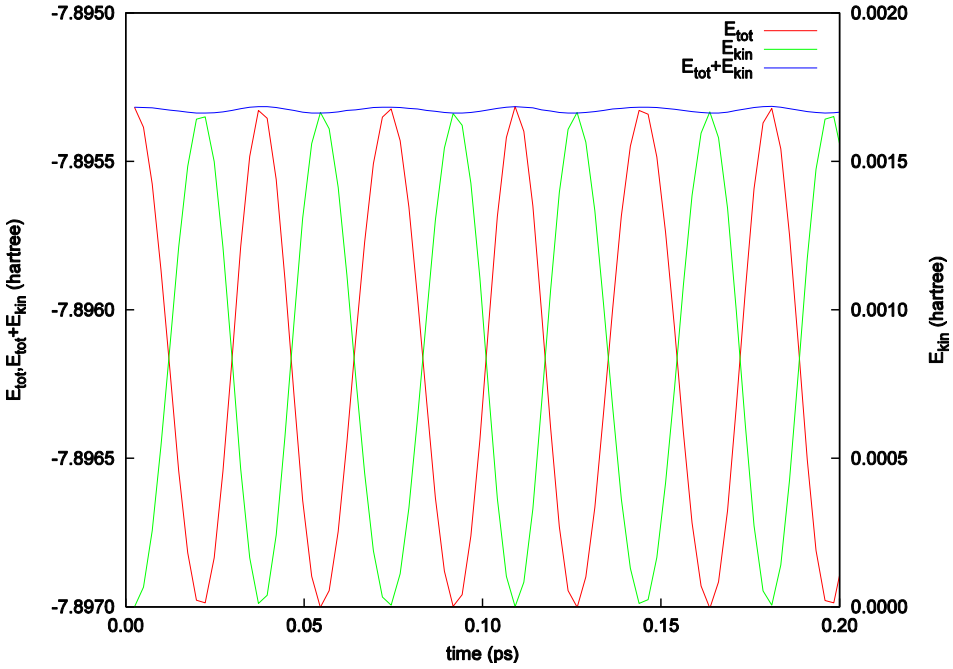


Figure 5.7 Time evolution of potential energy, kinetic energy, and total energy.

5.2.2.5 Usage: constant-temperature MD simulation

The following is an example of input parameters for a constant-temperature MD simulation. This sample file is in **sampe/molecular_dynamics/NVT**.

- Setting the thermostat

Define the **temperature_control** block as below:

```
structure_evolution{
    method = temperature_control
    dt  = 50.0
    temperature_control{
        thermostat{
            #tag temp  qmass
                 300   5000
        }
    }
}
```

In the above example, "temperature_control" is chosen for the MD method; this indicates that a constant-temperature MD simulation is to be carried out. The **dt** gives the time step in atomic units. The value "50.0" in this example is equivalent to about 1.2 fs. In addition, **temperature_control** is defined to give settings for the thermostat. The **temp** variable sets the target temperature, and **qmass** sets the effective mass Q.

To assign the above-defined temperature and mass to atoms, define the **thermo_group** attribute in the **atoms** block as follows:

```
structure{
     ...
    atom_list{
        num_atoms = 8
        cooordinate_system = internal
        atoms{
        !#tag rx ry rz  element mobile weight  thermo_group
            0.1159672611      0.1235205209      0.1215156388    Si   1   1  1
           -0.1329067626     -0.1264216714     -0.1225370484    Si   1   1  1
            0.1273740089      0.6305999369      0.6247606249    Si   1   1  1
           -0.1152089939     -0.6164829779     -0.6221565128    Si   1   1  1
            0.6299472943      0.1341313888      0.6253193197    Si   1   1  1
           -0.6305720382     -0.1290073650     -0.6187967685    Si   1   1  1
            0.6151271805      0.6206113965      0.1333834419    Si   1   1  1
           -0.6276524003     -0.6268549639     -0.1175099372    Si   1   1  1
        }
    }
    ...
}
```

In this example, the **thermo_group** attribute is defined for all atoms. The number given to this attribute corresponds to the order of thermostat parameters defined in the **thermostat** block. As well as other attributes, the default value of the thermostat parameter can be defined by the "#default" tag. Although the same group is given to the **thermo_group** in this example, you can set different groups to the atoms.

5.2.2.6 Precaution for use

There are no specific limitations for the MD simulation function. This function supports ultra-soft and PAW

pseudopotentials, parallel calculations, and continuation calculations (restarting). However, note the following.

- Masses of atoms must be correctly defined when an MD simulation is performed. The default unit of mass in PHASE is in atomic units. For example, the mass of a proton is 1822.877333 in atomic units.
- The kinetic energy E_{kin} [Hartree] is given by $E_{\mathrm{kin}} = \frac{3}{2} \times N_{\mathrm{atom}} \times k_{\mathrm{B}}T$, where N_{atom} represents the number of atoms, k_{B} represents the Boltzmann constant, and T represents the instantaneous absolute temperature. Therefore, to know the temperature of the system, divide the kinetic energy by the number of atoms, multiply by 3.1578×10^5, which is the unit conversion factor from Hartree to $k_{\mathrm{B}}T$, and then finally divide by $\frac{3}{2}$.
- The total simulation time can be obtained by multiplying the number of MD cycles by the time step given by the **dt** variable. Although the unit of time can be defined by users, the default unit is in atomic units. One can convert the time from atomic units to seconds by multiplying by 2.418×10^{-17}. For example, 100 a.u. corresponds to 2.418 fs.

- In constant-temperature MD simulations, the parameter Q should be carefully chosen. If the value of Q is very small, an artificial mode is created in the dynamics. This is caused by the thermostat and leads to a collapse of the calculation. Alternatively, if the value of Q is very large, the system requires a large number of steps to thermally equilibrate. Generally speaking, it is recommended to set the parameter Q such that the period of oscillation of the thermostat is almost equivalent to or longer than the period of characteristic oscillations of the system. The period of oscillation of the thermostat can be approximately estimated by the equation (S. Nosé, Progress of Theoretical Physics Supplement No 103, 1991, pp.1–46):

$$\tau = \frac{2\pi}{\omega} = 2\pi \left(\frac{Q}{2gk_BT} \right)^{1/2}$$

where τ and ω represent the period and frequency of the system, g is the number of degrees of freedom of the system ($3 \times$ N, where N is the number of atoms related to the thermostat), k_{B} is Boltzmann's constant, and T is the target absolute temperature of the thermostat. For example, if τ is 0.05 ps, the number of atoms is 8, and the target temperature is 300 K, then the parameter Q is estimated to about 4600 in atomic units.

5.3 Advanced DFT calculations

5.3.1 DFT+U Method

5.3.1.1 General features

The software PHASE, which is based on density functional theory (DFT), accurately calculates the electronic states of most materials. However, for strongly correlated systems, high accuracy cannot be expected owing to limitations in the local density approximation (LDA) adopted in DFT. To overcome this drawback, PHASE also supplies the LDA+U method, or alternatively DFT+U, in which repulsive interactions between localized electrons are incorporated as on-site Coulomb interactions.

Among various DFT+U models proposed, PHASE adopts a simplified rotationally invariant model, in which the total energy ($E_{\mathrm{DFT+U}}$) is written as a sum of the energy of DFT (E_{DFT}) and a "+U" correction energy. (The latter contribution is also called the Hubbard correction.) The Hubbard correction is a function of the occupation matrix ρ that is calculated on each atomic site.

$$E_{\mathrm{DFT+U}} = E_{\mathrm{DFT}} + \frac{U_{eff}}{2} \sum_{I,m,\sigma} \left\{ \rho^{I\sigma}_{m,m} - \sum_{m'} \rho^{I\sigma}_{m,m'} \rho^{I\sigma}_{m',m} \right\}$$

Here index I denotes the atomic site, m and m' are magnetic quantum numbers, and σ is the spin index. The quantity U_{eff} represents the strength of the effective Coulomb interaction.
The occupation matrix is constructed by projecting the wavefunctions onto the localized orbitals such as atomic orbitals.

$$\rho^{I\sigma}_{m,m'} = \sum_{k,n} f^{\sigma}_{kn} < \Psi^{\sigma}_{kn} | \phi^{I}_{m} >< \phi^{I}_{m'} | \Psi^{\sigma}_{kn} >$$

Here the index k denotes the wavenumber vector and n is the band index. The quantity f^{σ}_{kn} denotes the occupation number of the electronic state specified by the three indices k, n, and σ.

The Hubbard correction causes splitting of the degenerate energy levels of the localized orbitals. In particular, when the corresponding energy level is fully occupied (unoccupied), its energy is decreased (increased) by $\frac{U_{eff}}{2}$ (see Figure 5.8). The value of U_{eff} should be chosen so as to experimentally reproduce observed quantities; otherwise, use values reported in previous studies.

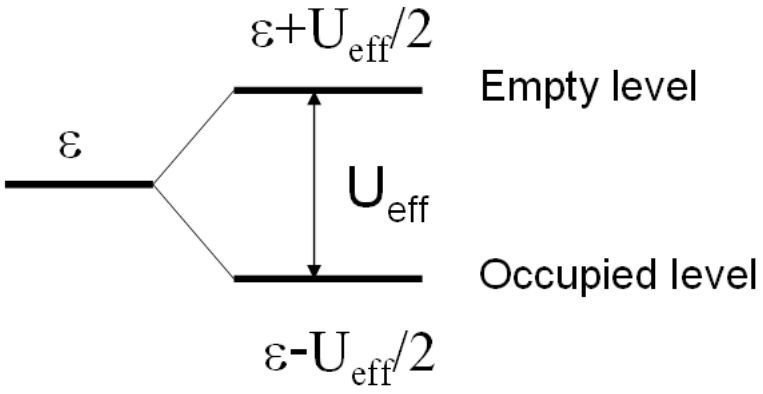


Figure 5.8 Energy level splitting caused by the Hubbard correction

5.3.1.2 Input parameters

To use the DFT+U method, the following steps are essential. First, in the "accuracy" block, you should add the "hubbard" and "projector_list" blocks. In the former, specify the strength of the effective Coulomb interaction (U_{eff}). Note that the keyword "sw_hubbard = on" is needed to declare that the Hubbard correction will be used. In the latter part, specify the radius of the atomic orbital that will be used in calculating the occupation matrix. The keyword "no" denotes the projector number, "group" denotes the projector group number, "radius" denotes the radius of the atomic orbital, and "l" denotes the azimuthal quantum number. Note that the projector number specified in the "hubbard" block corresponds to the projector number in the "projector_list" block.

```
accuracy{
        ...
        hubbard{
           sw_hubbard = on
           projectors{
             #units eV
             #tag no ueff
                  1  10.0
           }
        }
        projector_list{
          projectors{
            #tag no group radius l
                 1  1     2.75   2
          }
        }
...
}
```

Next, in the "structure" block, you should specify the atoms to which the Hubbard correction is applied. The numbers specified with the keyword "proj_group" correspond to the projector group numbers defined in the "accuracy" block. The number "0" indicates that the Hubbard correction is not to be applied to the corresponding atom.

```
structure{
  ...
  atom_list{
```

```
    coordinate_system = internal ! {cartesian|internal}
      atoms{
        !#default mobile=no
        !#tag  rx   ry   rz  element proj_group
               0.0  0.0  0.0 Sr      0
               0.5  0.5  0.5 Ti      1
               0.0  0.5  0.5 O       0
               0.5  0.0  0.5 O       0
               0.5  0.5  0.0 O       0
             }
        }
...
}
```

Finally, in the "wavefunction_solver" block, we recommend using the Davidson method to prevent electronic states from being trapped in a local energy minimum.

```
wavefunction_solver{
  solvers{
  !#tag   sol      till_n  dts  dte  itr  var  prec cmix
          Davidson     -1  0.1  0.1  100  tanh off  1
  }
}
```

5.3.1.3 Outputs

- Standard output file

In the standard output file, you will find the words "HE" and "HP" when you use the Hubbard correction. The former and latter terms correspond to the Hubbard energy and Hubbard potential energy, respectively.

```
 TOTAL ENERGY FOR     2 -TH ITER=    -79.756461901287   edel =   0.482992D+01
 KI=     45.2522902 HA=    125.6089055 XC=    -43.2979227 LO=   -147.0597534
 NL=     19.3280980 EW=    -92.0686823 PC=     12.2272681 EN=      0.0000000
 HE=      0.2533348 HP=      0.6709743
```

In the same file, you will also be able to confirm the elements of the occupation matrix on each of the atomic sites to which you apply the Hubbard correction. The keyword "is" denotes the spin index, "ia" denotes the atom index, and "l" denotes the azimuthal quantum number. Note that the dimensions of the occupation matrix are $(2l + 1) \times (2l + 1)$.

The $(m, m')^{th}$ element of this matrix indicates the occupation matrix between the atomic orbital with the magnetic quantum numbers, m and m′ ($1 <= m, m' <= 2l + 1$). The character of the m^{th} orbital used in PHASE is summarized in Table 5.1.
Subsequently, you will find the occupancy of the atomic orbitals by diagonalizing the occupation matrix. The first column indicates the eigenvalues of the occupation matrix, and the numbers on the right hand side of the colon indicate the corresponding eigenvectors.

```
Occupation Mattrix: is,ia,l=    1    2    2
   0.583   0.000   0.000   0.000   0.000
   0.000   0.583   0.000   0.000   0.000
   0.000   0.000   0.529   0.000   0.000
   0.000   0.000   0.000   0.529   0.000
   0.000   0.000   0.000   0.000   0.529
Diagonalizing Occupation Mattrix: is,ia,l=    1    2    2
```

```
   0.529:    0.000    0.000    0.000   -1.000    0.000
   0.529:    0.000    0.000    1.000    0.000    0.000
   0.529:    0.000    0.000    0.000    0.000    1.000
   0.583:    0.000    1.000    0.000    0.000    0.000
   0.583:   -1.000    0.000    0.000    0.000    0.000
```

occmat.data

In the file "occmat.data," you will find the elements of the occupation matrix at the last SCF iteration before the calculation is terminated. The first line, which contains the word "num_om," indicates the number of generated occupation matrices, N_{om}. Below that line, you will find the elements of the occupation matrix at each of the atomic sites to which you apply the Hubbard correction. The keyword "is" denotes the spin index, "ia" denotes the atom index, "iproj" denotes the projector number, "it" denotes the atom species, and "l" denotes the azimuthal quantum number. Note that the number of occupation matrices printed equalsN_{om}.

```
16 : num_om
…….
  1    3    1    3    1 : is, ia, iproj; it, l
    0.17441054E+01     -0.20464246E-02     -0.99899010E-03
   -0.20464246E-02      0.17539484E+01     -0.39442624E-02
   -0.99899010E-03     -0.39442624E-02      0.17529809E+01
  1    4    1    3    1 : is, ia, iproj; it, l
    0.17365161E+01     -0.12145064E-01     -0.11970673E-01
   -0.12145064E-01      0.17903944E+01     -0.85524320E-02
   -0.11970673E-01     -0.85524320E-02      0.17856965E+01
……
```

Table 5.1 Orbital character

magnetic quantum number	$l=0$	$l=1$	$l=2$	$l=3$
1	s	x	$3z^2-r^2$	$z(5z^2-3r^2)$
2		y	x^2-y^2	$x(5z^2-3r^2)$
3		z	xy	$y(5z^2-3r^2)$
4			yz	$z(x^2-y^2)$
5			zx	xyz
6				$x(x^2-3y^2)$
7				$y(3x^2-y^2)$

5.3.1.4 Sample : cubic SrTiO3

In the directory "sample/DFT+U/SrTiO3/cubic+u," you will find the following samples.

- DFT+U/SrTiO3/cubic+u （U_{eff} is set to 10 eV for the Ti 3d orbitals.）
- DFT+U/SrTiO3/cubic （U_{eff} is set to 0 eV）

These two samples are compared in Figure 5.9.

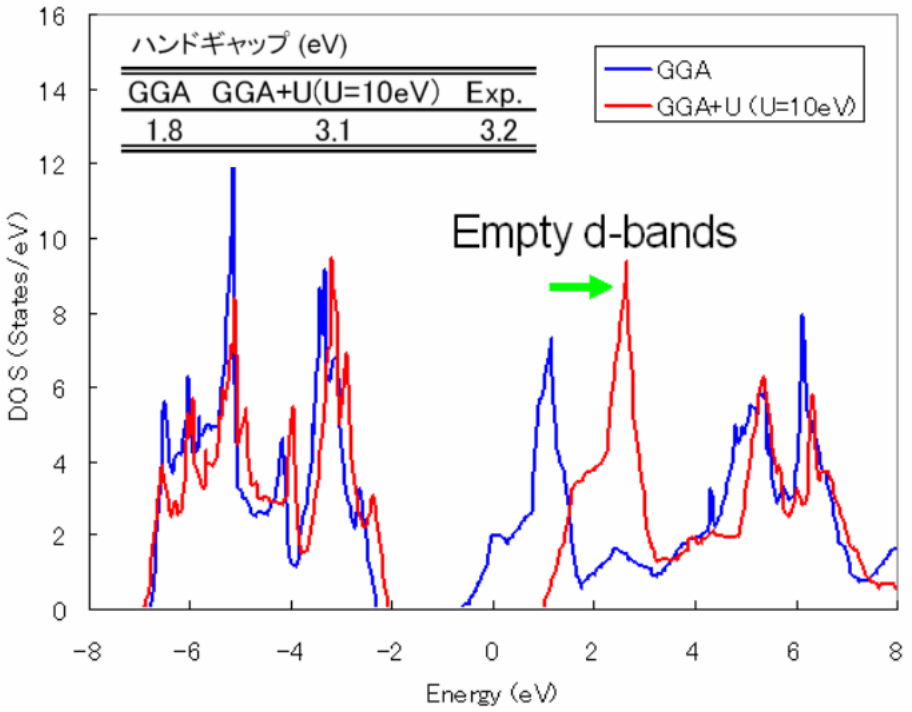


Figure 5.9 Density of states for cubic SrTiO3

5.3.1.5 Sample : cubic LaVO3

- DFT+U/LaVO3/cubic+u （U_{eff} is set to 20 eV for the La 4f orbitals.）
- DFT+U/LaVO3/cubic （U_{eff} is set to 0 eV）

In the latter, the 4f band appears at 1.5 eV above the Fermi level.
In the former, this band appears at 8.0 eV above the Fermi level.

5.3.1.6 Sample : orthrombic LaVO3

- DFT+U/LaVO3/orthrombic+u

（U_{eff} is set to 5 for the V 3d orbital and to 20 eV for La 4f orbital.）

- DFT+U/LaVO3/orthrombic （U_{eff} is set to 0 eV）

In the former, the magnetic moments on the V atoms are aligned in an anti-ferromagnetic manner..

5.3.1.7 Sample : cubic FeO

- DFT+U/FeO/gga+u（U_{eff} is 5 eV for the Fe 3d orbital.）
- DFT+U/FeO/gga　（U_{eff} is set to 0 eV）

Note that these two samples use the data in file occmat.data as initial values for the occupation matrix. For the up-spin component, the diagonal elements of the occupation matrix are set to 1. For the down-spin component, these elements are set to 0 except for the $3z^2 - r^2$ orbital.
In the former, the d-band with the $3z^2 - r^2$ character appears above the Fermi level. In the latter, this band occurs below the Fermi level, which indicates that a band gap is opened.

5.3.2 Hybrid functionals

5.3.2.1 Overview

The exact exchange energy is given by

$$E_{\mathrm{x}}^{\mathrm{exact}} = -\frac{1}{2}\sum_{\sigma}\sum_{n,m}^{\mathrm{occ}}\int d\,\boldsymbol{r}_1\int d\,\boldsymbol{r}_2\,\frac{\psi_{n\sigma}^{*}(\boldsymbol{r}_1)\psi_{m\sigma}(\boldsymbol{r}_1)\psi_{m\sigma}^{*}(\boldsymbol{r}_2)\psi_{n\sigma}(\boldsymbol{r}_2)}{|\boldsymbol{r}_1-\boldsymbol{r}_2|}$$

where $\psi_{n\sigma}(\boldsymbol{r})$ is a wavefunction of the n-th σ spin. Note that the summation over n,m only applies to occupied states. Here we define the hybrid exchange-correlation functional:

$$E_{\alpha}^{\mathrm{hybrid}} = \alpha E_{\mathrm{x}}^{\mathrm{exact}} + (1-\alpha)E_{\mathrm{x}}^{\mathrm{PBE}} + E_{\mathrm{c}}^{\mathrm{PBE}}$$

where $E_{\mathrm{x}}^{\mathrm{PBE}}$ represents the PBE exchange functional, and $E_{\mathrm{c}}^{\mathrm{PBE}}$ represents the PBE correlation functional. When $\alpha = \frac{1}{4}$, the $E_{\alpha}^{\mathrm{hybrid}}$ corresponds to the PBE0 functional.

5.3.2.2 Input parameters

To calculate the electronic states by the PBE0 functional, input parameters are set as follows:

```
accuracy{
    ksampling{
        method = gamma
        base_reduction_for_GAMMA = OFF
        base_symmetrization_for_GAMMA = OFF
    }
    xctype = ggapbe
    hybrid_functional{
        sw_hybrid_functional = ON
        alpha = 0.25
    }
}
```

In addition, wavefunctions and the charge density calculated by the PBE functional are given as initial guesses for the PBE0 functional calculation:

```
accuracy{
    initial_wavefunctions = file
    initial_charge_density = file
}
```

Make sure that the wavefunction file (zaj.data) and charge-density file (nfchgt.data) obtained by the PBE calculation are copied into the work directory. Note that for the hybrid functional, only the MSD method can be used as the wavefunction solver, and only norm-conserving pseudopotentials can be used as pseudopotentials.

Hartree–Fock calculations can also be performed by the following input.

```
accuracy{
    hybrid_functional{
        sw_hybrid_functional = ON
        sw_exchage_only = ON
        alpha = 1.00
    }
```

```
}
```

However, convergence of a Hatree–Fock calculation is significantly slower than that for a PBE0 calculation.

5.3.2.3 Examples: a hydrogen molecule

Sample input files of PBE, PBE0, and Hartree–Fock calculations of a hydrogen molecule are in the directory **samples/hybrid/H2**. By executing **go_h2.sh**, these calculations are executed in order. The results of these calculations and reference data obtained by Gaussian03 are compared in Figure 5.10.

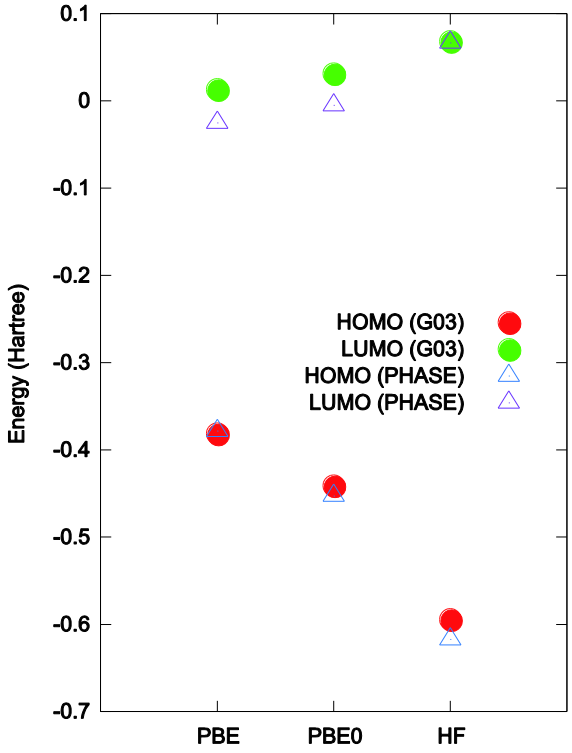


Figure 5.10 Energy levels of HOMO and LUMO of the hydrogen molecule, calculated by PBE functional, PBE0 functional, and the Hartree–Fock method. The results obtained from PHASE and Gaussian03 are compared.

5.3.2.4 Examples: a water molecule

Sample input files for PBE and PBE0 calculations for a water molecule are in the directory **samples/hybrid/H2O**. By executing **go_h2o.sh**, these calculations are executed in order. The results of these calculations and reference data obtained by Gaussian03 are compared in Figure 5.11.

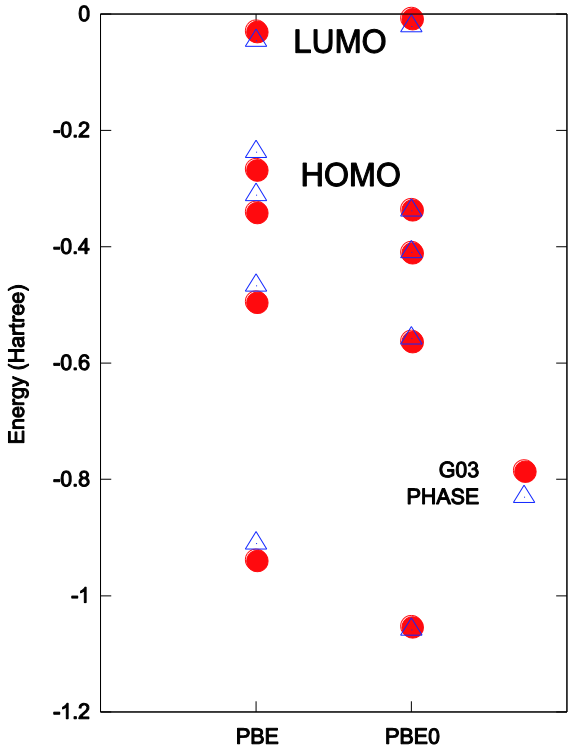


Figure 5.11 Energy levels of HOMO and LUMO of the water molecule calculated by the PBE functional and PBE0 functional.
The results obtained from PHASE and Gaussian03 are compared.

5.3.3 Non-local correlation term (van der Waals interaction)

5.3.3.1 Introduction for the van der Waals interaction

PHASE can calculate total energies and electronic states, including the van der Waals (vdW) interaction. In this section, the function used for the vdW term is explained. The vdW interaction is calculated by a nonempirical method. It is based on the van der Waals density functional (vdW-DF) given by Dion *et al.* [*1] It is widely known that the generalized gradient approximation (GGA) fails to reproduce the vdW interaction. Therefore, GGA cannot be used for systems in which the vdW interaction makes a large contribution, such as for interlayer interactions of stacked graphene sheets. The function described in this section avoids this defect. It can provide total energies and electronic states more accurately than GGA. The function contains no experimental parameters; thus, it is appropriate for any type of system.

This function is implemented via two Fortran 90 programs, vdW.F90 and vc_nl.F90._One program, vdW.F90, is used for "1-shot calculations (post-calculations)" to determine total energies, including the vdW interaction. The other, vc_nl.F90, is self-consistently implemented into the main program "PHASE." This program computes the vdW potential and directly implements it into the Kohn–Sham equation; hence, electronic states will be calculated with the vdW interaction included

5.3.3.2 Total energy (1-shot calculation)

15. Basic formula

The program vdW.F90 calculates the nonlocal correlation term $E_{\mathrm{c}}^{\mathrm{nl}}$ (i.e., the vdW term) and the local correlation term $E_{\mathrm{c}}^{\mathrm{LDA}}$. The total exchange-correlation term, including the vdW interaction, is obtained by adding the local and nonlocal terms to the GGA exchange term. Thus, the total exchange-correlation energy is written as

$$E_{\mathrm{xc}} = E_{\mathrm{x}}^{\mathrm{GGA}} + E_{\mathrm{c}}^{\mathrm{LDA}} + E_{\mathrm{c}}^{\mathrm{nl}} \tag{1}$$

The third term on the right hand side of Eq. (1) is the most difficult to calculate. This is what we call the van der Waals interaction; to calculate this term, we use the vdW-DF given by Dion *et al.* [*1]. They write $E_{\mathrm{c}}^{\mathrm{nl}}$ as

$$E_{\mathrm{c}}^{\mathrm{nl}} = \frac{1}{2}\int d\,\boldsymbol{r}_i \boldsymbol{r}_k \rho(\boldsymbol{r}_i)\phi(\boldsymbol{r}_i,\boldsymbol{r}_k)\rho(\boldsymbol{r}_k) \tag{2}$$

Equation (2) contains two spatial variables: $\boldsymbol{r}_i$and$\boldsymbol{r}_k$. This means that Eq. (2) considers nonlocal interactions between electron densities at points $\boldsymbol{r}_i$ and $\boldsymbol{r}_k$. This is the main difference between Eq. (2) and the formula used in GGA and LDA. The function containing these two variables, $\phi(\boldsymbol{r}_i,\boldsymbol{r}_k)$, is given by

$$\phi(\boldsymbol{r}_i,\boldsymbol{r}_k) = \frac{2}{\pi^2}\int_0^\infty d\,adba^2b^2WT \tag{3}$$

Here

$$\begin{aligned} W(a,b) = \frac{2}{a^3b^3}[&(3-a^2)b\cos\ b\sin\ a + (3-b^3)a\cos\ a\sin\ b \\ &+(a^2-b^2-3)\sin\ a\sin\ b - 3ab\cos\ a\cos\ b] \end{aligned} \tag{4}$$

and

$$\begin{aligned} T[x_i(a),x_i(b),x_k(a),x_k(b)] = \frac{1}{2}&\left[\frac{1}{x_i(a)+x_i(b)} + \frac{1}{x_k(a)+x_k(b)}\right] \\ &\times\left[\frac{1}{\big(x_i(a)+x_k(a)\big)\big(x_i(b)+x_k(b)\big)}\right. \\ &\left.+\frac{1}{\big(x_i(a)+x_k(b)\big)\big(x_i(b)+x_k(a)\big)}\right] \end{aligned} \tag{5}$$

The remaining variables are given by

$$x_j(a) = \frac{a^2}{2} \times \frac{1}{1 - exp\left(-\frac{4\pi a^2}{9d_j^2}\right)}, \tag{6}$$

$$d_j = |\boldsymbol{r}_i - \boldsymbol{r}_k| q_0(\boldsymbol{r}_j), \tag{7}$$

$$q_0(\boldsymbol{r}_j) = -\frac{4\pi}{3}\epsilon_{\mathrm{xc}}^{\mathrm{LDA}}\rho(\boldsymbol{r}_j) - \frac{Z_{ab}}{9}\left\{\frac{\nabla\rho(\boldsymbol{r}_j)}{2k_F(\boldsymbol{r}_j)\rho(\boldsymbol{r}_j)}\right\}^2 k_F(bfr_j), \tag{8}$$

$$k_F^3(\boldsymbol{r}_j) = 3\pi^2\rho(\boldsymbol{r}_j) \quad (j = i \text{ or } k) \tag{9}$$

The coefficient $Z_{ab} = -0.8491$ is determined by a first-principle calculation, and it does not change with a change in system. From these equations, we can see that the electron density $\rho(\boldsymbol{r})$ is the only input data to the functional $\phi(\boldsymbol{r}_i, \boldsymbol{r}_k)$. The quantity $\epsilon_{\mathrm{xc}}^{\mathrm{LDA}}$ in (8) is the exchange-correlation energy density in LDA [*2]. These formulas are based on the plasmon-pole model, and because of this, the vdW interaction can be obtained with a relatively low computational cost.

16. Algorism

The total energy, including the vdW interaction, is calculated using output files obtained from a GGA calculation implemented through PHASE. However, to avoid double counting, the GGA correlation term must be excluded from the original GGA. Thus, we represent the total energy obtained by this GGA-exchange-only calculation as $E_{\mathrm{total}}^{\mathrm{GGAx}}$. The correlation term, which will be calculated by vdW.F90, consists of two parts: a "local" part and a "nonlocal" part. These are determined in a 1-shot (post-) calculation by using the charge density file "nfchr.cube" generated by PHASE. The total energy including the vdW interaction $E_{\mathrm{total}}^{\mathrm{vdW-DF}}$ is obtained by adding these energy terms:

$$E_{\mathrm{total}}^{\mathrm{vdW-DF}} = E_{\mathrm{total}}^{\mathrm{GGAx}} + E_{\mathrm{c}}^{\mathrm{LDA}} + E_{\mathrm{c}}^{\mathrm{nl}} \tag{10}$$

Here $E_{\mathrm{c}}^{\mathrm{LDA}}$ is the local correlation term, and $E_{\mathrm{c}}^{\mathrm{nl}}$ is the nonlocal correlation term. Figure 5.12 shows the calculation flow to obtain $E_{\mathrm{total}}^{\mathrm{vdW-DF}}$ from the 1-shot program vdW.F90. The green box represents the vdW routine, while the blue box represents the GGA (exchange-only) routine. Before running vdW.F90, we need to run PHASE to obtain two output files: "nfchr.cube" for the electron density $\rho^{GGAx}(\boldsymbol{r})$ and "nfefn.data" for $E_{\mathrm{total}}^{\mathrm{GGAx}}$.

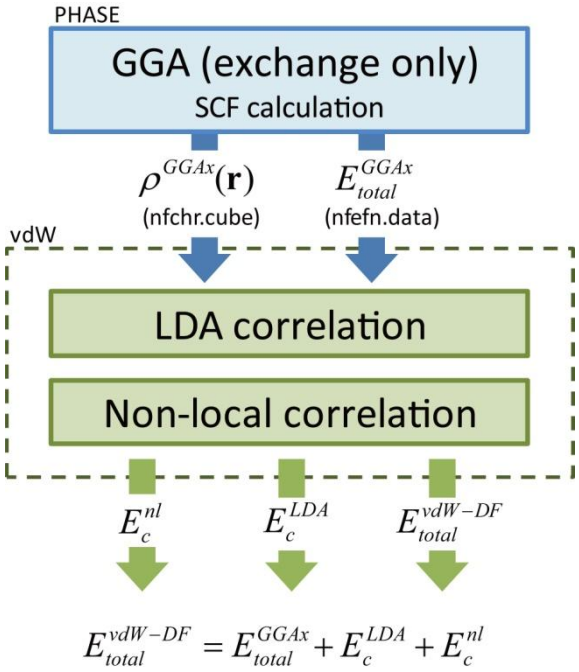


Figure 5.12 Calculation flow for vdW.F90

17. Execution of 1-shot calculations

- Running PHASE in advance

The program vdW.F90 is designed to perform a 1-shot (post-) calculation. This program needs two input files: "nfchr.cube" and "nfefn.data'. (See Fig.*1.) Here "nfchr.cube" contains the charge density and "nfefn.data" contains the total energy $E_{\mathrm{total}}^{\mathrm{GGAx}}$; both files are obtained by running PHASE.
To obtain "nfchr.cube" from PHASE, we need to set parameters in "nfinp.data". Here are representative examples for "file_names.data" and "nfinp.data."

file_names.data (PHASE) :

```
F_ENF   =  './nfefn.data'
F_CHR   =  './nfchr.cube'
```

nfinp.data (PHASE)

```
accuracy{
xctype = ggapbex
}

postprocessing{
        charge{
                sw_charge_rspace        =       ON
                 filetype                       =       cube
        }
}
```

Although a "ggapbe"-type pseudopotential will be used in PHASE, the correlation term must be excluded to avoid double counting. To exclude this term, a new word "ggapbex" is added as a possible value for "xctype" in PHASE. By setting "xctype = ggapbex," only the exchange term in ggapbe will be calculated. The same pseudopotential files are available for "ggapbex" with "ggapbe."

- Compiling vdW.F90

The program vdW.F90 is parallelized with OpenMP and is compiled by a Fortran 90 compiler. Add the "-openmp" option when compiling for parallel calculations,

```
$   ifort -openmp -o vdW vdW.F90
```

- Executing 'vdW'

Put the two input files "nfchr.cube" and "nfefn.data" into the same directory with the execution file "vdW" and do not change their names. These two files will be read automatically; hence, no additional input is needed for "vdW."

- Writing output from "vdW"

Output data from "vdW" will be written in the format shown below. The units are all in Hartree, the same as in PHASE.

Output example:
These results were calculated by a serial calculation using the output files "nfchr.cube" and "nfefn.data" in phase/samples/vdW/input_scf_Si.data.

```
E_total(GGA exchange) =       -7.5363221703000

Ec(LDA) =       -0.5429739815997
Ec(nl) =        0.0203272639208
Ec (= Ec(LDA) + Ec(nl) )      =       -0.5226467176789

E_total(vdW-DF)               =       -8.0589688879789

Given in Hartree atomic units

# Calculation time    0 : 11 : 33.7280
```

Meaning of each variable:

E_total(GGA exchange)	Total energy of GGA (exchange only)
Ec(LDA)	Local correlation term from LDA
Ec(nl)	Non-local correlation term
Ec (= Ec(LDA) + Ec(nl))	Total correlation term
E_total(vdW-DF)	Total energy including the vdW-DF
Calculation time	Hours : minutes : seconds

Here "E_total(vdW-DF)" is the main objective in "vdW."

5.3.3.3 Example: Silicone Diamond

These results can be tested by using files in "phase/samples/vdW/." First, execute PHASE with "file_names.data" and "input_scf_Si.data"; two output files "nfchr.cube" and "nfefn.data" will be automatically created. Next, compile the vdW.F90 program prepared in the same directory and execute it. Thus, use the following commands:

```
  $  cd phase/samples/vdW/           (Change directory to samples/vdW/)
  $  ../../bin/phase          (Execute PHASE and do the GGA (exchange only) calculation)
  $  ifort -openmp -o vdW vdW.F90  (Compile vdW.F90 and prepare the execution file 'vdW')
  $  ./vdW                          (Execute 'vdW' at the same directory)
```

Outputs

```
E_total(GGA exchange) = -7.5363221703000

Ec(LDA) = -0.5429739815997
Ec(nl) = 0.0203272639208
Ec (= Ec(LDA) + Ec(nl) ) = -0.5226467176789

E_total(vdW-DF) = -8.0589688879789

Given in Hartree atomic units

# Calculation time 0 : 11 : 33.7280
```

5.3.3.4 Electron state calculation (self-consistent field calculation)

18. Basic formula

To calculate electronic states, considering the vdW interaction, the vdW potential term must be directly implemented into the Kohn–Sham equation that is used in self-consistent field (SCF) calculations. The functional derivatives with respect to the charge density $\rho(\boldsymbol{r})$ of the energy terms $E_{\mathrm{c}}^{\mathrm{nl}}$ and $E_{\mathrm{c}}^{\mathrm{LDA}}$ (i.e., the potential terms) will be calculated by

$$v_{\mathrm{c}}^{\mathrm{nl}} = \frac{\delta E_{\mathrm{c}}^{\mathrm{nl}}}{\delta \rho(\boldsymbol{r})} \qquad (10)$$
$$v_{\mathrm{c}}^{\mathrm{LDA}} = \frac{\delta E_{\mathrm{c}}^{\mathrm{LDA}}}{\delta \rho(\boldsymbol{r})}$$

Then, self-consistent implementation will be done by adding each potential term,

$$v_{\mathrm{xc}} = v_{\mathrm{x}}^{\mathrm{GGA}} + v_{\mathrm{c}}^{LDA} + v_{c}^{\mathrm{nl}} \qquad (11)$$

Equation (3) shows that $E_{\mathrm{c}}^{\mathrm{nl}}$ and $E_{\mathrm{c}}^{\mathrm{LDA}}$ include $\rho(\boldsymbol{r})$ in a relatively simple manner; thus, those derivations can be obtained analytically.

The total energies should be calculated with SCF; however, the difference between the 1-shot result and the SCF result is confirmed to be negligible. [3] Therefore, for calculating only total energies, we recommend using only the 1-shot calculation since the CPU cost will be much lesser than when using SCF calculations.

19. Execution of SCF calculations

This routine self-consistently solves the Kohn–Sham equation by adding the local $v_{\mathrm{c}}^{\mathrm{LDA}}$ and nonlocal correlation potentials $v_{\mathrm{c}}^{\mathrm{nl}}$ to the GGA exchange potential $v_{\mathrm{x}}^{\mathrm{GGA}}$. Calculation of electronic states, including the vdW interaction, are performed only by running PHASE similar to normal GGA or LDA calculations. Note that a unit cell of a system needs to be cuboid in this version. There is no such limitation in the 1-shot program; e.g., rhombic unit cells can be used in vdW.F90.

To implement the vdW term in the self-consistent calculations, "nfinp.data" needs to be written as follows.

nfinp.data (PHASE) :

```
accuracy{
  xctype = vdwdf
}
```

The program vc_nl.F90, which is used for the self-consistent implementation of the vdW term, is located in directory src_phase/. It is parallelized by OpenMP and is automatically compiled by executing the command "make" for compiling a normal PHASE.

5.3.3.5 References

[1] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist: Phys. Rev. Lett. **92** (2004) 246401: Erratum, ibid, **95** (2005) 109902.

[2]O. Gunnarsson and B. I. Lundqvist: Phys. Rev. B **13** (1976) 4274.

[3] L. X. Benedict, N. G. Chopra, M. L. Cohen, A. Zettl, S. G. Louie, and V. H. Crespi: Chem. Phys. Lett. **286** (1998) 490.

[4] Y. Baskin and L. Mayer: Phys. Rev. **100**, (1955) 544.

[5] H. Rydberg, M. Dion, N. Jacobson, E. Schröder, P. Hyldgaard, S. I. Simak, D. C. Langreth, and B. I. Lundqvist: Phys. Rev. Lett. **91** (2003) 126402.

[6] T. Thonhauser, Valentino R. Cooper, Shen Li, Aaron Puzder, Per Hyldgaard, and David C. Langreth: Phys. Rev. B **76**, 125112 (2007).

5.3.4 Van der Waals corrected DFT

5.3.4.1 Overview

- Williams method (R.W. Williams, et al., Chemical Physics 327 (2006) 54–62)

$$E_{vdw} = \sum_{ij} \frac{C_6^{ij}}{R_{ij}^6} f(R_{ij})$$

$$f(R) = \left(1 - \exp\left[-d \left(\frac{R_{ij}}{R_0^{ij}} \right)^7 \right] \right)^4$$

$$C_6^{ij} = -S_C \times \frac{2 C_6^i C_6^j p_i p_j}{p_i^2 C_6^i + p_j^2 C_6^j}, \quad R_0^{ij} = S_R \times \frac{\left(R_0^{ii}\right)^3 + \left(R_0^{jj}\right)^3}{\left(R_0^{ii}\right)^2 + \left(R_0^{jj}\right)^2}, \quad R_0^{ii} = 2 \times R_0^i$$

Parameters:
vdw radius 20.0 bohr

scaling factor S_C 0.8095 (PHASE), S_R 0.80 reference PBE S_C 0.85 S_R 0.80

damping factor d 3.0

	polarizabilities A^3	vde coef C6 Hartree*bohr^6	vdw radius		polarizabilities A^3	vde coef C6 Hartree*bohr^6	vdw radius
H	0.387	2.831179918	1.17	NTE	0.964	20.89758657	1.50
F	0.296	3.94987377		NTR2	1.030	23.08003267	1.50
Cl	2.315	3.94987377		NPI2	1.090	25.12582491	1.50
Br	3.013	128.2756865		NDI	0.956	20.63799109	1.50
I	5.415	309.0603852		OTE	0.637	11.86370812	1.40
CTE	1.061	22.67403316	1.70	OTR4	0.569	10.01566303	1.40
CTR	1.352	32.61525204	1.70	OPI2	0.274	3.346856941	1.40
CAR	1.352	49.790/Sc	1.70	STE	3.000	121.2531939	1.80
CBR	1.896	54.16430826	1.70	STR4	3.729	168.0350502	1.80
CDI	1.283	30.15058105	1.70	SPI2	2.700	103.5277919	1.80
				PTE	1.538	42.11289383	1.80

- Grimme method (DFT-D2) (S. Grimme, J. Comp. Chem. 27 (2006) 1787)

$$E_{disp} = -s_6 \sum_{ij} \frac{C_6^{ij}}{R_{ij}^6} f(R_{ij})$$

$$f(R) = \frac{1}{1 + \exp\left[-d \left(\frac{R_{ij}}{R_0^{ij}} - 1 \right) \right]}$$

$$C_6^{ij} = \sqrt{C_6^i C_6^j}, \quad R_0^{ij} = R_0^i + R_0^j$$

Parameters:
vdw radius 30.0A

scaling factor s_6 0.75, damping factor d 20.0

	C6 Jnm⁶/mol	R0 A		C6 Jnm⁶/mol	R0 A
H	0.14	1.001	K	10.80	1.485
He	0.08	1.012	Ca	10.80	1.474
Li	1.61	0.825	Sc-Zn	10.80	1.562
Be	1.61	1.408	Ga	16.99	1.650
B	3.13	1.485	Ge	17.10	1.727
C	1.75	1.452	As	16.37	1.760
N	1.23	1.397	Se	12.64	1.771
O	0.70	1.342	Br	12.47	1.749
F	0.75	1.287	Kr	12.01	1.727
Ne	0.63	1.243	Rb	24.67	1.628
Na	5.71	1.144	Sr	24.67	1.606
Mg	5.71	1.364	Y-Cd	24.67	1.639
Al	10.79	1.716	In	37.32	1.672
Si	9.23	1.716	Sn	38.71	1.804
P	7.84	1.705	Sb	38.44	1.881
S	5.57	1.683	Te	31.74	1.892
Cl	5.07	1.639	I	31.50	1.892
Ar	4.61	1.595	Xe	29.99	1.881

1 J/mol = 3.8088e-7 hartree, 1 bohr = 0.5291772480 A

5.3.4.2 Input parameters

A list of tag keyword related to the vdW correction

1st level block	2nd, 3rd level block	Tag keyword	Description
Control	sw_vdw_correction *		
Accuracy	vdw_method	williams grimme or dft-d2	default
	vdw_radius		20 Bohr 30 A (Grimme DFT-D2)
	vdw_scaling_factor		0.805 (Williams) 0.75 (Grimme DFT-D2)
	vdw_scaling_factor_r		0.8 (Williams)
	vdw_damping_factor		3.0 (Williams) 20.0 (Grimme DFT-D2)
Structure	atom_list		
	atoms *	a type of vdW correction is specified by #tag vdw	
	vdw_list	parameters for each element for vdW correction are defined	Williams #tag type c6 r0 p Grimme #tag type c6 r0

* required for the vdW correction

Parameters for the vdW correction for each element

Parameters in the William's method and Grimme method (DFT-D2) for each element are internally defined in PHASE and used as default values. The **type** attribute in the **vdw_list** must correspond to the type of vdW in the **atom_list**.

Williams method

```
      vdw_list{
           #tag type  c6  r0  p
                H    2.831179918  1.17  0.387
                CTE   22.67403316  1.70  1.061
      }
```

Grimme method (DFT-D2)

```
      vdw_list{
           #tag type  c6  r0
                H    0.14  1.001
                C    1.75  1.452
      }
```

Example of input parameters

Input data for vdW corrections are illustrated below.

Methane Dimer by the Williams method

```
Control{
       sw_vdw_correction = ON
}
accuracy{
       vdw_method = williams
       vdw_radius = 20.0
       vdw_scaling_factor = 0.8095
       vdw_scaling_factor_r = 0.8
       vdw_damping_factor = 3.0
}
structure{
       atom_list{
           coordinate_system = cartesian ! {cartesian|internal}
           atoms{
              #units angstrom
              #default mobile=on
              #tag  element  rx       ry       rz   vdw
              C      0      0      0      CTE
              H      0      1.093   0      H
              H      1.030490282    -0.364333333    0      H
              H      -0.515245141    -0.364333333    0.892430763    H
              H      -0.515245141    -0.364333333    -0.892430763    H
              C      0      -3.7    0      CTE
              H      0      -4.793  0      H
              H      -1.030490282    -3.335666667    0      H
              H      0.515245141    -3.335666667    -0.892430763    H
              H      0.515245141    -3.335666667    0.892430763    H
           }
       }
      vdw_list{
           #tag type  c6  r0  p
                H    2.831179918  1.17  0.387
                CTE   22.67403316  1.70  1.061
      }
}
```

Methane Dimer by the Grimme method (DFT-D2)

```
Control{
```

```
       sw_vdw_correction = ON
}
accuracy{
       vdw_method = grimme
       vdw_radius = 30.0
       vdw_scaling_factor = 0.75
       vdw_damping_factor = 20.0
}
structure{
       atom_list{
           coordinate_system = cartesian ! {cartesian|internal}
           atoms{
              #units angstrom
              #default mobile=on
              #tag  element  rx       ry       rz   vdw
              C      0      0      0      C
              H      0      1.093   0      H
              H      1.030490282    -0.364333333    0      H
              H      -0.515245141   -0.364333333    0.892430763    H
              H      -0.515245141   -0.364333333    -0.892430763   H
              C      0      -3.7    0      C
              H      0      -4.793  0      H
              H      -1.030490282   -3.335666667    0      H
              H      0.515245141    -3.335666667    -0.892430763   H
              H      0.515245141    -3.335666667    0.892430763    H
           }
       }
      vdw_list{
           #tag type  c6  r0
                H    0.14  1.001
                C    1.75  1.452
      }
}
```

5.3.4.3 Calculation examples

- Water_Dimer (Williams, Grimme(DFT-D2))
- Methane_Dimer (Williams, Grimme(DFT-D2))
- Ethane_Dimer (Williams, Grimme(DFT-D2))
- ATstack (Williams)

5.4 Analysis of chemical reactions

5.4.1 The NEB method

5.4.1.1 Outline of the feature

The nudged elastic band (NEB) method and the climbing image (CI)–NEB method enable us to obtain the minimum energy path for a chemical reaction (or more generally, for any process having finite activation energy).

In reaction-path calculations based on the NEB and CI–NEB methods, we assume that the atomic configurations of the initial state ($\vec{R}_0$) and final state ($\vec{R}_N$) are known in advance. The atomic configurations and energies of the intermediate states ($\vec{R}_i,\ i = 2 \sim N-1$), hereafter referred to as either "images" or "replicas," are obtained by performing structural optimization under the constraint that adjacent images are coupled by hypothetical "springs." Here $\vec{R}_i$ denotes the atomic coordinates of the *i*-th image. Initial intermediate images can be arbitrarily generated, although they are usually built by a simple linear interpolation between the initial and final states.

- Ordinary NEB method

In the ordinary NEB method, forces acting on each image are calculated by

$$\vec{F}_i = \vec{F}_i^s|_{\parallel} - \nabla E(\vec{R}_i)|_{\perp}.$$

Here $\vec{F}_i^s|_{\parallel}$ is the component of the spring force that is parallel to the reaction path; it is calculated by

$$\vec{F}_i^s|_{\parallel} = k\left(|\vec{R}_{i+1} - \vec{R}_i| - |\vec{R}_i - \vec{R}_{i-1}|\right) \cdot \hat{\tau}\hat{\tau}.$$

Here *k* is the spring constant, and $\hat{\tau}$ is the unit vector along the reaction path, which is calculated from

$$\hat{\tau} = \frac{\vec{R}_i - \vec{R}_{i-1}}{|\vec{R}_i - \vec{R}_{i-1}|} + \frac{\vec{R}_{i+1} - \vec{R}_i}{|\vec{R}_{i+1} - \vec{R}_i|}.$$

$\nabla E(\vec{R}_i)|_{\perp}$ is the component of the atomic forces that is perpendicular to the reaction path; it is calculated by

$$\nabla E(\vec{R}_i)|_{\perp} = \nabla E(\vec{R}_i) - \nabla E(\vec{R}_i) \cdot \hat{\tau}\hat{\tau}.$$

- CI–NEB method

The CI–NEB method is a revision of the NEB method, in which the forces of the image with the highest energy (the image closest to the transition state) are modified. First, the reaction-path calculations are advanced to a certain extent by the ordinary NEB method. Then, the image with the highest energy is identified, and the forces acting on it are modified by

$$\vec{F}_{i,\max} = -\nabla E(\vec{R}_{i,\max}) + 2\nabla E(\vec{R}_{i,\max})$$

$$= \vec{F}_{i,\max}|_{\perp} - \vec{F}_{i,\max}|_{\parallel}$$

$$= \vec{F}_{i,\max}|_{\perp} - \vec{F}_{i,\max}|_{\parallel}$$

Using this formula, the highest-energy replica will "climb" the reaction path toward the transition state. When convergence is reached, the highest-energy replica will be exactly located at the transition state.

- Calculation method for the spring constants

When calculating the minimum energy path of a reaction, it is preferable to increase the accuracy of the states in the vicinity of the saddle point. Thus, it is preferable to increase the "density" of images near the saddle point and accurately calculate the tangent to the path. One way to do this is to strengthen the spring constant of images near the saddle point. To this end, the following formula for the spring constant has been suggested in the literature:

$$k = k_{\max} - \Delta\left(\frac{E_{max} - E_i}{E_{max} - E_{ref}}\right) \quad (E_i \geq E_{\mathrm{ref}}),$$

$$k = k_{\max} - \Delta k \qquad (E_i < E_{\mathrm{ref}}).$$

Here $k_{\max}$ is the maximum value of the spring constant, Δk is the difference between the maximum and minimum spring constants, E_i denotes the higher of the energies between the two images connected to the i-th spring, $E_{\max}$ is the highest energy among the images, and E_{ref} is the higher of the energies between the initial and final states.

5.4.1.2 Input parameters

20. Specification of the input parameter file

The tags related to the NEB method are tabulated below.

1st level block	2nd, 3rd level block	identifiers	description
Control			
		multiple_replica_mode	Set this switch to "ON" to operate PHASE in the NEB mode.
		multiple_replica_max_iteration	Specify the maximum number of NEB iterations.
multiple_replica			
	accuracy		
		dt	Specify the time step for NEB optimization.
		neb_time_integral	Specify the integration method (either quench or steepest_descent) for the NEB method. Defaults to steepest_descent.
		penalty_function	Specify whether to enable the penalty function. Defaults to NO.
		neb_convergence_condition	Specify the convergence condition for NEB optimization (further details will be given in the explanations below)
		neb_convergence_threshold	Specify the threshold for convergence.
	constraint		
		ci_neb	Switch to whichever specifies

			whether the CI–NEB should be enabled. Defaults to NO.
		sp_k_init	Specify the initial value of the spring constant.
		sp_k_min	Specify the minimum value of the spring constant.
		sp_k_max	Specify the maximum value of the spring constant.
		sp_k_variable	Switch to whichever specifies whether the spring constants should be variable. Defaults to NO.
	structure		
		number_of_replicas	Specify the number of replicas excluding the initial and final states.
	replica		Block to specify information regarding the replicas.
		endpoint_images	Specify the method (either "directin" or "file") used to specify the atomic coordinates of the initial and final states. When "directin" is specified, the coordinates are specified within the F_INP file. When "file" is specified, the coordinates are supplied from a separate file. Defaults to "directin."
	atom_list_end0		Block whichever specifies the atomic coordinates for the initial state.
	atom_list_end1		Block whichever specifies the atomic coordinates for the final state.

Now, we use concrete examples to illustrate the configuration of input parameters for the NEB method.

The following must be configured when performing NEB calculations.

- Enable the NEB method.
- Configure the convergence condition and threshold specific to the NEB method.
- Specify the atomic coordinates for the initial and final states.
- Specify the atomic coordinates of the intermediate images. (You can instruct PHASE to automatically build the intermediate images by a linear interpolation between the initial and final states.)

- Enable the NEB method

To instruct PHASE to perform NEB calculations, define the variable "multiple_replica_mode" under the "control" block and set its value to "on."

```
control{
  multiple_replica_mode = on
}
```

- Configure the convergence condition and threshold specific to the NEB method

The convergence condition is configured by the "neb_convergence_condition" variable under the "accuracy"

block under the "multiple_replica" block.

```
multiple_replica{
   accuracy{
     neb_convergence_condition = energy_e
   }
}
```

The value for the "neb_convergence_criteria" can be specified by either an integer or a string. The correspondence is as follows.

integer	string	description
1	energy_e	dE < threshold
2	phase_force	maximum force from PHASE < threshold
3	neb_force	maximum NEB force <threshold
4	force_at_transition_state	maximum force from PHASE for the highest-energy image < threshold
5	phase_force_normal	maximum of the PHASE force component perpendicular to the tangent of the reaction path < threshold

The threshold value is specified by the "neb_convergence_threshold" variable, definable under the same block as the "neb_convergence_criteria" variable. Note that the unit for this variable changes according to the value specified for the "neb_convergence_criteria" variable. Thus, the unit for this variable cannot be explicitly specified: the default units (atomic units) must be used.

- Specify the atomic coordinates for the initial and final states: direct specification

To directly specify atomic coordinates for the initial and final states in the F_INP file, set the "endpoint_images" variable to "directin" and define the "atom_list_end0" and "atom_list_end1" block. Here is an example.

```
multiple_replica{
    ....
    ....
    structure{
        ....
        ....
        endpoint_images = directin
        atom_list_end0{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx ry rz
            Si    0.000000000000      0.000000000000      0.000000000000
            Si    2.751721694800      2.751721694800      0.000000000000
            ....
            ....
            }
        }
        atom_list_end1{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx  ry  rz
            Si    0.000000000000      0.000000000000      0.000000000000
            Si    2.751721694800      2.751721694800      0.000000000000
            ....
            ....
            }
```

```
        }
        ....
        ....
    }
    ....
    ....
```

The atomic coordinates for the initial and final states are specified under the "atom_list_end0" block and the "atom_list_end1" block, respectively. The format of the specification is the same as that for the usual atomic coordinates, i.e., the specification under the "atom_list" block under the "structure" block.

- Specify the atomic coordinates for the initial and final states: specification from external files.

To specify atomic coordinates for the initial and final states from external files, set the value of the "endpoint_images" variable to "file."

```
multiple_replica{
    ...
    ...
    structure{
        endpoint_images = file
    }
    ...
    ...
}
```

The names of the external files are specified in the "file_names.data" file as usual. The corresponding file pointers are F_IMAGE (–1) and F_IMAGE (0). Here is an example.

```
&fnames
...
...
/
&nebfiles
F_IMAGE(0)  = './endpoint0.data'
F_IMAGE(-1) = './endpoint1.data'
...
...
/
```

Note that the F_IMAGE (0) and F_IMAGE (–1) pointers can only be used under the "&nebfiles" section.

The file formats of the F_IMAGE (0) and F_IMAGE (–1) files are as follows.

```
coordinate_system=cartesian

#units angstrom

Si      0.000000000000      0.000000000000      0.000000000000
Si      2.751721694800      2.751721694800      0.000000000000
Si      1.375860847400      1.375860847400      1.375860847400
Si      4.127582542200      4.127582542200      1.375860847400
Si      0.000000000000      2.751721694800      2.751721694800
Si      2.751721694800      0.000000000000      2.751721694800
Si      1.375860847400      4.127582542200      4.127582542200
Si      4.127582542200      1.375860847400      4.127582542200
Si      0.000000000000      0.000000000000      5.503443389600
```

```
Si      2.751721694800      2.751721694800      5.503443389600
Si      1.375860847400      1.375860847400      6.879304237000
H       1.644706293661      1.095414892118     11.000000000000
H       1.095414929519      1.644706317263     11.000000000000
```

- Specify the intermediate images from a linear interpolation

The atomic coordinates of intermediate images can be specified by a linear interpolation of the initial and final states. This is done as follows.

```
multiple_replica{
   structure{
      number_of_replicas = 6
      replicas{
      #tag replica_number  howtogive_coordinates   end0  end1
                1          proportional            0     -1 ! 0: end0, -1:end1
                2          proportional            0     -1
                3          proportional            0     -1
                4          proportional            0     -1
                5          proportional            0     -1
                6          proportional            0     -1
      }
   }
}
```

- Specify the intermediate images from external files

To specify intermediate images from external files, the value of the "howtogive_coordinates" variable is set to "file." Here is an example.

```
multiple_replica{
    ...
    ...
    structure{
        number_of_replicas = 3
        replicas{
                #tag replica_number  howtogive_coordinates   end0  end1
                1            file              0      -1 ! 0: end0, -1:end1
                2            file              0      -1
                3            file              0      -1
        }
    }
}
```

The names of the external files are specified in the "file_names.data" file. The corresponding file pointers are F_IMAGE(N), where N is the ID for the intermediate image. Note that, only for initial and final images, F_IMAGE(N) file pointers must be defined under the "&nebfiles" section.

```
&fnames
...
...
/
&nebfiles
F_IMAGE(0)  = './endpoint0.data'
F_IMAGE(-1) = './endpoint1.data'
F_IMAGE(1)  = './image1.data'
F_IMAGE(2)  = './image2.data'
F_IMAGE(3)  = './image3.data'

/
```

The file format for intermediate images is the same as that for the initial and final images described above.

Finally, we present an example of the full input parameter file.

```
Control{
    condition = initial   ! {initial|continuation|automatic}
    cpumax = 1 day ! {sec|min|hour|day}
    max_iteration = 10000000
    multiple_replica_mode = ON
    multiple_replica_max_iteration = 2000
}
accuracy{
    cutoff_wf =  10.00  rydberg
    cutoff_cd =  40.00 rydberg
    num_bands =  28
    ksampling{
        method = monk  ! {mesh|file|directin|gamma}
        mesh{  nx = 2, ny =  2, nz =  1  }
    }
    smearing{
        method = parabolic ! {parabolic|tetrahedral}
        width  = 0.001 hartree
    }
    xctype = ggapbe
    scf_convergence{
        delta_total_energy = 0.5e-7   hartree
        succession   = 2   !default value = 3
    }
    initial_wavefunctions = matrix_diagon  !{random_numbers|matrix_diagion}
    matrix_diagon{
        cutoff_wf =  3.00  hartree
    }
}
structure{
    unit_cell_type = primitive
    unit_cell{
         a_vector =  10.400      0.000      0.000
         b_vector =   0.000     10.400      0.000
         c_vector =   0.000      0.000     30.200
    }
    symmetry{
        sw_inversion = off
    }
    atom_list{
        coordinate_system = cartesian ! {cartesian|internal}
        atoms{
        #units angstrom
                #tag  element  rx  ry  rz   mobile
    Si    0.000000000000      0.000000000000      0.000000000000    0
    Si    2.751721694800      2.751721694800      0.000000000000    0
    Si    1.375860847400      1.375860847400      1.375860847400    0
    Si    4.127582542200      4.127582542200      1.375860847400    0
    Si    0.000000000000      2.751721694800      2.751721694800    0
    Si    2.751721694800      0.000000000000      2.751721694800    0
    Si    1.375860847400      4.127582542200      4.127582542200    0
    Si    4.127582542200      1.375860847400      4.127582542200    0
    Si    0.000000000000      0.000000000000      5.503443389600    0
    Si    2.751721694800      2.751721694800      5.503443389600    0
    Si    1.375860847400      1.375860847400      6.879304237000    0
```

```
    H    1.644706293661      1.095414892118     11.000000000000    1
    H    1.095414929519      1.644706317263     11.000000000000    1
         }
    }
    element_list{
            #tag element  atomicnumber  mass  zeta   dev
            #units atomic_mass
            Si        14     28.085
            H          1     1.008
    }
}
multiple_replica{
    method = nudged_elastic_band_method
    accuracy{
        dt = 40 au_time
        neb_time_integral = quench
        penalty_function = off
        neb_convergence_condition = 3
        neb_convergence_threshold = 5.0e-04
    }
    constraint{
        ci_neb = OFF
        sp_k_init = 0.03
        sp_k_min = 0.03
        sp_k_max = 0.03
        sp_k_variable = OFF
    }
    structure{
        number_of_replicas = 6
        replicas{
            #tag replica_number  howtogive_coordinates   end0  end1
            1            proportional             0      -1 ! 0: end0, -1:end1
            2            proportional             0      -1
            3            proportional             0      -1
            4            proportional             0      -1
            5            proportional             0      -1
            6            proportional             0      -1
        }
        endpoint_images = directin ! {no or nothing | file | directin}
        howtogive_coordinates = from_endpoint_images
        atom_list_end0{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx ry rz
            Si    0.000000000000      0.000000000000      0.000000000000
            Si    2.751721694800      2.751721694800      0.000000000000
            Si    1.375860847400      1.375860847400      1.375860847400
            Si    4.127582542200      4.127582542200      1.375860847400
            Si    0.000000000000      2.751721694800      2.751721694800
            Si    2.751721694800      0.000000000000      2.751721694800
            Si    1.375860847400      4.127582542200      4.127582542200
            Si    4.127582542200      1.375860847400      4.127582542200
            Si    0.000000000000      0.000000000000      5.503443389600
            Si    2.751721694800      2.751721694800      5.503443389600
            Si    1.375860847400      1.375860847400      6.879304237000
            H    1.644706293661      1.095414892118     11.000000000000
            H    1.095414929519      1.644706317263     11.000000000000
            }
        }
```

```
        atom_list_end1{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx  ry  rz
            Si    0.000000000000      0.000000000000      0.000000000000
            Si    2.751721694800      2.751721694800      0.000000000000
            Si    1.375860847400      1.375860847400      1.375860847400
            Si    4.127582542200      4.127582542200      1.375860847400
            Si    0.000000000000      2.751721694800      2.751721694800
            Si    2.751721694800      0.000000000000      2.751721694800
            Si    1.375860847400      4.127582542200      4.127582542200
            Si    4.127582542200      1.375860847400      4.127582542200
            Si    0.000000000000      0.000000000000      5.503443389600
            Si    2.751721694800      2.751721694800      5.503443389600
            Si    1.375860847400      1.375860847400      6.879304237000
            H    2.22686927     0.48813212     7.65400988
            H    0.48813224     2.22686933     7.65400957
            }
        }
    }
}
wavefunction_solver{
    solvers{
        #tag   sol    till_n  dts  dte  itr  var    prec cmix submat
          lmMSD   -1      0.2  0.2  1  linear   on    1   on
    }

}
charge_mixing{
    mixing_methods{
    #tag no   method   rmxs   rmxe   itr  var     prec istr  nbmix  update
          1  broyden2  0.10   0.10   1  linear  on  1  0   RENEW
    }
}
printoutlevel{
    base=1
}
```

21. specification of the files related to the NEB method

Files related to the NEB method is specified in the 'file_names.data' file as usual. Here is an example.

```
&fnames
F_INP='./nfinp.data'
F_POT(1)='./Si_ggapbe_nc_01.pp'
...
...
/
&nebfiles
F_IMAGE(0)  = './endpoint0.data'
F_IMAGE(-1) = './endpoint1.data'
F_NEB_OUT   = './output_neb'
F_NEB_ENF   = './nfnebenf.data'
F_NEB_DYNM  = './nfnebdynm.data'
/
```

Note that the "&nebfiles" section is used to specify NEB-related files. In Table 5.2, the file pointers usable

under the "&nebfiles" section are tabulated.

Table 5.2 NEB-related files

file pointer	unit number	default value	notes
F_IMAGE(-1:99)	201	./endpoint0.data (F_IMAGE(0)) ./endpoint1.data (F_IMAGE(1))	Atomic coordinates of the images.
F_NEB_STOP	202	./nfnebstop.data	File used to terminate NEB calculations.
F_NEB_OUT	203	./output_neb	Log file for the NEB calculations.
F_NEB_CNTN	204	./neb_continue.data	Restart file for the NEB calculations.
F_NEB_ENF	205	./nfnebenf.data	File that records energy and forces specific to the NEB method.
F_NEB_DYNM	206	./nfnebdynm.data	Output the history of the atomic coordinates

5.4.1.3 Execution

NEB calculations are typically executed by the following command.

```
% mpirun -n NP phase ne=NE nk=NK nr=NR
```

Note the presence of the "nr=NR" argument. This argument specifies the number of replicas to be handled in parallel. The "ne=NE" and "nk=NK" arguments specify band parallelization and $\boldsymbol{k}$ point parallelization as usual. Note that the relation `NP = NR x NE x NK` must be met.

5.4.1.4 Output of the results

When NEB calculations are executed, several extra files will be obtained, along with those obtained from usual PHASE calculations. First, the log file (output000 file) and the restart files (such as the "continue.data" file) will be obtained for all images. To identify each file, the string "_rxxx" will be appended to the original file name, where xxx is the ID for the replica. Further, the following files specific to NEB calculations are obtained.

- output_neb_pxxx

This is the log file for the NEB calculations (xxx will be replaced by the MPI process number).

- nfnebenf.data

File that records energy and force specific to the NEB method. The format of this file is as follows.

```
 #step  image  image_distance  energy   force_org  force_neb  force_normal
    1     1   0.0000000000E+00  -0.4399458479E+02   0.1112676571E-01   0.1112676571E-01   0.0000000000E+00
    1     2   0.1323772380E+01  -0.4397221867E+02   0.5212041989E-01   0.4899393390E-01   0.4899393390E-01
    1     3   0.2640972887E+01  -0.4393533860E+02   0.5368141337E-01   0.5023308254E-01   0.5023308254E-01
    1     4   0.3958252743E+01  -0.4389613534E+02   0.4830449879E-01   0.4474348402E-01   0.4474348402E-01
    1     5   0.5277489255E+01  -0.4389237657E+02   0.4486782793E-01   0.4486782793E-01   0.4486782793E-01
    1     6   0.6594794555E+01  -0.4396965451E+02   0.8881334200E-01   0.8881334200E-01   0.8881334200E-01
    1     7   0.7911999993E+01  -0.4404244254E+02   0.5849229655E-01   0.5849229655E-01   0.5849229655E-01
    1     8   0.9229437211E+01  -0.4405831588E+02   0.2414216682E-01   0.2414216682E-01   0.0000000000E+00
    2     1   0.0000000000E+00  -0.4399458479E+02   0.1112676571E-01   0.1112676571E-01   0.0000000000E+00
    2     2   0.1356841287E+01  -0.4398451885E+02   0.4270600251E-01   0.4018848625E-01   0.4018734489E-01
    2     3   0.2677587331E+01  -0.4394948430E+02   0.5479419750E-01   0.5096369018E-01   0.5096445426E-01
    2     4   0.4004269114E+01  -0.4390739111E+02   0.5004508819E-01   0.4463448973E-01   0.4464878761E-01
    2     5   0.5328036512E+01  -0.4389409127E+02   0.4291037894E-01   0.4291037894E-01   0.4291037894E-01
    2     6   0.6642907129E+01  -0.4397034020E+02   0.8879366098E-01   0.8879366098E-01   0.8879366098E-01
```

```
   2    7    0.7959713712E+01   -0.4404290631E+02    0.5713917408E-01    0.5713917408E-01    0.5713917408E-01
   2    8    0.9278358213E+01   -0.4405831588E+02    0.2414216682E-01    0.2414216682E-01    0.0000000000E+00
   3    1    0.0000000000E+00   -0.4399458479E+02    0.1112676571E-01    0.1112676571E-01    0.0000000000E+00
   3    2    0.1356624500E+01   -0.4399408010E+02    0.1114085905E-01    0.1114085905E-01    0.1114085905E-01
   3    3    0.2730952540E+01   -0.4397302719E+02    0.5096325231E-01    0.4680553493E-01    0.4683808222E-01
   3    4    0.4090362450E+01   -0.4392669466E+02    0.5272530274E-01    0.4351975945E-01    0.4355359239E-01
   3    5    0.5418808773E+01   -0.4389735067E+02    0.3886543373E-01    0.3886543373E-01    0.3886543373E-01
   3    6    0.6726370673E+01   -0.4397144829E+02    0.8809362538E-01    0.8809362538E-01    0.8809362538E-01
   3    7    0.8041492838E+01   -0.4404354368E+02    0.5543086596E-01    0.5543086596E-01    0.5543086596E-01
                                                        .......
                                                        .......
```

In each row, the force and energy for a single replica are recorded. The first column is the number of NEB steps, the second column is the ID of the replica, the third column is the hypothetical distance from the initial state, the fourth column is the energy of the replica, the fifth column is the maximum force acting on the replica, the sixth column is the maximum NEB force, and the seventh column is the maximum component of the force from the system.

- nfnebdynm.data

The history of the atomic coordinates is recorded in this file. Compared with the format of the "nfdynm.data" file obtained from usual PHASE calculations, a simpler format is adopted. Below is a typical example.

```
 #step  image  atom  cps
    0     1     1        0.0000000000        0.0000000000        0.0000000000
    0     1     2        5.2000000098        5.2000000098        0.0000000000
    0     1     3        2.6000000049        2.6000000049        2.6000000049
    0     1     4        7.8000000147        7.8000000147        2.6000000049
    0     1     5        0.0000000000        5.2000000098        5.2000000098
    0     1     6        5.2000000098        0.0000000000        5.2000000098
    0     1     7        2.6000000049        7.8000000147        7.8000000147
    0     1     8        7.8000000147        2.6000000049        7.8000000147
    0     1     9        0.0000000000        0.0000000000       10.4000000197
    0     1    10        5.2000000098        5.2000000098       10.4000000197
    0     1    11        2.6000000049        2.6000000049       13.0000000246
    0     1    12        3.1080442326        2.0700339938       20.7869859136
    0     1    13        2.0700340645        3.1080442772       20.7869859136
    0     2     1        0.0000000000        0.0000000000        0.0000000000
    0     2     2        5.2000000098        5.2000000098        0.0000000000
    0     2     3        2.6000000049        2.6000000049        2.6000000049
    0     2     4        7.8000000147        7.8000000147        2.6000000049
    0     2     5        0.0000000000        5.2000000098        5.2000000098
    0     2     6        5.2000000098        0.0000000000        5.2000000098
    0     2     7        2.6000000049        7.8000000147        7.8000000147
    0     2     8        7.8000000147        2.6000000049        7.8000000147
    0     2     9        0.0000000000        0.0000000000       10.4000000197
    0     2    10        5.2000000098        5.2000000098       10.4000000197
    0     2    11        2.6000000049        2.6000000049       13.0000000246
    0     2    12        3.2652054480        1.9060914168       19.8836995566
    0     2    13        1.9060915098        3.2652055024       19.8836994729
```

Each row corresponds to an atom belonging to some replica at some NEB step. The first column is the NEB step, the second column is the ID of the replica to which the atom belongs, the third column is the ID of the atom, and the fourth, fifth, and sixth columns are the Cartesian *x, y, z* coordinates of the atom, respectively, in Bohr units.

In usual PHASE calculations, the "nfefn.data" file and "nfdynm.data" file contain the history of the energies and atomic coordinates, respectively. In contrast, in NEB calculations, the energies and atomic coordinates corresponding to the most recent set of images are recorded; in other words, the energies and the atomic coordinates of the most recent reaction path are recorded in these files.

5.4.1.5 Example calculation: dissociative adsorption process of a hydrogen molecule on a silicon surface

Here we present an example calculation in which the dissociative adsorption process of a hydrogen molecule on a silicon surface is analyzed. The input files for this example can be found under the directory samples/neb.

The initial state for this example is a system with a surface and an H_2 molecule located far from the surface. The final state is a system with two hydrogen atoms adsorbed at the surface. The atomic configurations of the initial and final states are shown in Figure 5.13 and Figure 5.14, respectively. Since this is only an example calculation, structural optimizations for the initial and final states were not performed.

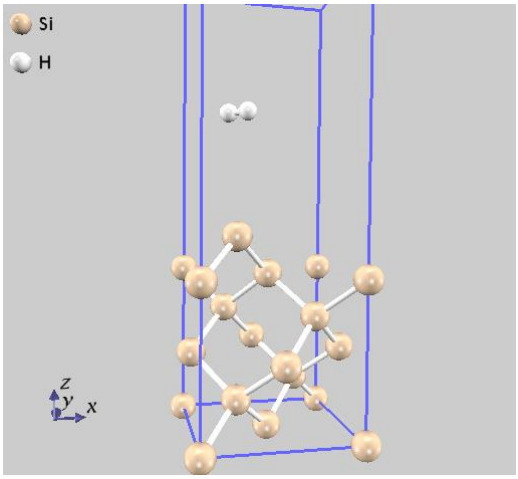


Figure 5.13 Initial state of the present example.

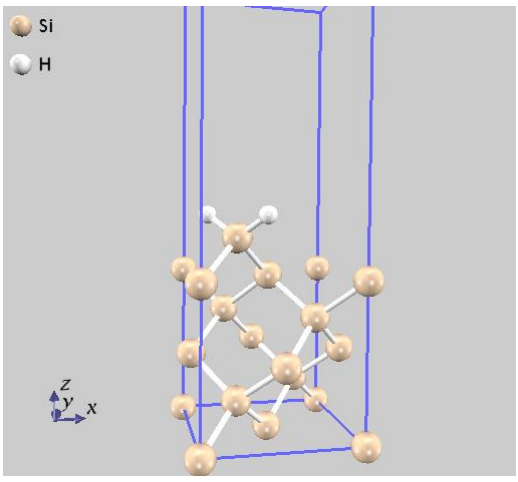


Figure 5.14 Final state of the present example.

22. input parameter file

Here we inspect the sample input parameter file. Under the control block, the overall conditions of the calculation are configured.

```
Control{
    condition = initial   ! {initial|continuation|automatic}
    cpumax = 1 day ! {sec|min|hour|day}
    max_iteration = 10000000
```

```
    multiple_replica_mode = ON
    multiple_replica_max_iteration = 2000
}
```

By setting the "multiple_replica_mode" variable to "ON," it is possible to perform NEB calculations. Also, the upper limit for the number of NEB iterations is set to 2000 by the "multiple_replica_max_iteration" variable.

Atomic configurations for the images are specified under the "structure" block under the "multiple_replica" block as follows.

```
multiple_replica{
    ....
    structure{
        number_of_replicas = 6
        replicas{
            #tag replica_number  howtogive_coordinates   end0  end1
            1            proportional              0      -1 ! 0: end0, -1:end1
            2            proportional              0      -1
            3            proportional              0      -1
            4            proportional              0      -1
            5            proportional              0      -1
            6            proportional              0      -1
        }
        endpoint_images = directin ! {no or nothing | file | directin}
        howtogive_coordinates = from_endpoint_images
        atom_list_end0{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx ry rz
            Si    0.000000000000      0.000000000000      0.000000000000
            ...
            ...
            }
        }
        atom_list_end1{
                coordinate_system = cartesian ! {internal|cartesian}
            atoms{
            #units angstrom
            #tag element rx  ry  rz
            Si    0.000000000000      0.000000000000      0.000000000000
            ...
            ...
            }
        }
    }
    ....
}
```

We set the value of the "number_of_replica" variable (number of replicas) to six. Note that this number is the number of intermediate images. In this example, the atomic coordinates for all intermediate images are constructed by a linear interpolation of the initial and final states. Under the "atom_list_end0" block and "atom_list_end1" block, the atomic coordinates for the initial and final states are specified. The format for this specification is the same as that for the usual atomic coordinate specification.

Under the "accuracy" block under the "multiple_replica" block, the optimization method and the convergence threshold are configured.

```
multiple_replica{
      ...
      accuracy{
                dt = 40 au_time
                neb_time_integral = quench
                penalty_function = off
                neb_convergence_condition = 3
                neb_convergence_threshold = 5.0e-04
      }
}
```

The time step is set to 40 au, the optimization method adopted is the "quench" method, the convergence method adopted is 3 (which means that the NEB force will be used to judge convergence), and the threshold for this convergence condition is $5(10)^{-4}$.

23. results

We now present the results that are obtained from the example. In Figure 5.15, the changes of the maximum NEB force with the number of NEB iterations are shown. The maximum force at the beginning of the simulation is significantly larger, but as the simulation proceeds, it becomes smaller. At the 41st iteration, the maximum force met the convergence criterion, and the calculation terminated normally.

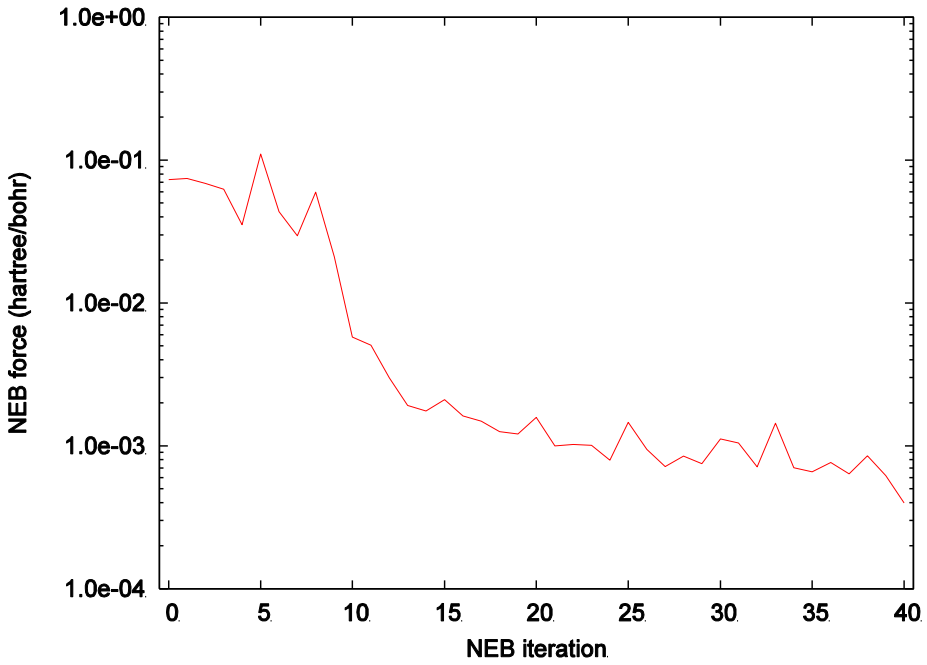


Figure 5.15 History of the maximum NEB force.

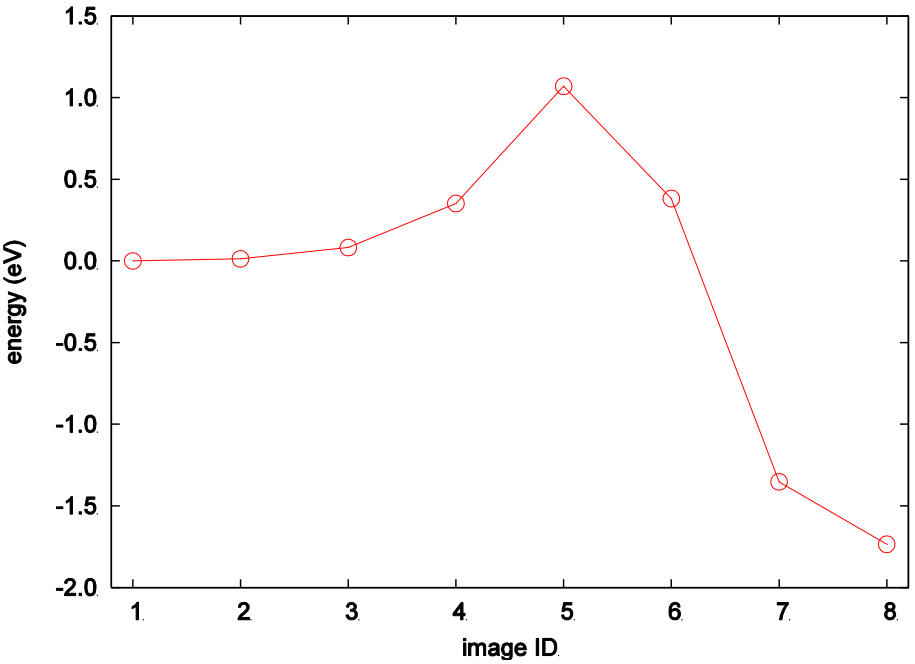


Figure 5.16 Energy of each converged image.

Figure 5.17 shows the atomic configuration of the transition state. As is clear from this figure, at the transition state, the H_2 molecule is located right above the Si surface.

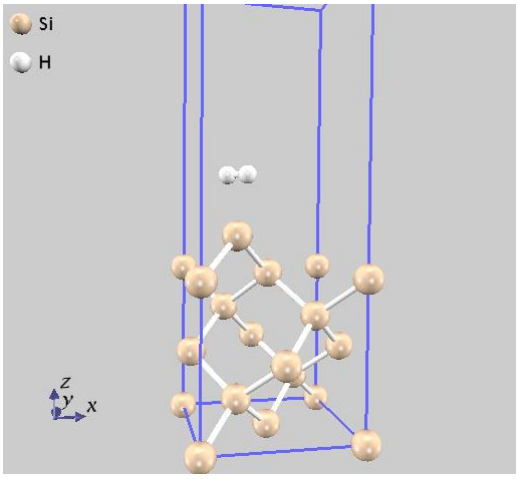


Figure 5.17 Atomic configuration of the transition state.

5.4.1.6 Notes

- Replica parallelization

The NEB method supports replica parallelization. To use this feature, an extra argument, `nr=NR`, must be supplied along with the usual parameters `ne=NE` and `nk=NK`. Here NR is the number of replica parallelizations, whose default value is 1. Note that the number of MPI processes must be equal to `NE x NK x NR`. Typically, PHASE is executed by the following command.

```
% mpirun -n N phase ne=NE nk=NK nr=NR
```

- Termination and restart of a calculation

The NEB method supports termination and restart of a calculation. Note that a slightly different termination procedure is adopted in comparison with ordinary calculations.

- Termination of a calculation

The calculation will terminate if the number of NEB iterations exceeds the value specified by the "multiple_replica_max_iteration" variable or the value specified in the "nfnebstop.data" file. The NEB calculation will also terminate if the termination conditions are met in each electronic-structure calculation of the images. In all cases, it is possible to restart the calculation from where it terminated.

The difference between the usual PHASE and NEB calculations is that the calculation will terminate right after the SCF iteration in progress is completed in the former, while in the latter, the calculation will not terminate unless all images have been processed at least once. This is needed because data for all images are required on restart.

- Restart calculation

To restart a calculation, set the "condition" variable under the "control" block to "continuation," similar to a usual PHASE calculation.

```
Control{
    condition = continuation
    ...
    ...
}
```

Files necessary to restart a calculation are as follows:
・Restart file for the NEB method: `neb_continue.data`
・Restart files for the electronic-structure calculation: restart files associated to each replica; their file names are `continue.data_r*`, `continue_bin.data_r*`, `zaj.data_r*`, and `nfchgt.data_r*` where * stands for the ID of each replica.

5.4.2 Constrained dynamics and free-energy analysis by the Blue Moon approach

5.4.2.1 Outline of the feature

One way to analyze a chemical reaction is to introduce a "reaction coordinate" that characterizes the reaction (examples of reaction coordinates are bond length, bond angle, and dihedral angle), constrain the reaction coordinate to a certain value, and then perform constrained structural optimization or molecular dynamics (MD). Since the reaction coordinates are constrained, it is possible to simulate states that would otherwise be unstable. By sequentially changing the reaction coordinate along a supposed reaction path and repeating the constrained optimization or MD, it is possible to obtain insight into the reaction. When structural optimization is performed, it is possible to obtain the minimum energy path, as in the NEB method. When a constant-temperature MD is performed, it is possible to obtain the free-energy difference between the initial and final states. In this section, we describe the method for following constrained dynamics by PHASE.

5.4.2.2 Input parameters

Tags related to this feature are tabulated in Table 5.3.

Table 5.3 List of tags related to constrained dynamics

1st level block	2nd, 3rd level block	identifiers	description
control			
		driver	Select the type of dynamics by this variable. Set this variable to "constraints" to use the constrained-dynamics feature.
structure			Block used to specify the atomic coordinates
	constrainablexx		Block in which constraints are defined. xx is the identifier for the constraint and must be an integer beginning from 1.
		type	Specify the "type" of constraint from one of the following: bond_length, bond_angle, dihedral_angle bond_length_diff, bond_angle_diff, distance_from_pos, plane, center_of_mass, or coordination_number
		atomx	Specify the ID of the atom to which this constraint is associated. x is an integer that identifies the atom. For example, when type = bond_length, two atoms will be involved; thus, the ID of the atom is specified by the variable atom1 and atom2.
		mobile	Specify whether this constraint is "mobile." The default value is "off." This variable is mainly for debugging purposes.
		monitor	Specify whether to "monitor" this constraint. When set to "on," the value of the reaction coordinate will be output to the log file after each update of the atomic coordinates. The default value is "off."
	reaction_coordinate		Block used to configure a reaction path.
		sw_reaction_coordinate	If set to "on," the constraint will be regarded as a reaction coordinate.
		init_value	Initial value of the reaction path.
		final_value	Final value of the reaction path.
		increment	Increment for the change in the reaction coordinate.
	plane		Block used to configure the "plane" when type=plane.
		normx,normy,normz	x,y,z component of the normal of the plane.
	distance_from_pos		Block used to configure the "distance_from_pos" constraint.
		posx,posy,posz	x, y, z coordinate of the target position
	coordination_number		Block used to configure the "coordination_number" constraint.
		kappa_inv	Specify the parameter $1/\kappa$ in units of length

		kappa	Specify the parameter κ in units of 1/Bohr
		rcut	Specify the parameter r_c in units of length
	center_of_mass		Block used to configure the "center of mass" constraint. Direction in which the center of mass will change can be configured under this block.
		directionx	x-direction
		directiony	y-direction
		directionz	z-direction
structure_evolution			Block used to configure the method used for the update of atomic coordinates.
		method	Specify the method of the atomic coordinates update. For constrained dynamics, one of the following—quench, damp, velocity_verlet, or temperature_control—can be used.

To activate the constrained-dynamics feature, the "driver" variable under the "control" block must be set to "constraints."

```
condtion{
  ...
  driver=constraints
  ...
}
```

Next, the "constrainablexx" block must be defined under the "structure" block. Here xx stands for an integer beginning from 1.

```
structure{
    ...
    ...
    constrainable1{
       type=bond_length
       atom1=1
       atom2=2
       mobile = off
       monitor = off
       reaction_coordinate{
          sw_reaction_coordinate=on
          init_value = 2.4 angstrom
          increment  = 0.1 angstrom
          final_value = 8.0 angstrom
       }
       plane{
           normx=1
           normy=0
           normz=0
       }
       coordination_number{
           kappa = 5.0
           rc     = 2.0 angstrom
       }
    }
    ...
    ...
}
```

There are no upper limits on the number of constraints that can be defined. Note that consecutive integers must be used for xx. For example, if three blocks "constrainable1," "constrainable2," and "constrainable4" are defined, only the first two will be interpreted. Also, note that if inconsistent constraints are defined, the program will abort.

Under the constrainablexx block, the following variables/blocks can be defined.

"type" variable		Specify the "type" of the constraint.
	bond_length	Constrain the distance between two atoms.
	bond_angl	Constrain the angle among three atoms.
	dihedral_angle	Constrain the dihedral angle among four atoms.
	bond_length_diff	Constrain the difference between a pair of bond lengths
	bond_angle_diff	Constrain the difference between a pair of bond angles.
	distance_from_pos	Constrain the distance between an atom and a specified position.
	plane	Constrain the atom within a specified plane.
	center_of_mass	Constrain the center of mass of the specified atoms.
	coordination_number	Constrain the coordination number of a specified atom. Here the coordination number of atom 0 is defined by $$\sigma = \sum_{i \neq 0} S(\lvert \boldsymbol{r}_i - \boldsymbol{r}_0 \rvert),$$ $$S(r) = \frac{1}{\exp\left[\kappa\left(r - r_c\right)\right] + 1}$$ κ, r_c are parameters that should be defined so the function S becomes sufficiently small at the first coordination shell.
"atomx" variable		Specify the atoms involved in the current constraint. x is an integer that identifies the atoms; for example, when the "type" is "bond_length," specify the ID of the first atom by the "atom1" variable and that of the second atom by the "atom2" variable. If the "type" is "coordination_number," then the central atom for which the coordination number is calculated is specified by the "atom1" variable.
"mobile" variable		Specify whether the constraint is "mobile." If "on" is specified, the constraint is considered to be "mobile," and thus will not be constrained. The default value is "off."
"monitor" variable		Specify whether to "monitor" the constraint. When set to "on," the value of the constraint will be calculated and written to the log file. The default value is "off."
"reaction_coordinate" block		Block used to specify that the constraint is a "reaction coordinate" (i.e., can be sequentially varied).
	sw_reaction_coordinate	When set to "on," the constraint is regarded as a "reaction coordinate."
	init_value	Specify the initial value of the reaction coordinate in the corresponding units. If unspecified, then the value calculated from the input atomic coordinates will be used as the initial value. If the value specified by the "init_value" variable and that calculated from the input atomic coordinates are different, then the input atomic coordinates are first adjusted to fulfill the input specification. Thus, the maximum force (which includes the force of constraint) for the first ionic iteration can become significantly large, but this is normal.
	final_value	Specify the final value for the reaction coordinates in the corresponding units.
	increment	Specify the increment for the change of the reaction coordinate. The number of reaction coordinates considered will approximately be (final_value − init_value)/increment When the reaction coordinate is sequentially varied, the

		situations below are special cases. These cases should be handled with caution.
	case when type=plan	In this case, the origin of the plane is varied. The origin of the plane will automatically be resolved from the normal vector and the atomic coordinates; the origin thus resolved will be varied along the direction of the normal vector when "sw_reaction_coordinate" is set to on. The "init_value" variable should be set to 0, and the "increment" and "final_value" variables should be chosen so that the desired shift of the origin is realized.
	case when type=center_of_mass	In this case, the center of mass will be shifted in the specified direction. The "init_value" variable should be set to 0, and the "increment" and "final_value" variables should be chosen so that the desired shift for the center of mass is realized.
"plane" block		
	normx	x coordinate for the normal of the plane.
	normy	y coordinate for the normal of the plane.
	normz	z coordinate for the normal of the plane.
"distance_from_pos" block		Specify a point in real space. This block is used when type=distance_from_pos is specified.
	posx	Specify the x coordinate of the target point.
	posy	Specify the y coordinate of the target point.
	posz	Specify the z coordinate of the target point.
"coordination_number" block		Block used to specify the parameters κ, r_c in the formula for the calculation of the coordination number. The following variables can be defined.
	kappa_inv	Specify the value of $1/\kappa$ in units of length.
	kappa	Specify the value of κ itself in 1/Bohr units. This variable will be preferred over kappa_inv. Note that the unit cannot be explicitly specified for this variable, since the unit 1/length is not registered in the PHASE unit list.
	rcut	Specify the value of r_c in units of length.
"center_of_mass" block		When type=center_of_mass and sw_reaction_coordinate=on, configure the direction of the shift for the center of mass at this block.
	directionx	Specify the x coordinate of the abovementioned direction.
	directiony	Specify the y coordinate of the abovementioned direction.
	directionz	Specify the z coordinate of the above-mentioned direction.

After specification of the constraints, the algorithm used for ion dynamics is specified. As in usual PHASE calculations, this is done under the "structure_evolution" block

```
structure_evolution{
    method=quench
    dt=40
    ...
}
```

For the "method" variable, the following values are supported: quench, damp, velocity_verlet, and temperature_control. Note that the BFGS, GDIIS, and CG optimizers are not available when constraints are imposed. The value "damp" is used to perform optimization by the damped molecular dynamics method. In many cases, this method allows a larger time step than the "quench" method, leading to a faster convergence.

Variation of a single reaction coordinate can be done by specifying the "init_value," "final_value," and the

"increment" variable under the "reaction_coordinate" block. In this case, the reaction coordinate will simply change from the "init_value" to "final_value" uniformly. However, the behavior is more complicated when multiple reaction coordinates are allowed to vary.

- Method to vary multiple reaction coordinates

Here we describe the behavior of the program when multiple reaction coordinates are defined. For example, consider the following input specification.

```
structure{
    ....
    ....
    constrainable1{
      mobile = off
      monitor = on
      type = dihedral_angle
      atom1 = 2
      atom2 = 4
      atom3 = 3
      atom4 = 1
      reaction_coordinate{
        sw_reaction_coordinate = on
        init_value = -179 degree
        final_value = -1 degree
        increment = 5 degree
      }
    }
    constrainable2{
      type=bond_length
      monitor=on
      atom1=3
      atom2=4
      reaction_coordinate{
        sw_reaction_coordinate=on
        init_value = 1.2 angstrom
        final_value = 1.6 angstrom
        increment = 0.05 angstrom
      }
    }
    ....
    ....
}
```

Under the "constrainable1" block, the dihedral angle is specified to change from –179° to –1° in increments of 5°. Under the "constrainable2" block, the bond length is configured to change from 1.2 Å to 1.6 Å in increments of 0.05 Å. In this case, the bond length will first be fixed at 1.2 Å, and the dihedral angle will be changed from –179° to –1°. After the optimization or MD is done at –1°, the bond length is increased to 1.25 Å, and the dihedral angle is then changed from –1° to –179°. This variation scheme prevents radical changes between adjacent sets of reaction coordinates.

When n_α reaction coordinates are constrained, where α denotes the type of constraint, the number of reaction coordinates will be $\prod_\alpha n_\alpha$. This may lead to a massive number of reaction coordinates that need to be considered. If a more flexible specification of the reaction coordinates is desired, it is possible to specify the manner in which the reaction coordinates change via an external file. This is described in the next section.

- Method to vary the reaction coordinates through an external file

The method of varying the reaction coordinate is basically defined under the "reaction_coordinate" block. By this method, only uniform variations in reaction coordinates can be specified. If a more flexible specification is desired, reaction coordinates can be read from an external file. To use this feature, first configure the constrainablexx block as usual. Under the "structure" block, set the following variable.

```
structure{
  ....
  reac_coord_generation = via_file
  ....
}
```

Finally, create a "reac_coordinate.data" file under the working directory and edit its contents as follows.

```
       1       -1.9373154697        2.2676711906
       2       -1.7627825445        2.2676711906
       3       -1.5882496193        2.2676711906
       4       -1.4137166941        2.2676711906
       5       -1.2391837689        2.2676711906
       6       -1.0646508437        2.2676711906
       7       -0.8901179185        2.2676711906
       8       -0.7155849933        2.2676711906
       9       -0.7155849933        2.3621574902
      10       -0.8901179185        2.3621574902
      11       -1.0646508437        2.3621574902
      12       -1.2391837689        2.3621574902
      13       -1.4137166941        2.3621574902
      14       -1.5882496193        2.3621574902
      15       -1.7627825445        2.3621574902
      16       -1.9373154697        2.3621574902
      17       -1.9373154697        2.4566437898
      18       -1.7627825445        2.4566437898
      19       -1.5882496193        2.4566437898
      20       -1.4137166941        2.4566437898
      21       -1.2391837689        2.4566437898
      22       -1.0646508437        2.4566437898
      23       -0.8901179185        2.4566437898
      24       -0.7155849933        2.4566437898
                ......
                ......
                ......
```

Each line corresponds to a "reaction coordinate set." In the first column, specify the integer that identifies each reaction coordinate set. In the remaining columns, the reaction coordinate is specified in the same order as that used in the input parameter file. In this example, two types of reaction coordinates are considered. In the first reaction coordinate set, the first reaction coordinate will take the value "–1.9373154697," while the second reaction coordinate will take the value "2.26711906." These values must be in PHASE default units, i.e., Bohr for length, radian for angle.

5.4.2.3 Execution

When using the constrained-dynamics feature, PHASE should be executed by the following command.

```
% mpirun -np NP phase ne=NE nk=NK nr=NR
```

Here `NP` is the number of MPI processes, `NE` is the number of band parallelizations, `NK` is the number of ***k*** point parallelizations, and `NR` is the number of reaction coordinates to be handled in parallel. The relation

`NP = NE x NK x NR` must hold. The default values are ne=NP, nk=1, and nr=1, but it is strongly recommended to explicitly specify each parameter.

5.4.2.4 Output of the results

When the reaction coordinates are not allowed to vary, the output data are the same as those obtained from the usual PHASE calculations. The history of the energy and force will be recorded in the file specified by the F_ENF file pointer in the "file_names.data" file, while the history of atomic coordinates will be recorded in the file specified by the F_DYNM file pointer in the "file_names.data" file. Note that the maximum force recorded includes the contribution from the force of constraint.
However, when the reaction coordinates are allowed to vary, the following files will be output. (We assume here that the file name for the F_ENF file is "nfefn.data," while that for the F_DYNM file is "nfdynm.data.")

nfefn.data.reacxx	F_ENF file for the xxth reaction coordinate.
nfefn.data.converged (structural optimization only)	This file records the converged energy at each reaction coordinate. Each line of the file corresponds to a reaction coordinate. The value of the reaction coordinate itself, converged energy, and the maximum force acting on the atoms are written. By plotting the relation between the reaction coordinate and the converged energy, it is possible to analyze the relation between the reaction path and energy.
nfdynm.data.reacxx	F_DYNM file for the xxth reaction coordinate.
nfdynm.data.converged (structural optimization only)	This file records the converged atomic configuration at each reaction coordinate.
nfbluemoon.data.reacxx (molecular dynamics only)	The Lagrange multiplier, which is necessary for the calculation of the free energy, is recorded for the xxth reaction coordinate.

In addition, the restart files, wave function files, and charge files are generated for each reaction coordinate.

5.4.2.5 Free-energy calculation by the Blue Moon approac

24. Outline of the feature

By using data obtained from the constrained MD, it is possible to calculate the free-energy difference along the reaction path. [?]。 The free-energy difference when the reaction coordinate changes from ξ_1 to ξ_2 can be calculated from

$$W(\xi_1) - W(\xi_2)) = \int_{\xi_2}^{\xi_1} d\,\xi \frac{\partial W}{\partial \xi}.$$

The derivative of the free energy with respect to the reaction coordinate, $\left\langle \frac{\partial W}{\partial \xi} \right\rangle_{\xi}$, is called the mean force. It is related to the derivative of the Hamiltonian with respect to the reaction coordinate by

$$\frac{\partial W}{\partial \xi} = \left\langle \frac{\partial H}{\partial \xi} \right\rangle_{\xi}^{\mathrm{cond}}.$$

Here $\langle \cdots \rangle^{\mathrm{cond}}$ is the "conditional statistical average." The conditional statistical average and the statistical average obtained from the constrained MD are not simply related. However, (21) can be obtained from the Lagrange multiplier λ that is calculated during the constrained MD,

$$\frac{\partial W}{\partial \xi} = -\frac{\langle |\Xi|^{-1/2} \lambda \rangle}{\langle |\Xi|^{-1/2} \rangle}$$

$$\Xi = \sum_i \frac{1}{m_i} \frac{\partial \xi}{\partial \vec{r}_i} \frac{\partial \xi}{\partial \vec{r}_i}$$

To be exact, (22) has a correction term, but in practice, the correction is not important.

To calculate the free-energy difference from the results obtained from the constrained MD by PHASE, it is necessary to use the "bluemoon" program included in the PHASE package. Note that the bluemoon program can be applied only to the case of a single reaction coordinate.

25. Compilation of the bluemoon program

The source code for the bluemoon program is located under the "src_bm" directory under the PHASE installation directory. The bluemoon program requires a Fortran90 and a C compiler. To complile the bluemoon program, set the environment variable F90 to your Fortran90 compiler and the environment variable CC to your C compiler and then type "make." The example below assumes that the shell in use is bash, the command for the Fortran90 compiler is f90, and the command for the C compiler is cc.

```
% cd phase_v1000
% cd src_bm
% export F90=f90
% export CC=cc
% make
% make install
```

If the environment variables F90 and CC are not defined, "gfortran" and "gcc," respectively, will be used by default. By issuing the command `% make install`, it is possible to move the created program to the "bin" directory.

26. Input parameter file for the bluemoon program

The input parameter file for the bluemoon program is the same as that for PHASE. The bluemoon program is configured under the thermodynamic_integration block. The following is a typical example.

```
thermodynamic_integration{
  nsteps=2000
  nequib=1000
  istart_reac_coords=1
  nreac_coords=14
  nsample=10
  smooth=off
  basedir=.
}
```

Under the "thermodynamic_integration" block, it is possible to define the following variables.

nsteps	Total MD steps performed for each reaction coordinate. The default value is 2,000; set it according to the calculation performed.
nequib	Specify the number of steps to be ignored for equilibration.
istart_reac_coords	Specify the ID of the reaction coordinate that should be considered first. The default value is 1.
nreac_coords	Specify the ID of the reaction coordinate that should be considered last.
nsample	Specify the number of samples to be used for estimating statistical errors.
smooth	Specify whether the results should be smoothed by a cubic spline. The default value is "off."
basedir	Specify the directory in which the results should be written. The default is the current

directory.

27. Execution of the bluemoon program

After the input parameter file is configured, run the bluemoon program as follows.

```
% bluemoon inputfile
```

The argument is the input parameter file. If no argument is given, the string "nfinp.data" will be adopted.

28. Output of the results

Calculations by the bluemoon program should finish within a few seconds. The following files will be created.

- potential_of_mean_force.data

The calculated free energy is written. The file format is as follows.

```
#value, potetial of mean force in Hartree, eV, kcal/mol, kJ/mol
2.4566437898 -0.0215821952 0.0003443042 -0.5872816633 0.0093689992 -13.5430301648
0.2160541460 -56.6640534911 0.9039707906
2.2676711910 -0.0224669448 0.0003796767 -0.6113569350 0.0103315334 -14.0982188431
0.2382507016 -58.9869635475 0.9968412043
2.0786985910 -0.0226882285 0.0004435350 -0.6173783747 0.0120692073 -14.2370764737
0.2783223931 -59.5679440305 1.1645012069
                                                    .............
                                                    .............
                                                    .............
```

Each line corresponds to one reaction coordinate. The first column is the value of the reaction coordinate, the remaining correspond to the free energy and estimated statistical error. The second and third columns contain the free energy and the error in Hartree units, the fourth and fifth columns contain the free energy and error in eV units, the sixth and seventh columns contain the free energy and error in kcal/mol units, and the eighth and ninth columns contain the free energy and error in kJ/mol units, respectively..

- mean_force_raw.data

This file contains the mean force for the considered reaction coordinates. The format of the file is as follows.

```
        2.4566437898        0.0066082098        0.0188118786
        2.2676711910        0.0034758686        0.0099291734
        2.0786985910       -0.0009537509        0.0028573953
        1.8897259920       -0.0074922663        0.0213420952
        1.7007533930       -0.0098143395        0.0279585555
        1.5117807940       -0.0157974842        0.0449758051
        1.3228081950       -0.0161451965        0.0459534340
                            .............
                            .............
                            .............
```

As in the "potential_of_mean_force.data" file, each line corresponds to one reaction coordinate. The first column is the value of the reaction coordinate, the second column is the mean force (the unit is Hartree/unit of the corresponding reaction coordinate), and the third column is the statistical error.

- mean_force_smoothed.data

If the data were smoothed by cubic splines, the mean force is first smoothed and then the integration is done. This file contains the results for the smoothed mean force. The format of the data is the same as that in the "mean_force_raw.data" file, but without the statistical error column.

5.4.2.6 Example calculation: rotation barrier of H_2O_2 and H_2S_2 molecules

As an example of the constrained-dynamics feature, we present an analysis of the rotational barrier for H_2O_2 and H_2S_2 molecules. H_2O_2 and H_2S_2 are simple molecules whose structures are shown in Figure 5.18. It is known that the rotational potential of the dihedral angle formed by HOOH (HSSH) atoms is a W-type potential. This originates from competition between the H–H interaction and H–O(S) interaction. By performing structural optimization with the constrained dihedral angle, we determine whether such behavior can be reproduced.

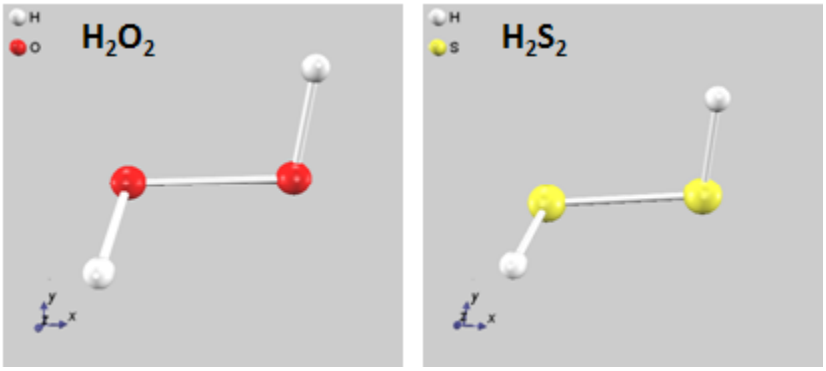


Figure 5.18 Molecular structure of H_2O_2 and H_2S_2 molecules.

The input data for this example are located under the subdirectories of the samples/constraints directory, H2O2 and H2S2. Under the "structure" block, the following settings can be found.

```
structure{
    constrainable1{
      type = dihedral_angle
      atom1 = 2
      atom2 = 4
      atom3 = 3
      atom4 = 1
      reaction_coordinate{
        sw_reaction_coordinate = on
        init_value = 9 degree
        final_value = 179 degree
        increment = 10 degree
      }
    }
    ...
    ...
}
```

The "constrainable1" block is defined, and the constraint is configured. In this example, only a single constraint is defined; any number of constraints can be defined, provided that they are consistent with one another. In this example, the dihedral angle is constrained; thus, the "type" variable has the value "dihedral_angle." Also, the four atoms needed to calculate the dihedral angle are specified by the variables "atom1" through "atom4." Further, the "reaction_coordinate" block is defined, and the variation of the constraint is defined. The "sw_reaction_coordinate" variable is set to "on," and the "init_value" is set to 9°, the "final_value" is set to 179°, and the "increment" is set to 10°.

Figure 5.19Figure 5.19 shows the changes in the computed energy with the dihedral angle. The figure also includes experimental results. From this figure, we see that the calculated results are in good agreement with experimental results (the difference is about 1 kcal/mol). There are two main differences between the results obtained for H_2O_2 and H_2S_2. The first is the value of the stable dihedral angle. For H_2O_2, the stable dihedral angle is around the tetrahedral angle of 109.5°, while for H_2S_2, the stable dihedral angle is close to

90°. The second difference is the height of the *trans* barrier (the barrier close to 180°). Compared with H_2O_2, the *trans* barrier for H_2S_2 is about six times higher. Both points are well reproduced, supporting the validity of this example calculation.

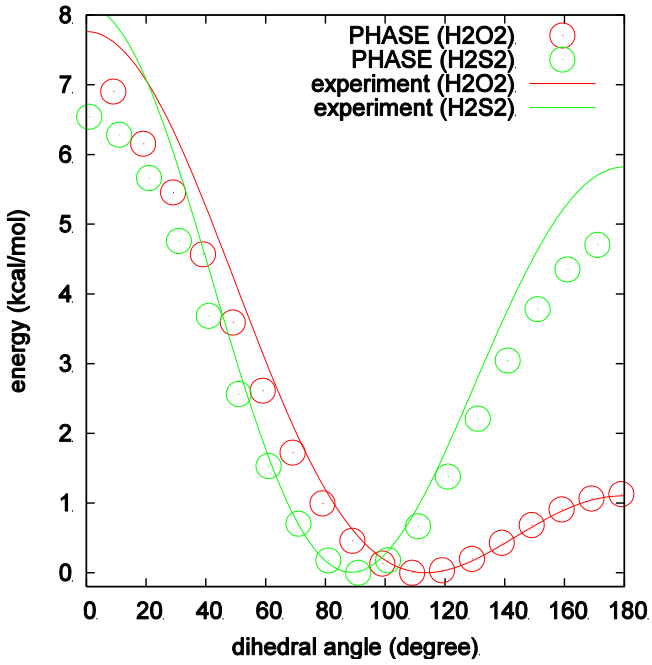


Figure 5.19 Relation between the energy and dihedral angle for H_2O_2 and H_2S_2 molecules.

5.4.2.7 Notes

- Constrained dynamics can be used with any type of pseudopotential.
- Restart calculations are supported.
- When varying the reaction coordinate, the reaction coordinates can be handled in parallel by the following command, as in the case for the NEB method:

```
% mpirun -n NP phase ne=NE nk=NK nr=NR
```

5.4.3 Metadynamics

5.4.3.1 Outline of the feature

Metadynamics is a method to efficiently analyze processes that have finite activation barriers, such as chemical reactions. [?,?] In the metadynamics method, "collective variables" (denoted by $S_\alpha(r)$) are introduced. These "collective variables" are a set of reaction coordinates defined from the atomic configuration under interest. Each collective variable is assigned a fictitious "particle," and metadynamics refers to the dynamics of this fictitious "particle." By aptly designing the algorithm of metadynamics, it is possible to efficiently explore the free-energy surface spanned by the considered collective variables. In this section, we describe the use of the metadynamics method implemented in PHASE.

In the metadynamics method, history-dependent bias potentials are added at intervals (usually a few ten to a few hundred MD steps). By this operation, points in the free-energy space once visited will be disfavored. If a sufficient number of bias potentials are added, the $V(t,s)$ will fill the free-energy space, and the reaction under consideration will freely occur. The negative of the accumulated $V(t,s)$ that realizes such a situation is regarded as the free energy itself.

A schematic of the metadynamics simulation is shown in Figure 5.20. In this figure, the simulation starts from the valley numbered 1. After the bias potentials labeled 2 and 3 are added, the system escapes from the valley and evolves to a new local minimum (the left-most valley in the figure). Further, after the bias potentials 4, 5, and 6 are added, the system escapes from the second valley and evolves to the state with the lowest energy (the right-most valley in the figure). Finally, after the bias potentials 7 and 8 are added, the collective variables freely evolve in the space spanned by the collective variables. By changing the sign of the accumulated bias potential, the free-energy surface is obtained.

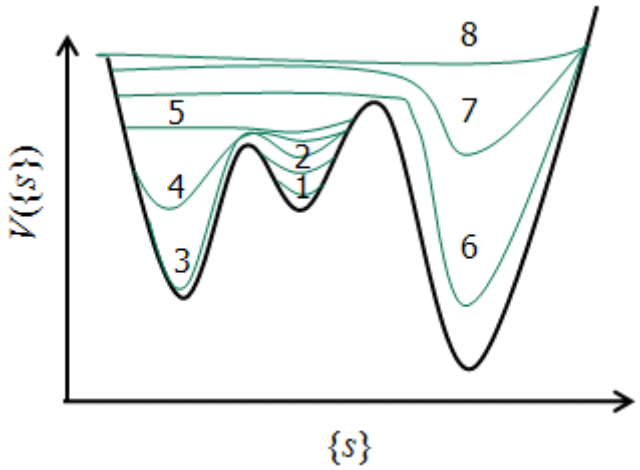


Figure 5.20 Schematic illustration of a metadynamics simulation.

The marked characteristic of metadynamics is that it follows the trajectories of the dynamic variables associated with the reaction coordinates. With this idea, it is expected that the dynamic variable itself will automatically search for a plausible reaction path. Further, it is much easier to consider multiple reaction coordinates, compared with other methods, such as the blue moon approach. Therefore, the method is suitable when the reaction path is not obvious or when multiple reaction coordinates must be considered.

Depending on the addition of the bias potentials, it is possible either to perform an approximate exploration

of the free-energy surface or to obtain a detailed and accurate free-energy profile. A metadynamics simulation proceeds by adding the bias potentials at predetermined intervals. When a bias potential is added, the potential must be accumulated for every bias potential added at that point; thus, the construction of the accumulated bias potential requires an $O(t^2)$ operation (although in first-principle calculations, this should not be a problem).

In metadynamics, the Hamiltonian is written as

$$H_{\text{meta}} = H_{\text{MD}} + \sum_{\alpha} \frac{1}{2}\mu_\alpha \dot{s}_\alpha^2 + \sum_{\alpha} \frac{1}{2} k_\alpha (S_\alpha(\boldsymbol{r}) - s_\alpha)^2 + V(t,s),$$

$$V(t,s) = \sum_{t_i<t} w \exp\left[-\sum_{\alpha} \frac{\left(s_\alpha(t) - s(t_i)\right)^2}{2\Delta s_\alpha^2} \right]$$

Here α is a variable that distinguishes the collective variables, μ_α and s_α are the mass and coordinate of the hypothetical particle, respectively, $S_\alpha(\boldsymbol{r})$ is the value of α, k_α is the spring constant for the spring that binds the coordinate of the hypothetical particle, and $V(t,s)$ is the bias potential. By recording the accumulated bias potential, it is possible to obtain the free-energy surface. The dynamics derived from the above Hamiltonian can be summarized as follows.

- The system will be loosely tied to the coordinates of the fictitious particles via the collective variables.
- The point in free space where the fictitious particles have already visited will be disfavored by the effect of the bias potential.

If the time scale for the dynamics of the fictitious particles is sufficiently longer than that for the system, it is expected that the dynamics of the system will be decoupled from that of the fictitious particle. Therefore, over a shorter time scale, the system will follow the correct dynamics, while over a longer time scale, the system will slowly explore the free-energy space spanned by the collective variables. Therefore, the mass of the collective variables should be chosen such that the vibrational modes of the collective variables are slower than those of the system.

Metadynamics simulations have also been performed in another manner, in which fictitious particles are not used. Instead, the bias potentials directly affect the system, not indirectly through fictitious particles [?]. In this approach, the mass and spring constant of the collective variables need not be defined, leading to a simpler execution of the method.

5.4.3.2 Input parameters

Table 5.4 identifies tags related to the metadynamics method.

Table 5.4 List of tags related to the metadynamics method

1st level block	2nd, 3rd level block	identifiers	description
control			
		driver	Select the type of dynamics by this variable. Set this variable to "meta_dynamics" to use the metadynamics feature.
meta_dynamics			Block used to configure meta dynamics.
		meta_dynamics_type	Select the "type" of metadynamics from one of the following: "bias_and_fictitious," "bias_only," or "bias_generation." The default value is "bias_only."
		max_bias_update	Set the maximum number of bias potential

			updates. If the number of bias potential updates exceeds this specification, the program will terminate. The default value is −1; note that when a negative value is specified for this variable, the program will not terminate by this condition.
		extensive_output	Set this option to "on" to obtain debug output in the log file. "Off" (default value) is recommended.
		output_per_rank	When set to "on," the output per replica will be obtained when replica parallelization is enabled. "Off" (default value) is recommended.
	collective_variable		Block to define collective variables.
		type	Variable that specifies the "type" of the collective variable. The choices are the same as those for the constrained dynamic, namely: "bond_length," "bond_angle," "dihedral_angle" "bond_length_diff," "bond_angle_diff," "distance_from_pos," "plane," or "center_of_mass," "coordination_number"
		atomx	Specify the ID of the atom to which this constraint is associated. x is an integer. For example, when type = bond_length, two atoms will be involved; thus, the ID of the atom is specified by the variables atom1 and atom2.
		k	Variable to specify the spring constant.
		delta_s	Variable to specify the "width" of the bias potential, Δs.
		smin	Variable to specify the minimum value for the bias potential output.
		smax	Variable to specify the maximum value for the bias potential output.
		ds	Variable to specify the increment for the bias potential output.
		control_velocity	When set to "on," control_velocity will control the velocity of the collective variables.
		mass_thermo	Mass of the thermostat when "control_velocity" is "on"
		target_KE	Target kinetic energy of the collective variables.
	plane		Block used to configure the "plane" when type=plane.
		normx,normy,normz	x,y,z component for the normal of the plane
	distance_from_pos		Block used to configure the "distance_from_pos" constraint.
		posx,posy,posz	x, y, z coordinate of the target position
	coordination_number		Block used to configure the "coordination_number" constraint.
		kappa_inv	Specify the parameter $1/\kappa$ in units of length
		kappa	Specify the parameter κ in units of 1/Bohr
		rcut	Specify the parameter r_c in units of length
	bias_potential		Block used to configure the properties of the bias potential.
		height	Set the "height" of the bias potential in units of energy.
		update_frequency	Specify the interval of the bias potential update. For example, if this variable is set to 10, the bias potential will be added after every 10 MD steps. The default value is 20.
		output_frequency	When "meta_dynamics_type" is "bias_generation," then this variable is used to specify the interval of the bias

			potential output. The default value is 10.
	continuation_strategy		Block to configure the method to perform restart calculations when replica parallelization is enabled.
		randomize_velocity	When set to "on," the initial velocity for restart calculations will be calculated from the random normal distribution.
		scale_velocity	When set to "on," the initial velocities for restart calculations will be scaled according to the value of "velocity_scaling_factor" described below.
		velocity_scaling_factor	Scaling factor for the velocity, as described above. The default value is 1.
		configuration_from_input	When set to "on," configuration_from_input will read the coordinates for restart calculations from the input parameter file, not from the restart files. The default value is "off."

We now describe in detail the configurations necessary to perform metadynamics simulations. The following configurations are required.

- Enable the metadynamics method.
- Configure the behavior of the method (the method to be used for the dynamics, format of the output, etc.).
- Configure constant-temperature MD.
- Configure the collective variables (definitions are needed for all reaction coordinates that are included in the collective variables).
- Configure the bias potential (height and width of the bias potential and the update frequency must be configured).
- If replica parallelization is to be enabled, configure replica parallelization-related variables.

- Enable the metadynamics method

To enable metadynamics, the "driver" variable under the "control" block is set to "meta_dynamics."

```
control{
    driver = meta_dynamics
}
```

By this specification, PHASE will call the main routine of the metadynamics method instead of the usual PHASE main routine.

- Configure the overall behavior of metadynamics

To configure the overall behavior of the metadynamics method, create a "meta_dynamics" block and define variables and blocks under it. Here is an example:

```
meta_dynamics{
  meta_dynamics_type = bias_only
  max_bias_update = -1
  extensive_output=on
  output_per_rank=on
  output_cvar_every_step=off
  continuation_strategy{
    randomize_velocity=on
    scale_velocity=off
    velocity_scaling_factor=0.7
    configuration_from_input=off
    ...
    ...
  }
```

- Under the "meta_dynamics" block, the following blocks and variables can be defined.

"meta_dynamics_type" Select the dynamics type for meta dynamics.

variable		The choices are "bias_and_fictitious," "bias_only," or "bias_generation." When "bias_and_fictitious" is chosen, the dynamics of the system and fictitious particles are followed. When "bias_only" is chosen, metadynamics will be performed without fictitious particles. When "bias_generation" is chosen, metadynamics will not be performed; only the construction and output of the bias potential using files under the current directory will be performed.
"max_bias_update" variable		This variable is used to specify the maximum number of bias potential updates. If the number of bias potential updates exceeds the value specified here, the simulation will terminate. If a negative value is specified, the program will not terminate by this condition. This is the default behavior.
"output_per_rank" variable		By setting this variable to "on," it is possible to obtain output data per MPI process rank when replica parallelization is enabled. The default value is "off."
"extensive_output" variable		By setting this variable to "on," it is possible to output data that are usually not important, such as the velocity and force of the fictitious particles.
"continuation_strategy" block		Block to configure how restart calculations should be performed when replica parallelization is enabled. If the number of replica parallelizations is changed from consecutive simulations, it is not possible to rigorously reproduce the situation of the last run. Under this block, we configure the method to resolve restart data in such situations.
	"randomize_velocity" variable	If set to "on," the velocity will be created from a random normal distribution instead of being read from the restart file. The default value is "off."
	"scale_velocity" variable	If set to "on," the velocity will be scaled by the value of the "velocity_scaling_factor" variable explained below. The default value is "off."
	"velocity_scaling_factor" variable	The scaling factor used to scale the velocity when the "scale_velocity" variable is set to "on." The default value is 1.
	"configuration_from_input" variable	When set to "on," the atomic coordinates will be read from the input parameter file instead of the restart file. The default value is "off."

- Define the collective variables

Collective variables are, in short, a set of reaction coordinates. Specification of collective variables is done under the "meta_dynamics" block. A typical example follows.

```
meta_dynamics{
  ....
  ....
  collective_variable{
    mass=1000
```

```
    k=100
    delta_s = 0.08
    control_velocity=on
    mass_thermo = 50
    target_KE = 0.1
  }
  collective_variable1{
     type=bond_length
     atom1=5
     atom2=4
     delta_s=0.05 angstrom
     smin=1 angstrom
     smax=5 angstrom
     ds = 0.1 angstrom
   }
  ....
  ....
}
```

First, we create a "collective_variable" block under the "meta_dynamics" block. Under the "collective_variable" block, we configure settings that are common to all collective variables. The preferred settings under the "collective_variable*xx*" block are described below.

Next, we define the "collective_variable*xx*" block to the extent needed. Here *xx* is the ID for the collective variable. Any number of collective variables can be defined, but consecutive integers beginning from 1 must be specified for *xx*. For example, if three blocks "collective_variable1," "collective_variable2," and "collective_variable4" are defined, only "collective_variable1" and "collective_variable2" will be interpreted.

The following variables can be defined under the "collective_variable" and "collective_variable*xx*" blocks, as in the case of constraints.

"type" variable		"Type" of the collective variable is specified.
	bond_length	Distance between two atoms will be used as the collective variable.
	bond_angle	Angle among three atoms will be used as the collective variable.
	dihedral_angle	Dihedral angle among four atoms will be used as the collective variable.
	bond_length_diff	Difference between two bond lengths will be used as the collective variable.
	plane	Position of an atom in some plane will be used as the collective variable.
	center_of_mass	Center of mass for a group of atoms will be used as the collective variable.
	coordination_number	Coordination number of the specified atom will be used as the collective variable.
	distance_from_pos	Distance between a specified atom and a specified position in space will be used as the collective variable.
"atomx" variable		Specify the atom(s) associated to the collective variable under consideration. x is a number that identifies the atoms. For example, if the distance between two atoms is the collective variable, the ID of the first atom is specified by "atom1," while that of the second atom is specified by "atom2." When "coordination_number" is specified for the "type" variable, the atom whose coordination number is to be calculated is specified by

		the "atom1 variable.
"plane" block		Block used to specify the normal vector of the plane when type=plane. The following values can be defined.
	normx	x-coordinate of the normal vector.
	normy	y-coordinate of the normal vector.
	normz	z-coordinate of the normal vector.
"coordination_number" block		In case of coordination number constraints, the coordination number of atom i, n_i is defined as $n_i = \sum_{j \neq i} \frac{1}{exp\,[\kappa(\lvert \boldsymbol{r}_j - \boldsymbol{r}_i \rvert - r_c)]+1}$. This block is used to specify the value of κ, r_c used in the definition above.
	kappa_inv	Specify the value of $\frac{1}{\kappa}$ in units of length.
	kappa	Specify the value of κ in 1/bohr units. This will be preferred over kappa_inv.
	rcut	Specify the value of r_c in units of length.
"mass" variable		Specify the mass of the fictitious particle. It will be interpreted only when the value of "meta_dynamics_type" is "bias_and_fictitious."
"k" variable		Specify the spring constant that determines the coupling between the collective variable and the coordinate of the fictitious particle. It will be interpreted only when the value of "meta_dynamics_type" is "bias_and_fictitious."
"delta_s" variable		Specify the value of δs_α in eq. (23)
"smin" variable		Specify the minimum value for the bias potential output.
"smax" variable		Specify the maximum value for the bias potential output.
"ds" variable		Specify the interval for the bias potential update.
"control_velocity" variable		When set to "on," the velocity of the fictitious particle will be controlled by a thermostat. It will be interpreted only when the value of "meta_dynamics_type" is "bias_and_fictitious."
"mass_thermo" variable		Specify the "mass" of the thermostat when the "control_velocity" is set to "on."
"target_KE" variable		Specify the target "temperature" of the fictitious particle when the "control_velocity" is set to "on."

- Configure the bias potential

The bias potential can be configured by the "bias_potential" block, definable under the "meta_dynamics" block. A typical example is as follows.

```
bias_potential{
  height = 0.02 eV
  update_frequency=20
  output_frequency=100
}
```

- Variables definable under the "bias_potential" block include the following.

"height" variable Specify the "height" of the bias potential, update in units of energy, to be added at each bias potential. Note that the "width" of the bias potential is a quantity specific to each collective variable; therefore, it is to be specified under the "collective_variable*xx*" block described above.

"output_frequency" Specify the frequency of the bias potential output. For example, if 100 is specified, the

variable	bias potential will be written to file once every 100 bias potential updates. This variable will be interpreted only when "meta_dynamics_type" is "bias_generation."
"update_frequency" variable	Specify the frequency of bias potential updates. For example, if 20 is specified for this variable, the bias potential will be updated once every 20 MD steps. The default value is 20.

- Configure replica–parallelization-related variables

➢ Resolution of the initial atomic coordinates and velocities when replica parallelization is enabled

When replica parallelization is enabled, the initial coordinates are all the same, and only the initial velocities differ among the replicas if no special configurations are performed. Since each replica starts from a different point in a phase space, each replica will follow a distinct trajectory, despite the fact that their atomic coordinates were all the same at the beginning of the simulation. Note that at early stages of the simulation, atomic coordinates of each replica are obviously very similar.

➢ Method to change the initial atomic coordinates per rank

It is possible to specify the atomic coordinates per replica in the input parameter file. This can be done by defining the atomsxx block (where xx is the MPI rank) and specifying the coordinates under that block. For example, to specify different initial atomic coordinates for rank0 replica and rank1 replica, insert the following in the input parameter file.

```
structure{
    atom_list{
        ....
        atoms0{
            #units angstrom
            #default weight = 1, element = Si, mobile = 1
            #tag element rx ry rz mobile weight
            C 5.0157363043        5.6563796505        5.8043454319 1 1
            C 4.7499007526        4.2727134018        5.7364572058 1 1
            ...
            ...
        }
        atoms1{
            #units angstrom
            #default weight = 1, element = Si, mobile = 1
            #tag element rx ry rz mobile weight
            C       4.5897384578    5.5998560107    5.7723226564 1 1
            C       5.1658344359    4.3217914066    5.6857269157 1 1
            ...
            ...
        }
    }
}
}
```

5.4.3.3 Execution

To perform metadynamics simulations, the following command is issued as in standard PHASE calculations.

```
mpirun -n NP phase ne=NE nk=NK nr=NR
```

Here `NP` is the number of MPI processes, `NE` is the number of band parallelizations, `NK` is the number of ***k*** point parallelizations, and `NR` is the number of replicas to be handled in parallel. The relation `NP = NE x NK x NR` must hold. The default values are ne=NP, nk=1, and nr=1, but it is strongly

recommended to explicitly specify each parameter.

Usually, only the most recent bias potential will be output when metadynamics is performed. It is possible to rebuild the bias potential and output it to a file. To use this feature, specify "bias_generation" for the "meta_dynamics_type" variable defined under the "meta_dynamics" block. In this case, the "bias_output_frequency" variable definable under the "bias_potential" block is used to specify the frequency of output. For example, if 10 is specified for "bias_output_frequency" and the total number of bias updates is 100, then the bias potential at the 10th, 20th, 30th, … , 100th updates will be obtained, with each written to a separate file. The file name will be "bias_potential_dataxx," where xx is the number of bias potential updates. After the above configuration is done, run PHASE under the directory where the metadynamics simulation was performed. This operation only reads the collective variable from the file and builds the bias potential; thus, it is not necessary to run PHASE in parallel.

5.4.3.4 Output of the results

When a metadynamics simulation is performed, extra files will be obtained in comparison with standard PHASE calculations. We now describe the files specific to a metadynamics simulation.

- 'curr_bias_potential.data' file

This file records the current bias potential. The format of the file is as follows.

```
      1.2000000000      -3.1400000000       0.0000000000
      1.3000000000      -3.1400000000       0.0000000000
      1.4000000000      -3.1400000000       0.0000000000
      1.5000000000      -3.1400000000       0.0000000000
      1.6000000000      -3.1400000000       0.0000000000
      1.7000000000      -3.1400000000       0.0000000000
                          ....
                          ....

      1.2000000000      -3.0400000000       0.0000000000
      1.3000000000      -3.0400000000       0.0000000000
      1.4000000000      -3.0400000000       0.0000000000
      1.5000000000      -3.0400000000       0.0000000000
      1.6000000000      -3.0400000000       0.0000000000
      1.7000000000      -3.0400000000       0.0000000000
                          ....
                          ....
```

Each line corresponds to a "collective variable set." First, all collective variables defined are recorded and then the value of the corresponding bias potential is recorded.

- "bias_potential.dataxx" file

This file records the bias potential at bias potential update number xx. This file is obtained when "meta_dynamics_type" is set to "bias_only." The file format is exactly the same as that for the "curr_bias_potential.data" file.

- 'nfdynm.data_at_bias' file

This file records the atomic coordinates at bias potential updates. The file format is the same as that for the F_DYNM file, the standard coordinate data file format of PHASE.

- 'nfefn.data_at_bias' file

This file records the total energy at bias potential updates. The file format is the same as that for the F_ENF file, the standard energy data file format of PHASE.

- 'collective_variables.data' file

This file records the value of the collective variable at bias potential updates. The file format is as follows.

```
       2        1.6399047278         0.0906233310
       3        1.6933783940         0.2327954221
       4        1.6487636847         0.0655806009
       5        1.7510381463        -0.1403803460
       6        1.7880912692        -0.2122517967
       7        1.7558411086        -0.2557274737
       8        1.7939362867        -0.0296094373
       9        1.7595919709         0.1959354384
      10        1.7773637731         0.3761827029
      11        1.7657919080         0.3998392061
      12        1.7604309483        -0.0107912799
      13        1.6218441177        -0.3366407543
                         ....
                         ....
```

Each line corresponds to a bias potential update. The first column gives the number of bias potential updates, and the second column and others contain values of the collective variables that are recorded in the order defined in the input parameter file.

- 'bias_potential_parameters.data' file

This file records the bias potential parameters. This file is needed when bias potential parameters are changed on restart calculations, since there is no way of resolving the parameters in the previous run without the information recorded in this file. The file format is as follows.

```
       2        0.0200000000         0.1000000000         0.1000000000
       3        0.0200000000         0.1000000000         0.1000000000
       4        0.0200000000         0.1000000000         0.1000000000
       5        0.0200000000         0.1000000000         0.1000000000
                         ....
                         ....
```

Each line corresponds to a bias potential update. The first column gives the number of the bias potential update, the second column contains the value of w in eq. (23), and the third column and others correspond to the value of δs_α for each collective variable.

5.4.3.5 Example calculation: energy surface of hydrocarbon molecules

29. Outline

To illustrate use of the metadynamics method, we explore the energy surface for the hydrocarbon molecule C_4H_6. The C_4H_6 molecule has three stable molecular structures—trans-1,3-butadiene, cis-1,3-butadiene, and cyclobutene. Cyclobutene is a cyclic molecule. trans-1,3-butadiene is planar, while cis-1,3-butadiene is not; the stable structure for cis-1,3-butadiene is a structure in which the dihedral angle is twisted by about 30° (the so-called gauche conformation). The molecular structures for these molecules are depicted in Figure 5.21. Cyclobutene has the highest energy, followed by cis-1,3-butadiene, and trans-1,3-butadiene has the lowest energy. Possible reactions of the molecule are the "electric cyclic" reaction, in which the closing (opening) of the ring for 1-3 butadiene (cyclobutene) leads to cyclobutene (1-3 butadiene) or a cis–trans transformation between the two 1-3 butadienes. Since the electrocyclic reaction leads to the breaking a chemical bond, the activation barrier is expected to be high, on the order of 1 eV. However, the cis–trans transformation should have a lower activation barrier, on the order of 100 meV. Note that this problem is difficult to analyze by classical force fields because the electronic structures of the two 1,3-butadienes and cyclobutene are completely different (for example, the number of double bonds differ). We confirm that PHASE can correctly

describe the properties of this molecule.

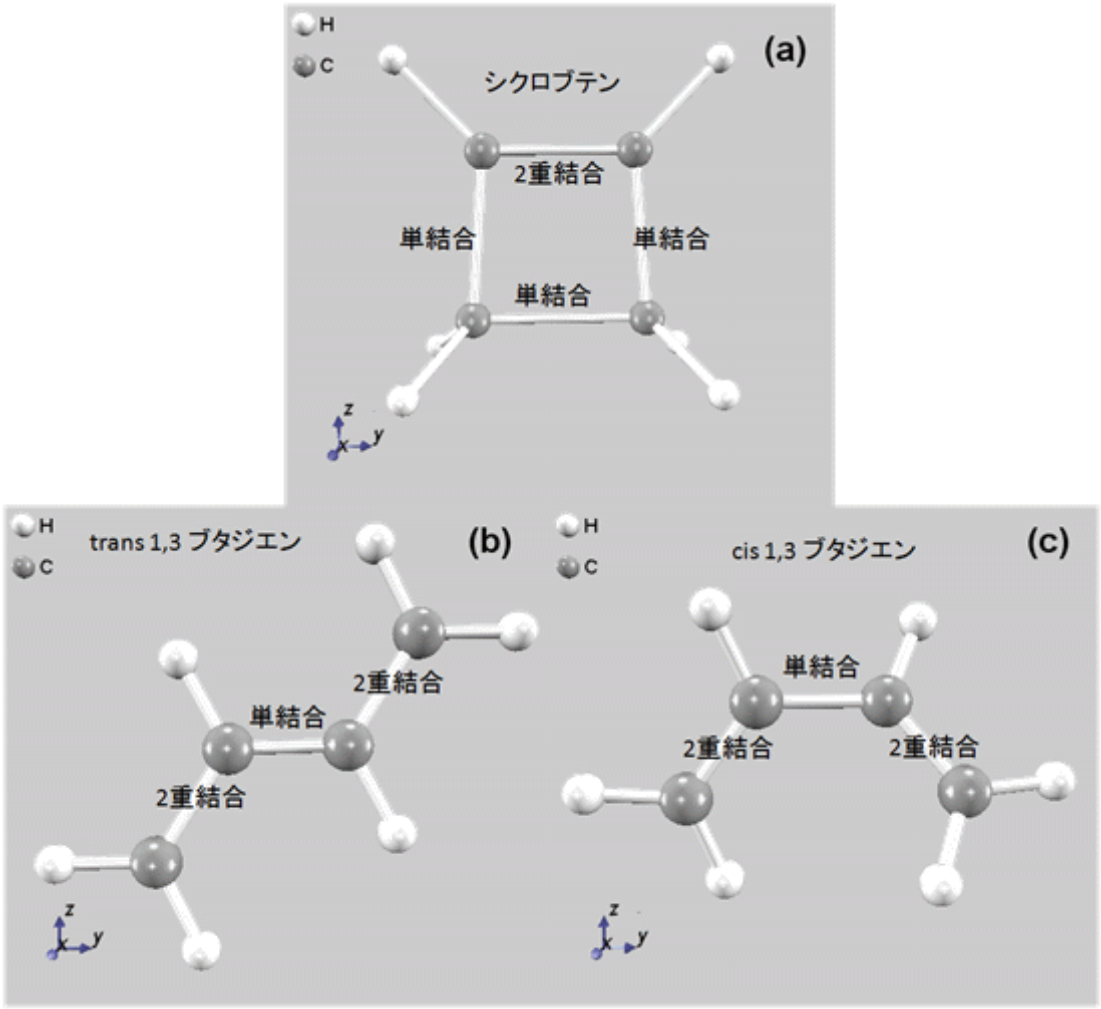


Figure 5.21 Molecular structure of the C_4H_6 molecule.

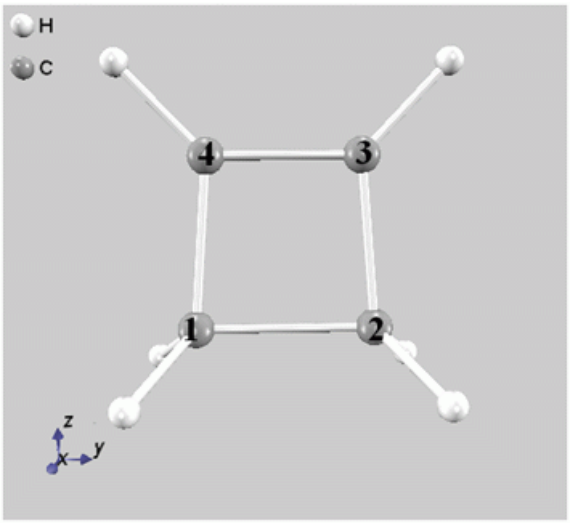


Figure 5.22 Molecular structure of the cyclobutene molecule

First, the metadynamics method is enabled. This is done by specifying "meta_dynamics" for the "driver" variable under the "condition" block.

```
condition{
    driver = meta_dynamics
    ....
}
```

Next, the collective variables are defined. In this example, the following collective variables are adopted.

1. The distance between atoms 1 and 2 shown in Figure 5.22. Values for the variables "ds" and "delta_s_s" are 1 Å and 0.05 Å, respectively.
2. The dihedral angle formed by atoms 1, 4, 3, and 2 shown in Figure 5.22. Values for the variables "ds" and "delta_s" are 10°and 5°, respectively.

The above configuration can be realized by the following.

```
meta_dynamics{
....
....
  collective_variable1{
     type=bond_length
     atom1=5
     atom2=4
     delta_s=0.05 angstrom
!for bpot output
     smin=1 angstrom
     smax=5 angstrom
     ds = 0.1 angstrom
   }
   collective_variable2{
     type=dihedral_angle
     atom1=5
     atom2=3
     atom3=2
     atom4=4
     delta_s = 5 degree
!for bpot output
     smin = -180 degree
     smax = +180 degree
     ds = 10 degree
   }
}
```

For the bias potential, the height is set to 0.02 eV (0.46 kcal/mol), and it is set to be updated once every 20 MD steps. This configuration is done by the variables "height" and "update_frequency" under the "bias_potential" block under the "meta_dynamics" block.

```
meta_dynamics{
    ....
    ....
    bias_potential{
        update_frequency = 20
        height=0.02 eV
    }
}
```

The number of bias potential updates is arbitrary, but to obtain a reliable free-energy surface, we need an ample number of updates.

31. Results

We present the results obtained from this simulation. Figure 5.23 shows the contour plot of the energy surface obtained after 18,140 bias potential updates.

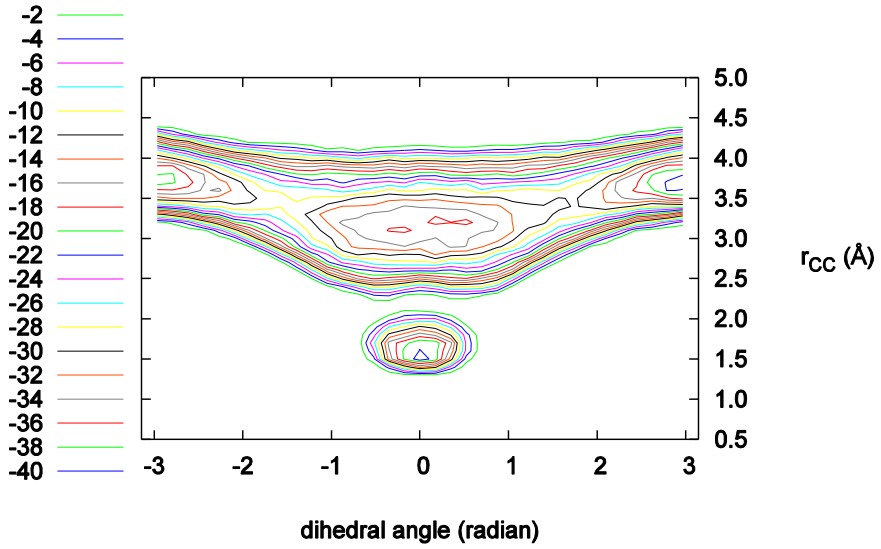


Figure 5.23 Free-energy surface of the C_4H_6 molecule at 300 K.

From this figure, four stable points are found in the energy contour: the point at which (i) the atomic distance is about 1.5 Å and the dihedral angle is 0 radian, (ii) the atomic distance is about 3.3 Å and the dihedral angle is around 0 radian, and (iii, iv) the atomic distance is about 3.7 Å and the dihedral angle is around ± 3 radians. These stable points correspond to cyclobutene, cis-1,3-butadiene, and trans-1,3-butadine, respectively. In a calculation at absolute zero, the gauche conformation instead of the cis conformation is stable, but in the metadynamics simulation at 300 K, the two conformations were not clearly resolved. The difference in energy between cyclobutene and trans-1,3-butadiene is about 16 kcal/mol, while the diffference in energy between cyclobutene and cis butadiene is about 12 kcal/mol. These values are larger than those obtained from calculations at absolute zero.

Figure 5.23 and Figure 5.24 show the variation of collective variables-dihedral angle and carbon-carbon distance-against the number of bias potential updates, respectively. The simulation starts from cyclobutene (carbon–carbon distance of about 1.5 Å, and dihedral angle of about 0 radian). After about 700 bias potential updates, the system overcomes the saddle point, and the molecule transforms to butadiene. Then, from this point to about 18,000 bias potential updates, a wide exploration of the energy surface occurs. From Figure 5.23, it is understood that besides cyclobutene, the range of the dihedral angle is significantly broad. Thus, many bias potential updates are required to fill the potential valley. Finally, after about 18,000 bias potential updates, the system returned to cyclobutene; thus, we terminated the simulation at that point.

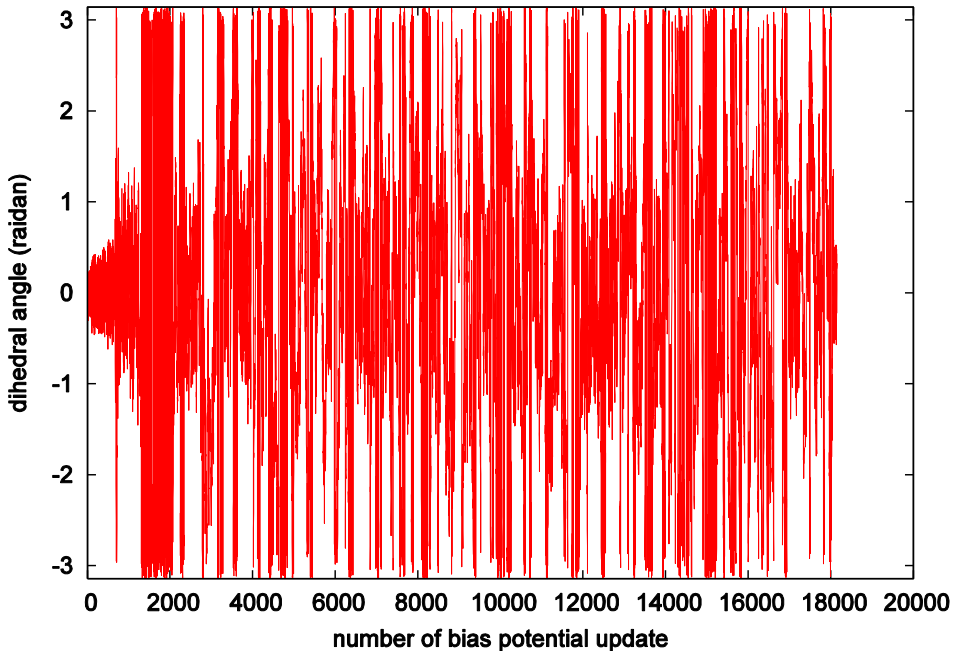


Figure 5.24 Variations in the dihedral angle with the number of bias potential updates.

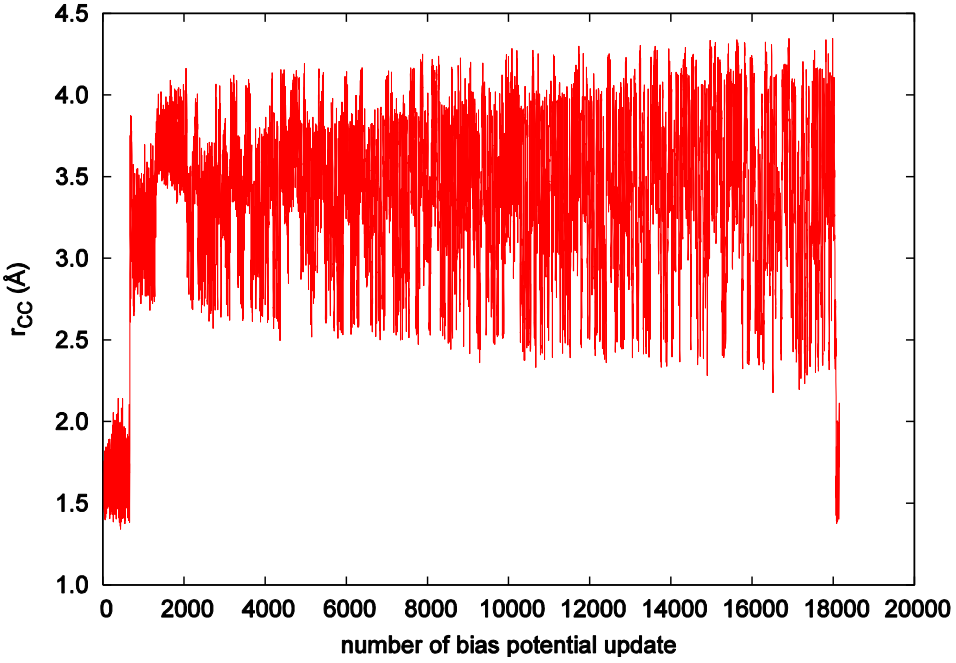


Figure 5.25 Variations in the carbon–carbon distance with the number of bias potential updates.

In Figure 5.26 (a)–(d), snapshots of atomic configurations obtained during the metadynamics simulation are shown. From these figures, we see that, because of the effect of the bias potential, various molecular structures are obtained during the metadynamics simulation.

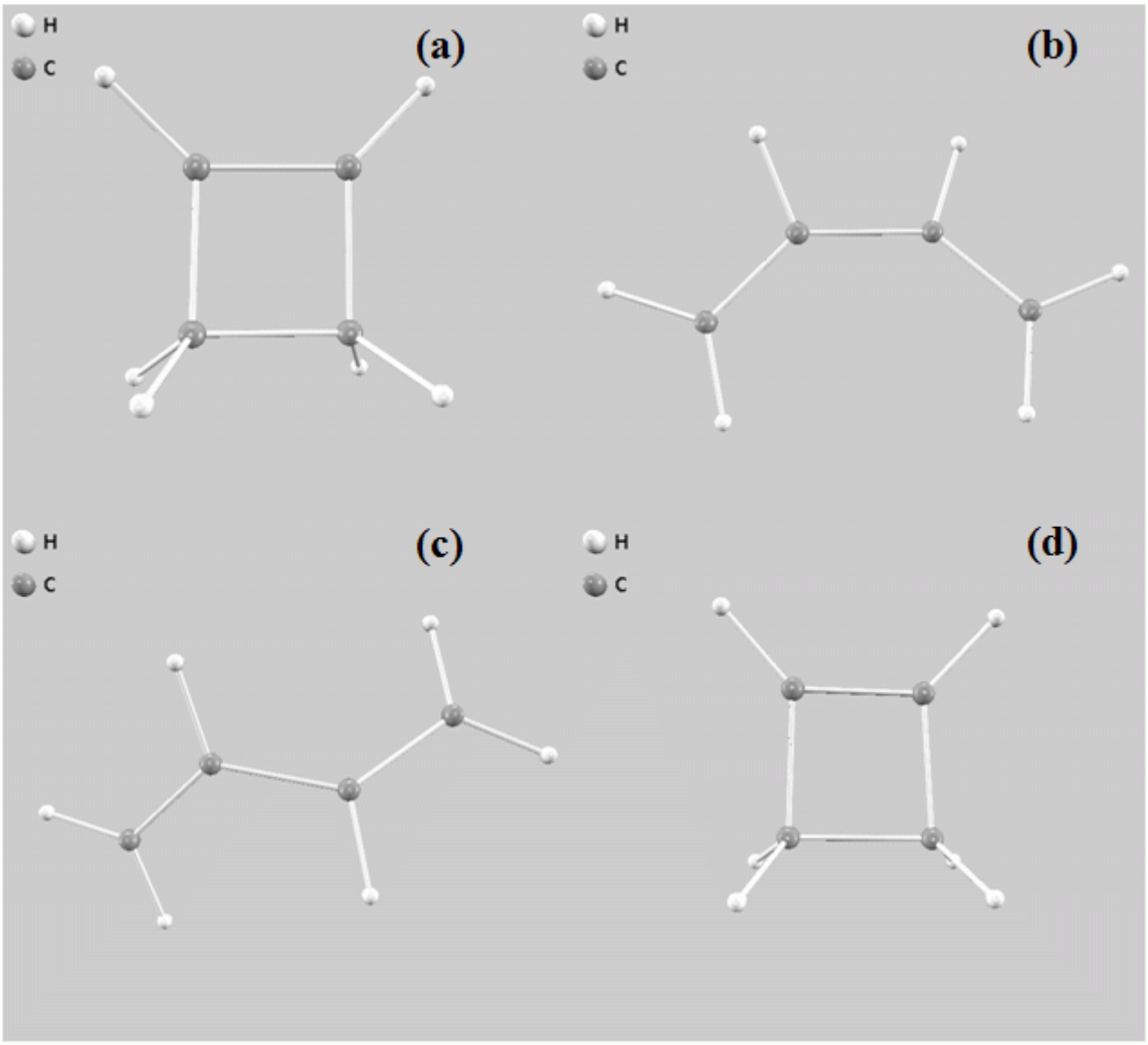


Figure 5.26 Snapshots of atomic configurations obtained from the metadynamics simulation. Molecular structure obtained after (a) 2 bias potential updates, (b) 690 bias potential updates, (c) 1,500 bias potential updates, (d) 18,070 bias potential updates.

5.4.3.6 Notes

The metadynamics method can be used in conjunction with all pseudopotentials. It is possible to perform calculations in parallel, including replica parallelization. To obtain meaningful results, a large calculation burden is required. When restart calculations are performed with replica parallelization enabled, it is possible that the corresponding restart files do not exist for a certain replica. In this case, the restart file for the neighboring replica is read, and the initial replica is built according to the specifications defined under the "continuation_strategy" block.

5.5 Time-dependent density functional theory (TDDFT) calculations

5.5.1 Optical spectrum calculations of molecules by real-time TDDFT (RT-TDDFT)

5.5.1.1 Calculation methods

Based on RT-TDDFT, electron dynamics simulations are performed by solving time-dependent one-electron equations for given initial one-electron wave functions,

$$i\hbar\frac{\partial}{\partial t}\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t) = H(t)\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t)$$

Here $\phi_n^{\boldsymbol{k}}$ is the one-electron wave function for wave vector $\boldsymbol{k}$ and band index n, and H is the one-electron Hamiltonian. Time evolutions of one-electron wave functions are formally written as

$$\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t+\Delta t) = \exp\left(-\frac{i}{\hbar}\int_t^{t+\Delta t} dt' H(t')\right)\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t)$$

An efficient numerical computation requires approximations for the time integral and for the exponential parts. Many approximations are possible, but this code uses the simplest one. If the time step Δt is sufficiently small, the time integral can be approximated by

$$\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t+\Delta t) \cong \exp\left(-\frac{i}{\hbar}\Delta t H(t)\right)\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t)$$

The exponential can be approximated by a Taylor expansion,

$$\exp\left(-\frac{i}{\hbar}\Delta t H(t)\right) = \sum_{N=0}^{\infty}\frac{1}{N!}\left(-\frac{i}{\hbar}\Delta t H(t)\right)^N$$

Input parameters, which need to be set carefully to balance accuracy against computation time, are the time step Δt and the number of terms in the Taylor expansion $N_{\max}$.

Because one-electron wave functions at time $t = 0^-$ are prepared by ground-state wave functions, we need to perform DFT ground-state calculations before starting RT-TDDFT calculations. The initial wave function at $t = 0^+$ is generated by

$$\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t=0^+) = e^{-i\varepsilon\boldsymbol{q}\cdot\boldsymbol{r}}\phi_n^{\boldsymbol{k}}(\boldsymbol{r},t=0^-)$$

This is equivalent to applying an impulsive electric field for a molecule at t = 0. During RT-TDDFT calculations, dipole moments or current densities [$\mathbf{d}$(t) or $\mathbf{J}$(t)] are calculated at each time step, and after RT-TDDFT simulations, optical spectra can be obtained by Fourier transformation of $\mathbf{d}$(t) or $\mathbf{J}$(t).

5.5.1.2 Input parameters

An input example is shown below. RT-TDDFT simulations will start only if the DFT ground-state calculation has converged.

```
postprocessing{
  rttddft{
    sw_rttddft = on
    time_step_delta = 0.1
    time_step_max = 1000
    ext_pulse_epsilon = 0.01, ext_pulse_kx = 1.0, ext_pulse_ky = 0.0, ext_pulse_kz =
0.0
  }
}
```

Input parameter	Default value	Description
sw_rttddft	OFF	ON or OFF
time_step_delta	0.1 (in atomic unit)	Time interval (Total simulation time equals to time_step_delta*time_step_max.)
time_step_max	100	Number of time step

For generating initial states by applying an impulsive electric field $\boldsymbol{E}(t) = E_0 \boldsymbol{e} \delta(t)$, the input parameters are as follows:

Input parameter	Default value	Description
ext_pulse_epsilon	0.0d0	Field magnitude E_0
ext_pulse_kx	0.0d0	Field x-direction $\boldsymbol{e}$
ext_pulse_ky	0.0d0	Field y-direction $\boldsymbol{e}$
ext_pulse_kz	0.0d0	Field z-direction $\boldsymbol{e}$

5.5.1.3 Notes

- Use only norm-conserving pseudopotentials for RT-TDDFT calculations. We cannot use ultra-soft-type pseudopotentials.
- Set molecular positions to the middle of unit cells. Do not set molecule positions across unit-cell boundaries.
- Set "base_reduction_for_GAMMA = off" and "base_symmetrization_for_GAMMA = off" in the "ksampling{}" tag.
- Set "method = manual" and "sw_inversion = off" in the "symmetry{}" tag.
- This code does not support bulk-system simulations.
- The current version of our code is limited to systems having fixed-ion positions.

5.6 Structure optimization

5.6.1 Optimizing a unit cell by using the stress tensor

5.6.1.1 Input parameters

First, prepare an input parameter file as usual. If parameters for structure optimization are specified as usual, atomic positions are also optimized at every step after optimizing the unit cell. Add the **lattice** block shown below to optimize the unit cell.

```
structure_evolution{
  lattice{
    sw_optimize_lattice = on
  }
}
```

Optimization of the unit cell is enabled by setting the variable **sw_optimize_lattice** to "on." The following variables can be defined in the **lattice** block.

sw_optimize_lattice	If on, the unit cell of the system will be optimized. Defaults to off. Note that if this switch is on, **sw_stress** is also automatically set to on.
sw_uniform	If on, the three axes of the unit cell are uniformly varied. In that case, the volume of the unit cell is changed by the average value of the diagonal elements of the stress tensor. Defaults to off.
method	Specify a method of optimization. Options are bfgs, quench, and sd. Defaults to bfgs.
delta_stress	When the method is quench or sd, this variable sets the step size for updating. Defaults to 1.
max_stress	Specify the convergence criterion for the maximum stress. Defaults to $1.e{-6}$ Hartree/$Bohr^3$. If **sw_uniform** is set to on, this criterion applies to the average value of the diagonal elements of the stress tensor.
sw_optimize_coordinates_once	If on, optimization of atomic position is performed at the first step only.

As described below, the stress tensor will not converge when the cutoff energy is not sufficiently large (i.e., a small cutoff energy does not give accurate results). If minima in the stress and total energy are inconsistent, the cutoff energy may not be sufficiently large.

5.6.1.2 Calculation results

Calculation results are dumped into **output000**, **nfefn.data**, and **nfdynm.data**. The stress tensor is printed into **output000**. The values can be extracted by the following command.

```
% grep -A3 'STRESS TENSOR$' output000
```

```
  STRESS TENSOR
       0.0002326236       0.0000000000       0.0000000000
       0.0000000000       0.0002326236       0.0000000000
       0.0000000000       0.0000000000       0.0002142790
--
  STRESS TENSOR
       0.0002272841       0.0000000000       0.0000000000
       0.0000000000       0.0002272841       0.0000000000
       0.0000000000       0.0000000000       0.0002077216
```

```
--
       ……..
       ……..
```

Although the stress tensor is normally printed only once, optimization of the unit cell outputs the history of the stress tensor calculation.

Maximum value of stress tensor (or averaged value of diagonal elements if **sw_uniform**=on) for each step is also printed into **nfefn.data** file in addition to total energy and maximum atomic force etc. The example is shown below:

```
iter_unitcell, iter_ion, iter_total, etotal, forcmx, stressmx
    1   1     18    -181.4043211413       0.0020128619
    1   2     27    -181.4043355689       0.0015666906
    1   3     36    -181.4043464493       0.0011267018
    1   4     44    -181.4043509953       0.0008837770
    1   5     53    -181.4043582176       0.0000137026       0.0002326236
    2   1     73    -181.4044226903       0.0000645338       0.0002272841
                      ……..
                      ……..
```

The **nfdynm.data** file is almost the same as usual, but the header, which is normally printed only once, is printed at every step when the cell vectors are changed.

```
#
#   a_vector =        8.6795114819       0.0000000000       0.0000000000
#   b_vector =        0.0000000000       8.6795114819       0.0000000000
#   c_vector =        0.0000000000       0.0000000000       5.5916992108
#   ntyp =       2 natm =       6
# (natm->type)    2   2   1   1   1   1
# (speciesname)    1 :  O
# (speciesname)    2 :  Ti
#
 cps and forc at (iter_ion, iter_total =    1     18 )
   1   0.000000000   0.000000000   0.000000000   0.000000   0.000000   0.000000
   2   4.339755741   4.339755741   2.795849605   0.000000   0.000000   0.000000
   3   2.643779197   2.643779197   0.000000000  -0.001423  -0.001423   0.000000
   4   6.983534938   1.695976544   2.795849605  -0.001423   0.001423   0.000000
                      ……
                      ……
#
#   a_vector =        8.7672856463       0.0000000000       0.0000000000
#   b_vector =        0.0000000000       8.7672856463       0.0000000000
#   c_vector =        0.0000000000       0.0000000000       5.6429940606
#   ntyp =       2 natm =       6
# (natm->type)    2   2   1   1   1   1
# (speciesname)    1 :  O
# (speciesname)    2 :  Ti
#
 cps and forc at (iter_ion, iter_total =    1    111 )
   1   0.000000000   0.000000000   0.000000000   0.000000   0.000000   0.000000
   2   4.383642823   4.383642823   2.821497030   0.000000   0.000000   0.000000
   3   2.663907294   2.663907294   0.000000000   0.001773   0.001773   0.000000
   4   7.047550117   1.719735530   2.821497030   0.001773  -0.001773   0.000000
   5   1.719735530   7.047550117   2.821497030  -0.001773   0.001773   0.000000
   6  -2.663907294  -2.663907294   0.000000000  -0.001773  -0.001773   0.000000
                      ……
                      ……
```

5.6.1.3 Examples: rutile type TiO_2

In the input parameter file

- The cutoff energy was set to 80 Rydberg.

- The pseudopotential files Ti_ggapbe_us_02.pp and O_ggapbe_us_02.pp, which can be downloaded from the website of CISS, were used.
- In optimizing atomic positions, the BFGS method was employed. Convergence criterion for the maximum force was set to 2e−4.
- Initial atomic positions and lattice parameters for rutile-type TiO_2 were downloaded from the Inorganic Material Database AtomWork (http://crystdb.nims.go.jp/).
- Default settings were used for the wavefunction solver and the charge-density mixer.

The adopted cutoff energy, 80 Rydberg, is relatively large, but TiO_2 requires such a large cutoff energy, as discussed latter.

The following shows the content of the **nfefn.data** file.

```
iter_unitcell, iter_ion, iter_total, etotal, forcmx, stressmx
   1    1     18    -181.4043211413      0.0020128619
   1    2     27    -181.4043355689      0.0015666906
   1    3     36    -181.4043464493      0.0011267018
   1    4     44    -181.4043509953      0.0008837770
   1    5     53    -181.4043582176      0.0000137026      0.0002326236
   2    1     73    -181.4044226903      0.0000645338      0.0002272841
   3    1     92    -181.4044839579      0.0001241955      0.0002222588
   4    1    111    -181.4056948858      0.0025074070      0.0002222588
   4    2    120    -181.4057176163      0.0020195652      0.0002222588
   4    3    130    -181.4057600852      0.0000156213      0.0000444895
                        ......
                        ......
   9    1    248    -181.4058191217      0.0001647915      0.0000332105
  10    1    268    -181.4058328662      0.0000709369      0.0000119789
  11    1    287    -181.4058349707      0.0000268520      0.0000015502
  12    1    306    -181.4058351835      0.0000244918      0.0000006790
```

In the above example, atomic positions were optimized five times. Meanwhile, the stress tensor was not calculated, and hence, the sixth column of these steps is empty. In the fifth step, the maximum atomic force become smaller than the threshold, and then the stress tensor was calculated, and cell vectors were changed. At that time, the number in the first column, which represents the number of steps for optimizing the unit cell, increased to 2, and the maximum value of the stress tensor was printed in the sixth column. Although optimization of atomic positions was not performed in the second and third steps because the calculated atomic force was smaller than the threshold, optimization was carried out in the fourth step. In this manner, optimizations of atomic positions and the unit cell were alternately performed, until finally the maximum value of the stress tensor converged in the 12th iteration. The following figure shows the convergence progress for optimizing the unit cell.

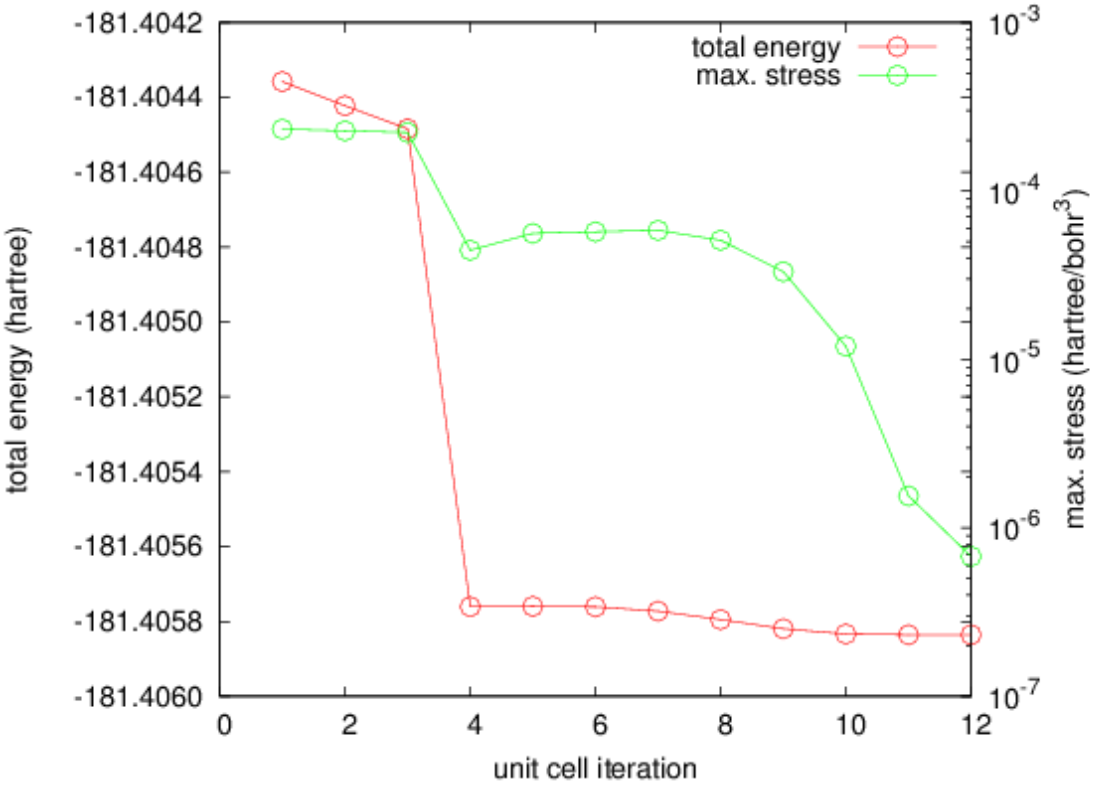


Figure 5.27 Convergence progress for optimizing a unit cell. Total energy (red) and maximum value of stress tensor (green) are displayed.

Optimized lattice constants can be obtained from the cell vectors in the final step printed in the **nfdynm.data** file. In this example, $a = 8.7934$ Bohr and $c = 5.6164$ Bohr are the optimized lattice constants.

Stress tensor and cutoff energy

Compared to the total energy or atomic force, the stress tensor does not converge well when a small cutoff is used for the energy. The following figure shows the relationship between stress tensor and cutoff energy for rutile-type TiO_2. Here lattice parameters are taken from experimental data.

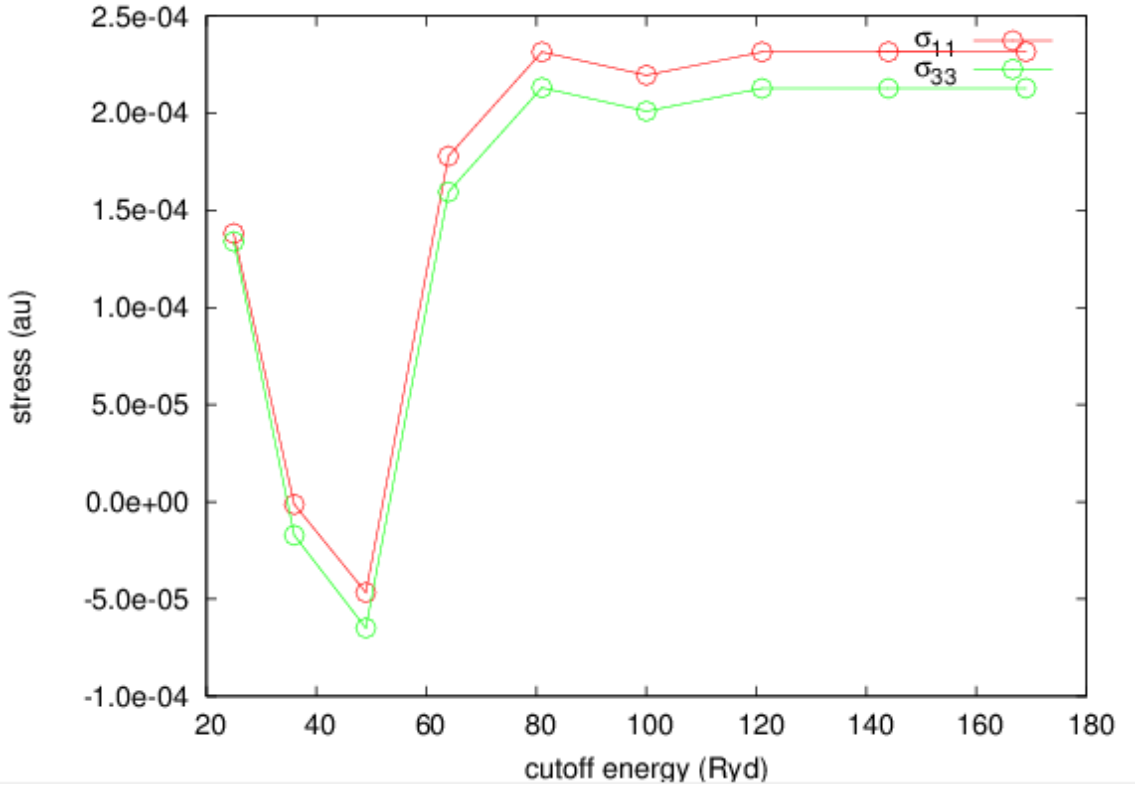


Figure 5.28 Relationship between stress tensor and cutoff energy for rutile-type TiO_2

As shown in this figure, a negative value was obtained for the stress tensor when the cutoff energy was around 50 Rydberg. The figure shows that the cutoff energy should be larger than 80 Rydberg to obtain an adequate value for the stress tensor of TiO_2.

6. Calculation by the PAW method

6.1 Overview

The projector-augmented wave (PAW) method is strongly related to the ultrasoft pseudopotential (USPP) method. However, compared with the USPP method, the PAW method is superior in terms of accuracy, particularly when spin polarization is considered, or when a large charge transfer is expected to occur. Here we describe how to perform calculations using the PAW method in PHASE.

6.2 How to use the PAW method

To perform calculations based on the PAW method, we use the corresponding PAW potential files. The PAW potential files reside in the pseudopotential directory by the following name.

```
elementname_ggapbe_paw_xxx.pp
```

The potential files are specified by the F_POT identifier in the "file_names.data" file, in the same way as USPP and norm-conserving pseudopotential files..

PAW potentials can be used for both PAW and non-PAW calculations. To perform non-PAW calculations using PAW potentials, specify the following in the input parameter file.

```
accuracy{
    paw = off
}
```

If the above specification is not present (or if "paw" is set to "on"), calculations with the PAW method will be performed.

When using the PAW method, it is possible to improve convergence by changing the way the deficit charge is handled. In many cases, convergence can be improved by including the following in the input parameter file.

```
charge_mixing{
    ...
    sw_mix_charge_hardpart = on
}
```

When performing fixed-charge calculations using the "ekcal" program, it is necessary to specify a data file specific to the PAW method. This is done by the F_CNTN_BIN_PAW identifier in the "file_names.data" file. For example, if the directory in which the corresponding SCF calculation was performed is the parent directory, the following line must be added in the "file_names.data" file.

```
&fnames
...
...
F_CNTN_BIN_PAW='../continue_bin_paw.data'
/
```

6.3 Example

To illustrate the PAW method, we calculate the lattice constant and bulk modulus for bcc chromium. When the USPP method is employed for this problem, the lattice constant is overestimated, while the bulk modulus is underestimated.
The input data for this example can be found under samples/Cr. Under this directory, the following files/directories exist.

```
Cr_ggapbe_paw_002.gncpp2
Cr_ggapbe_us_02.pp
paw/
us/
```

The PAW potential file is "Cr_ggapbe_paw_002.gncpp2," while the USPP file is "Cr_ggapbe_us_02.pp." Input files for PAW calculations are under the directory "paw," while those for USPP calculations are under the directory "us." Under both "paw" and "us" directories, the following file/subdirectories exist.

```
catenergy.sh
vol20/
vol21/
.....
```

The simple shell script "catenergy.sh" concatenates the energy data files after all calculations are done. Under directories "vol20," "vol21," …, input files corresponding to unit-cell volumes of 20Å^3, 21Å^3, …, respectively, exist. After calculations for all directories are completed, you can run the "catenergy.sh" script and get the "energy.data" file in which energies are recorded for all volumes that were considered in the calculation.

The following lines can be found in the "file_names.data" file under each subdirectory of the "paw" directory.

```
 F_POT(1) = '../../Cr_ggapbe_paw_002.gncpp2'
 F_POT(2) = '../../Cr_ggapbe_paw_002.gncpp2'
```

Because of these specifications, the PAW potential file is used for the calculations. The "file_names.data" file under the "us" directory points to the "Cr_ggapbe_us_02.pp file" in the same manner.

The *EV* curves obtained for bcc chromium are depicted inFigure 6.1. Obviously, the results from the PAW and USPP methods differ completely. In Table 6.1, we tabulate the lattice constants and bulk moduli obtained from the two *EV* curves. Clearly, there is a better agreement with experiment for the PAW method.

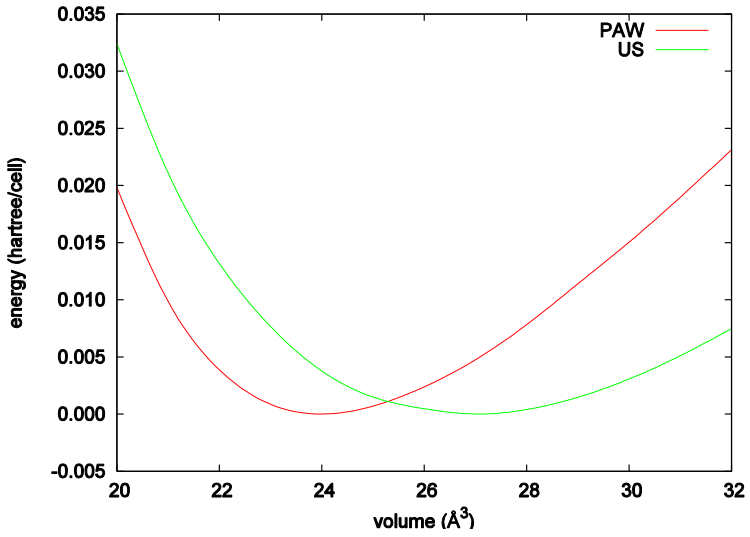


Figure 6.1 *EV* curves for bcc chromium. Red curve: PAW method, green curve: USPP method.

Table 6.1 Lattice constants, bulk moduli, and cohesive energies for bcc chromium.

	US	PAW	experiments
a (Å)	2.994	2.886	2.88
B (GPa)	89.2	150.5	190.1
Ecoh (eV/atom)	4.01	3.065	4.10

6.4 Supported features

Features that can be used with PAW include the following.

- total energy
- symmetry
- spin polarization
- structural optimization
- output of charge densities
- calculation of various densities of states
- band structure
- stress tensor
- work function
- phonon
- molecular dynamics
- DFT+U
- ESM method
- constrained dynamics
- meta dynamics
- NEB
- unit cell optimization
- non-collinear magnetization
- spin-orbit coupling
- features provided by UVSOR-Epsilon
- features provided by UVSOR-Berry-Phonon

7. Appendix

7.1 Calculation accuracy

7.1.1 Cutoff energy

In DFT calculations using a plane-wave basis set, a larger cutoff energy always leads to a more exact total energy. As an example, Figure 7.1 shows the relationship between cutoff energy and total energy for a face-centered cubic aluminum crystal.

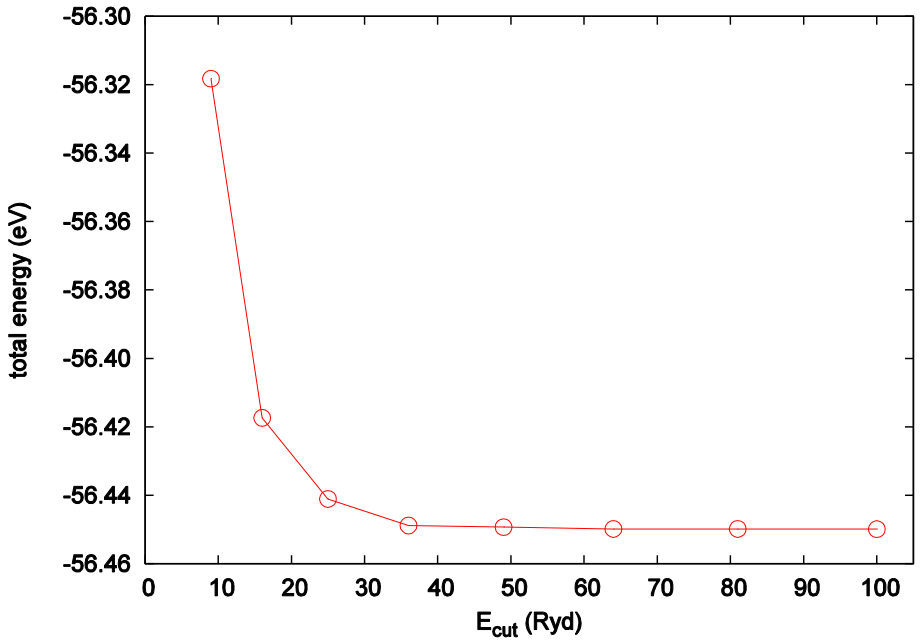


Figure 7.1 Relationship between cutoff energy and total energy for a face-centered cubic aluminum crystal

This figure clearly shows that a larger cutoff energy gives a lower total energy, and the total energy converges to certain value. This behavior depends on the pseudopotentials that are used. In this example, the total energy converges in the range of about 1 meV when the cutoff energy is 36 Rydberg. Although the required accuracy depends on the purpose of the calculation, normally it is sufficient if the total energy converges in the range of about 10 meV. Moreover, if you focus on the relative energy rather than absolute energy, a much smaller cutoff energy may be sufficient.

7.1.2 k-point sampling

Since PHASE employs plane-wave functions, it can only treat periodic systems. Therefore, physical quantities need to be integrated over the first Brillouin Zone (BZ). The k-point sampling determines the resolution of the integration in k-space taken over the entire volume of the first BZ. Figure 7.2 shows the relationship between total energy and the number of k-points in the irreducible BZ for an aluminum crystal.

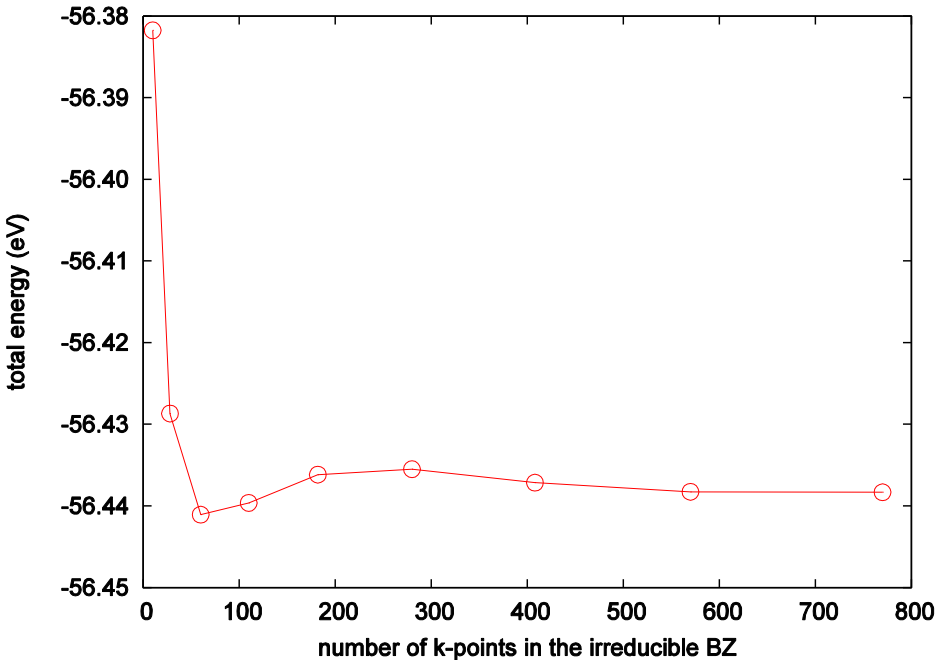


Figure 7.2 Relationship between total energy and the number of k-points for a face-centered cubic aluminum crystal

Note that the total energy does not monotonically decrease because the variational principle is not satisfied on increasing the number of k-points. In the above example, the total energy converges after passing through a minimum. As in the case of the cutoff energy, if you focus on relative energy rather than absolute energy, a much smaller number of k-points may be sufficient.

7.1.3 Convergence criterion

Strict convergence criteria for SCF calculations enable us to calculate atomic forces more accurately. In normal structure optimizations, 10^{-8} Hartree is usually sufficient for the SCF convergence threshold. However, MD simulations require a smaller convergence threshold to conserve total energy or temperature. Figure 7.3 shows the relationship between the convergence criterion and the maximum atomic force for a SiO_2 crystal. This figure indicates that a strict convergence criterion, less than 10^{-10} Hartree, is required for the atomic force to satisfactorily converge.

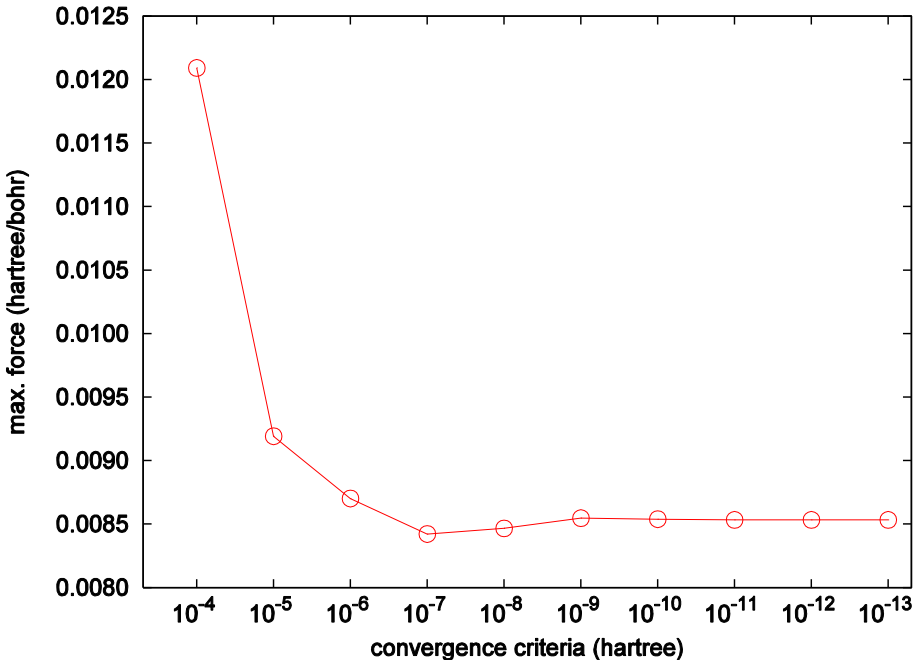


Figure 7.3 Relationship between convergence criterion and maximum atomic force for SiO_2

7.1.4 Benchmark calculation (comparison of wavefunction solver)

7.1.4.1 FCC-Cu

Here we examine how well each wavefunction solver performs for FCC-Cu. The sample input data introduced here are in the directory **samples/sol_cmix_test/Cu**.

32. Input data

Input data, excluding the **wavefunction_solver** block, are shown below.

```
Control{
    condition = initial
    cpumax = 1 day
}
accuracy{
    cutoff_wf =  25.0 rydberg
    cutoff_cd = 225.0 rydberg
    num_bands = 10
    ksampling{
        mesh{
            nx = 10
            ny = 10
            nz = 10
        }
    }
    scf_convergence{
        delta_total_energy = 1.e-10 hartree
        succession   = 3
    }
    initial_wavefunctions  = atomic_orbitals
    initial_charge_density = atomic_charge_density
}
structure{
    unit_cell_type = primitive
    unit_cell{
        !#units bohr
        a_vector =  0.0000000  3.4704637  3.4704637
        b_vector =  3.4704637  0.0000000  3.4704637
        c_vector =  3.4704637  3.4704637  0.0000000
    }
    symmetry{
         method = automatic
         tspace{
            lattice_system = fcc
         }
         sw_inversion = on
    }
    atom_list{
         atoms{
            !#tag  rx       ry       rz       weight   element    mobile
                  0.000    0.000     0.000       1        Cu        0
        }
    }
    element_list{
     #tag element  atomicnumber
               Cu          29
    }
}
wavefunction_solver{
        See later section
}
charge_mixing{
    mixing_methods{
        !#tag  method   rmxs    rmxe    itr   var   prec istr  nbmix  update
             broyden2  0.60    0.60    *     *     on     3    15     RENEW
    }
}
printlevel{
    base = 1
```

```
}
```

Examples of the **wavefunction_solver** block are shown below:

- matrix diagonalization

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  matrixdiagon   -1    *      *    *      *   on 1 off
    }
}
```

- lm+msd, partial space diagonalization is performed after updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  lm+msd   1    *      *    *      *   on 1 on
    }
    submat{
      before_renewal=off
    }
}
```

- lm+msd, partial space diagonalization is performed before updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  lm+msd   1    *      *    *      *   on 1 on
    }
    submat{
      before_renewal=on
    }
}
```

- lm+msd → rmm3, partial space diagonalization is performed after updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  lm+msd   1    *      *    *      *   on 1 on
                2  rmm3      -1    *      *    *      *    on   1    on
    }
    rmm{
      edelta_change_to_rmm = 5.0e-3
    }
    submat{
      before_renewal=off
    }
}
```

- lm+msd → rmm3, partial space diagonalization is performed before updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  lm+msd   1    *      *    *      *   on 1 on
                2  rmm3      -1    *      *    *      *    on   1    on
    }
    rmm{
      edelta_change_to_rmm = 5.0e-3
    }
    submat{
      before_renewal=on
    }
}
```

- Davidson → rmm3, partial space diagonalization is performed after updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  davidson   1    *     *    *     *   off 1 off
                2  rmm3      -1    *     *    *     *    on   1    on
    }
    rmm{
      edelta_change_to_rmm = 5.0e-3
    }
    submat{
      before_renewal=off
    }
}
```

- Davidson → rmm3, partial space diagonalization is performed before updating wavefunctions

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  davidson   1    *     *    *     *   off 1 off
                2  rmm3      -1    *     *    *     *    on   1    on
    }
    rmm{
      edelta_change_to_rmm = 5.0e-3
    }
    submat{
      before_renewal=on
    }
}
```

- Davidson

```
wavefunction_solver{
    solvers{
          !#tag   id sol    till_n  dts  dte   itr   var  prec cmix submat
                1  davidson   -1    *     *    *     *   off 1 off
    }
}
```

33. Results

Figure 7.4 shows the results from the benchmark calculations, and Table 7.1 lists the calculation times. In these benchmark tests, the PHASE program was compiled using the Intel Fortran Compiler 11.1 on Linux, and a computer cluster equipped with a 2.4GHz Opteron280 processor was used. The degree of parallelism for k-points was set to 4.

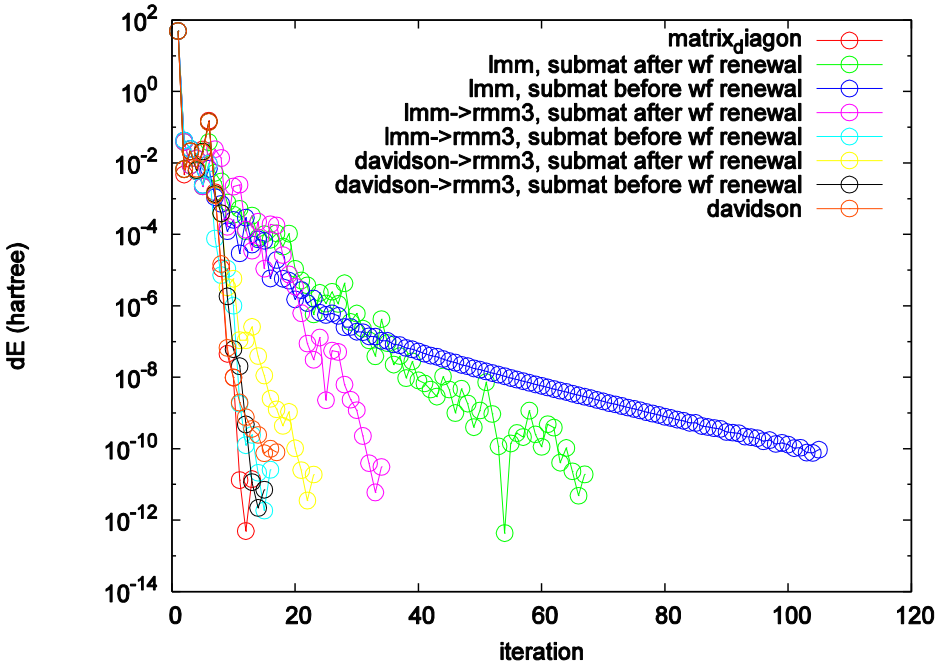


Figure 7.4 Convergence progress for each wavefunction solver

エラー! 参照元が見つかりません。 contains the calculation time for each wavefunction solver. Here the number of iterations indicates the number of charge mixings performed before convergence occurs. Note that the calculation times, obtained on the computer cluster equipped with an Opteron 280 2.4GHz processor, are tabulated here just for reference.

Table 7.1 Calculation times for each wavefunction solver

method (when partial space diagonalization is performed)	number of iteration	calculation time (sec)
matrix diagonalization	13	19.2
lm+msd, (after updating wavefunctions)	67	22.2
lm+msd, (before updating wavefunctions)	105	32.4
lm+msd → rmm3, (after updating wavefunctions)	34	12.4
lm+msd → rmm3, (before updating wavefunctions)	16	8.4
Davidson → rmm3, (after updating wavefunctions)	23	11.2
Davidson → rmm3, (before updating wavefunctions)	15	9.5
Davidson	17	11.8

In Figure 7.4, the horizontal axis contains the number of iterations, and the vertical axis gives the energy relative to the converged energy. Since the variational principle holds in the SCF calculation, the lower energy is more accurate.

Here matrix diagonalization seems to converge faster, but the amount of calculation for one step is generally large, and it cannot be applied to large systems. In these results, the methods that switch to rmm3 converge faster. In particular, the RMM3 method, in which partial space diagonalization is performed before wavefunctions are updated, converges much faster. Depending on the systems being studied, speed and stability of convergence change accordingly. It is recommended to select the best optimization method in each case. In most cases, LM+MSD→RMM3, Davidson's, and Davidson→RMM3 are stable and converge faster. If the RMM3 method is employed, partial space diagonalization should be applied before updating wavefunctions. If Davidson's method is employed, post-processing (**precon**) should be set to "off."

7.1.4.2 Fe(100) surface

A sample calculation for an Fe(100) surface is introduced here to illustrate spin-considered calculations. In this example, the same wavefunction solver was used, and several charge-mixing methods were tested. The input data introduced here are in the directory **samples/sol_cmix_test/Fe100**.

34. Input data

Input data, excluding the **charge_mixing** block, are shown below.

```
control{
  condition = initial
  max_iteration = 200
}
accuracy{
  num_bands = 52
  ksampling{
    method=monk
    mesh{
      nx = 6
      ny = 6
      nz = 1
    }
  }
  cutoff_wf = 30 rydberg
  cutoff_cd = 300 rydberg
  initial_wavefunctions = atomic_orbitals
  initial_charge_density = atomic_charge_density
  scf_convergence{
    delta_total_energy = 1e-9
    succession = 3
  }
  force_convergence{
    max_force = 0.0005 hartree/bohr
  }
}
structure{
  atom_list{
    atoms{
      #tag    element    rx    ry    rz    mobile    weight
            Fe    0.5    0.5    0    off    1
            Fe    0    0    0.0948333333333    off    2
            Fe    0    0    0.2845    off    2
            Fe    0.5    0.5    0.189666666667    off    2
    }
  }
  ferromagnetic_state{
    sw_fix_total_spin=off
    total_spin=14
    spin_fix_period=5
  }
  unit_cell{
    a_vector = 5.3762704477 0.0 0.0
    b_vector = 0.0 5.3762704477 0.0
    c_vector = 0.0 0.0 28.3458898822
  }
  element_list{
    #tag    element    atomicnumber    mass    zeta    deviation
          Fe    26    101802.230406    0.375    1.83
  }
  symmetry{
    method = automatic
    sw_inversion = on
  }
  magnetic_state=ferro
}
```

```
structure_evolution{
  method = gdiis
  gdiis{
    initial_method = cg
    c_forc2gdiis = 0.005 hartree/bohr
  }
}
wavefunction_solver{
  solvers{
    #tag    sol    till_n    prec    cmix    submat
          davidson    1    off    1    off
          rmm3       -1     on    1    on
  }
  rmm{
    edelta_change_to_rmm = 5e-3 hartree
  }
  submat{
    before_renewal = on
  }
}
charge_mixing{
        See later section
}
printoutlevel{
  base = 1
}
```

Examples of the **charge_mixing** block are shown below.

Case 0: Different mixing ratios are adopted for the sum and difference of the charge density. The Broyden2 method is employed for the charge-mixing algorithm.

```
charge_mixing{
  spin_density_mixfactor=4
  mixing_methods{
    #tag    no    method    rmxs    rmxe    itr    var    prec    istr    nbmix    update
          1    broyden2    0.1    0.1   40    linear    on    3    5    renew
  }
}
```

Case 1: Different mixing ratios are adopted for the sum and difference of the charge density. The Pulay method is employed for the charge-mixing algorithm.

```
charge_mixing{
  spin_density_mixfactor=4
  mixing_methods{
    #tag    no    method    rmxs    rmxe    itr    var    prec    istr    nbmix    update
          1    pulay    0.1    0.1   40    linear    on    3    15    renew
  }
}
```

Case 2: The same mixing ratios are adopted for the sum and difference of the charge density. The Broyden2 method is employed for the charge-mixing algorithm.

```
charge_mixing{
  spin_density_mixfactor=1
  mixing_methods{
    #tag    no    method    rmxs    rmxe    itr    var    prec    istr    nbmix    update
          1    broyden2    0.1    0.1   40    linear    on    3    15    renew
  }
}
```

Case 3: The same mixing ratios are adopted for the sum and difference of the charge density. The Pulay method is employed for the charge-mixing algorithm.

```
charge_mixing{
  spin_density_mixfactor=1
  mixing_methods{
```

```
    #tag    no    method    rmxs    rmxe    itr    var    prec    istr    nbmix    update
          1    pulay    0.1    0.1    40    linear    on    3    15    renew
  }
}
```

35. Results

The results of the benchmark tests are listed in Table 7.2. The total energies for these methods are also shown to ensure that these results converge to the same electronic state. In this example, case3 (different mixing ratios and the Pulay method) converges in the least number of iterations. Although case3 usually gives the best convergence, the Broyden2 method should be employed in some cases, or larger mixing ratios for the difference in charge density may be better in some problems. Moreover, if spin is considered, convergence is affected by whether initial spin polarization is fixed. When the calculation does not converge, find an optimal charge-mixing method by referring to these benchmark tests.

Table 7.2 Number of SCF iterations required for convergence and the resulting total energy for an Fe (100) surface

	number of SCF iterations	total energy (ha.)
case0	36	-153.877775988322
case1	32	-153.877775991437
case2	34	-153.877775825592
case3	29	-153.877775990755

7.2 Structure optimization

7.2.1 Optimization methods

7.2.1.1 Calculation examples

To examine the behavior of the optimization algorithms implemented in PHASE, we applied each algorithm to the following systems.

- case1: cis-dichlorohexane
- case2: rutile-type TiO_2
- case3: SiO_2
- case4: Si(001) surface

Input files for these examples are in the directory **samples/strevl_test**.

In all cases, the convergence criterion for the maximum atomic force was set to 10^{-4} Hartree/Bohr (This threshold is relatively strict.), and the convergence criterion for SCF calculations was set to 10^{-10} Hartree (succession=1). Updating atomic positions was carried out up to 200 times. The optimization calculations that exceed this limit were regarded as not converging. Each optimization algorithm was specified as follows:

36. quenched MD method

```
structure_evolution{
  method = quench
}
```

37. CG method

```
structure_evolution{
  method = cg
}
```

38. GDIIS method

```
structure_evolution{
  method = gdiis
  gdiis{
    initial_method = cg
    c_forc2gdiis = 0.01 hartree/bohr
  }
}
```

By the above input, the optimization is first processed by the CG method, and then the algorithm switches to the GDIIS method when the maximum atomic force became smaller than the threshold given by the **c_forc2gdiis** variable. Note that the first three steps are processed by the CG method even if the maximum force is smaller than the threshold.

39. BFGS method

```
structure_evolution{
  method = bfgs
  gdiis{
    initial_method = cg
    c_forc2gdiis = 0.01 hartree/bohr
  }
}
```

For the BFGS method, the optimization is first processed by the CG method, and the algorithm switches to the BFGS method when the maximum atomic force became smaller than the threshold given by the **c_forc2gdiis** variable. Note that the first three steps are processed by the CG method even if the maximum force is smaller than the threshold.

7.2.1.2 Results

エラー! 参照元が見つかりません。 lists the results of these benchmark tests. These results indicate that the quenched MD method converges slowly or not at all. In these tests, the time step was set to 100 au (default value). Convergence might improve by changing this value. Although the GDIIS method converges faster for SiO_2, it does not work well in the other cases. The behavior of the GDIIS method may be improved by changing **c_forc2gdiis** to a smaller value or by giving a more precise SCF convergence threshold. The CG method was relatively stable. The BFGS method converged in all cases, and it required fewer iterations, on average.

Table 7.3 Comparisons of the numbers of iteration required for convergence of the optimization algorithms. The label "unconverged" means the optimization did not converge to less than 10^{-4} within 200 cycles. Case1 is dichlorohexane, case 2 is rutile-type TiO_2, case 3 is SiO_2, and case 4 is the Si(001) surface.

	case 1	case 2	case 3	case 4
quenched MD	unconverged	115	166	unconverged
cg	195	62	28	124
GDIIS	unconverged	71	13	176
BFGS	87	38	16	67

7.3 Units in PHASE

The units used in PHASE are basically Hartree atomic units. The following table lists conversion factors from atomic units to other units.

energy	1 Hartree = 2 Rydberg = 27.21139615 eV = $4.359745836 \times 10^{-18}$ J
length	1 Bohr = 0.5291772480 Å= $0.5291772480 \times 10^{-10}$m
mass	1 au mass = 1822.877333 amu = 9.1093897×10^{-31} kg
volume	1 au volume = 0.1481847426 Å^3 = $1.48184726 \times 10^{-29}$ m^3
velocity	1 au velocity = $2.187691417 \times 10^{-2}$ Å/s = $2.187691417 \times 10^{8}$ m/s
force	1 Hartree/Bohr = 51.42208259 eV/Å= $8.238725025 \times 10^{-8}$ N
time	1 au time = $2.418884327 \times 10^{-2}$ fs = $2.418884327 \times 10^{-17}$ s
stress	1 au stress = $2.903628623 \times 10^{9}$ atm = $2.942101703 \times 10^{13}$ Pa
density	1 au density = 1.23013834×10^{4} amu/Å^3 = 9.1093897×10^{-4} g/cm^3 = 9.1093897×10^{-1} kg/m^3

7.4 FAQ

Questions and answers are summarized below.

Q: Band dispersion calculated by ekcal program seems wrong. What is a possible cause of the problem?
A: Check if the calculated wavefunctions converged sufficiently. See the log file output000.

Q: An error, which warns charge density is negative, occurs during a SCF calculation. How can I solve the error? (e.g., ** WARN CHG.DEN < 0.0 AT 1 - 0.8220751D-1)
A: Please check if the accuracy of the charge mesh is sufficient, and use a larger cutoff energy for the cutoff_cd variable. (Note: if the above warning appears only during the initial steps of an SCF calculation, it can be ignored.)

Q: How should I determine values for cutoff_wf and cutoff_cd?
A: An appropriate value for cutoff_wf mainly depends on the pseudopotentials used in the calculation. Normally, one should calculate a target crystal using several cutoff energies and then adopt cutoff energies that can reproduce experimental values of equilibrium lattice parameters or the bulk modulus. For normal TM potentials, cutoff_cd is determined by cutoff_cd=4 × cutoff_wf. However, cutoff_cd needs to be larger than this if pseudopotentials using PCC or ultra-soft potentials are employed in the calculation.

Q: mpirun command does not work. What is a possible cause of the problem?
A: Please check if the hostnames and user IDs are correctly written in .rhost.

Q: Optimization of electronic states (i.e., an SCF calculation) does not work well (i.e., charge density does not get converged). How can I solve the problem?
A: Try different options for the optimization algorithm by referring to Section 10.

Q: I specified xctype in the input parameter file. Is it necessary to use pseudopotentials generated by the same xctype functional?
A: Yes. It is basically necessary for consistency. If xctype is not specified in the input file, xctype defined in the pseudopotential files is employed. If xctype variables written in pseudopotential files employed are not consistent, the job will not execute.

Q: I plotted the band structure for a metal system using band.pl, but the Fermi level in this figure slipped out of the correct position. How can I fix this?
A: Change the Fermi energy written in nfenergy.data to that obtained by the SCF calculation.

Q: band.pl (or dos.pl) does not work. What is a possible cause of the problem?
A: Please check if the Perl scripts are executable. Try again after executing "chmod +x dos.pl." See chapter 9 for use of these scripts.

Q: SCF calculation was terminated, but it seems like the SCF convergence criterion was not satisfied. Why did the calculation terminate?
A: An SCF calculation terminates if the energy per an atom becomes lower than the threshold. However, the energy printed to standard output **output000** is the total energy, not the energy per an atom. Therefore, it is necessary to divide the total energy by the number of atoms to check whether the convergence criterion is satisfied.

Q: How can I ensure that the calculation converged?
A: If an SCF calculation converges, you can find the results in **nfefn.data** or **nfdynm.data.** If structure optimization converges, you can find the following description at the end of the file **continue.data**.

```
convergence
         2
```

Q: How can I visualize the unit cell that is shown by the blue line in Figure 3 of the tutorial.
A: After making the mol2 file shown below, read it with the PHASE Viewer.

```
@<TRIPOS>MOLECULE
Si8 frame
    8   12    0
    0    0    0    0

grid file
@<TRIPOS>ATOM
     1 N     0.000000  0.000000  0.000000    N.4    1  GLY         0.0000
     2 N     5.430000  0.000000  0.000000    N.4    1  GLY         0.0000
     3 N     0.000000  5.430000  0.000000    N.4    1  GLY         0.0000
     4 N     0.000000  0.000000  5.430000    N.4    1  GLY         0.0000
     5 N     0.000000  5.430000  5.430000    N.4    1  GLY         0.0000
     6 N     5.430000  0.000000  5.430000    N.4    1  GLY         0.0000
     7 N     5.430000  5.430000  0.000000    N.4    1  GLY         0.0000
     8 N     5.430000  5.430000  5.430000    N.4    1  GLY         0.0000
@<TRIPOS>BOND
     1    1  2        1
     2    1  4        1
     3    1  3        1
     4    2  6        1
     5    2  7        1
     6    3  5        1
     7    3  7        1
     8    4  5        1
     9    4  6        1
    10    5  8        1
    11    6  8        1
    12    7  8        1
```

8. Installation of PHASE

8.1 Operating environment

PHASE works at various computer environments from PCs to advanced supercomputers. The PHASE program is written in Fortran90 and C; thus, compilers of these languages are required. The MPI library is also necessary if parallel processing is required.

The required and optional software and libraries are as follows:

- Fortran90 compiler and C compiler (required)
- MPI library (required for parallel calculations)
- Libraries for matrix operations: LAPACK, BLAS (optional)
- FFT library: FFTW (optional)
- Perl (optional, but required for PHASE tools)
- Gnuplot (optional, but required for PHASE tools)

The computer platforms that support PHASE are tabulated below.

Computer platforms that support PHASE

platform	compilers	available libraries
Linux	GNU Compiler Intel Compiler PGI Compiler	LAPACK, BLAS, ScaLAPACK MKL, ACML FFTW3
Windows XP	GNU Compiler Intel Compiler PGI Compiler	MKL, ACML FFTW3
Intel ver. Mac OS X	GNU Compiler Intel Compiler	MKL, ACML FFTW3
Oracle Solaris	GNU Compiler SUN Compiler	Sun Perf.(LAPACK) ACML FFTW3
SGI Altix	Intel Compiler	SCSL, MKL FFTW3
IBM AIX	IBM XL	ESSL(LAPACK) FFTW3
Hitachi SR11000	HITACHI IBM XL	MATRIX/MPP(FFT) HITACHI LAPACK ESSL(LAPACK)
NEC SX Series	Fortran90/SX	Mathkeisan(LAPACK) ASL(FFT)
Fujitsuu FX10	Fujitsu Compiler	

- MPICH1, MPICH2, and OpenMPI are available for the MPI library.
- The newest AMD Core Math Library for various platforms is available at http://developer.amd.com/.
- The GNU compiler (gfortran, gcc) must be newer than ver. 4.1. The newest GNU Compiler (Windows ver., MacOS ver., Linux ver.) can be downloaded from http://gcc.gnu.org/.
- The PGI compiler must be newer than ver. 6.2.
- The Intel compiler must be newer than ver. 9.1.

8.2 Installation

This section introduces installation of PHASE in a Linux environment. In this example, the Intel Fortran compiler is employed for installation. If another compiler is used for the installation, choose that compiler when the installer asks you which compiler is used. In this example, OpenMPI is employed for the MPI library. If "Serial" is chosen in the interactive installation process, you can compile the PHASE program without the MPI library.

First, decompress the PHASE package file **phase_v1200.tar.gz** at the directory in which PHASE is installed.

```
$ tar zxf phase_v1200.tar.gz
```

Go into the directory phase_v1100 and run the installer.

```
$ cd phase_v1000
$ ./install.sh
```

```
 === PHASE installer ===
 Do you want to install PHASE? (yes/no) [yes]
```

The installer asks you whether to install PHASE. Press the Enter key to start the installation.

```
Supported platforms
 0) GNU Linux (IA32)
 1) GNU Linux (EM64T/AMD64)
 2) NEC SX Series
 x) Exit
Enter number of your platform. [0]
```

Supported platforms are displayed. Input "0," which corresponds to GNU Linux (AI32), and press the Enter key.

```
Supported compilers
 0) GNU compiler collection (gfortran)
 1) Intel Fortran compiler
 x) Exit
Enter number of a desired compiler. [0]
```

Supported compilers are displayed. Input "2," which corresponds to the Intel Fortran compiler 9.x, and press the Enter key.

```
Supported programming-models
 0) Serial
 1) MPI parallel
 x) Exit
Enter number of a desired programming-model. [0]
```

Supported programming models are displayed. Input "1," which corresponds to MPI parallel, and press the Enter key.

```
Supported MPI libraries
 0) MPICH1/MPICH2/Open MPI
 1) Intel(R) MPI
 x) Exit
Enter number of a desired MPI library. [0]
```

Supported MPI libraries are displayed. Input "0" which corresponds to OpenMPI, and press the Enter key.

```
Supported BLAS/LAPACK
 0) Netlib BLAS/LAPACK
 1) Intel Math Kernel Library (MKL)
 x) Exit
Enter number of a desired library. [0]
```

Supported BLAS/LAPACK libraries are displayed. Input "0," which corresponds to Netlib BLAS/LAPACK, and press the Enter key.

```
Supported FFT libraries
 0) Built-in FFT subroutnes
 1) FFTW3 library
 x) Exit
Enter number of a desired library. [0]
```

Supported FFT libraries are displayed. Input "0," which corresponds to built-in FFT subroutines, and press the Enter key

```
Do you want to edit the makefile that has been generated? (yes/no/exit) [no]
```

The installer asks you whether to edit the generated Makefile. Press the Enter key if you do not need to edit the Makefile.

```
 Do you want to make PHASE now? (yes/no) [yes]
```

Press the Enter key to start the installation of PHASE.

```
PHASE was successfully installed.
Do you want to check the installed programs? (yes/no) [no]
```

After the message "PHASE was successfully installed," the installer asks you whether to execute a test calculation. Input "yes" and press the Enter key if you want to run the test calculations. If the installed PHASE program works correctly, the following results will be obtained.

```
Do you want to check the installed programs? (yes/no) [no]
yes
Checking total-energy calculation.
 Total energy : -7.897015156331 Hartree/cell
 Reference    : -7.897015156332 Hartree/cell
Checking band-energy calculation.
 Valence band maximum : 0.233846 Hartree
 Reference            : 0.233846 Hartree
```

PHASE is executed using the mpirun or mpiexec command in the MPI library.

If you add the directory **$HOME/phase_v1200/bin** to the environmental variable **PATH**, you can execute PHASE programs without entering the path to these programs.

For the Bourne shell, add the following line to **$HOME/.bashrc**.

```
export PATH=$HOME/phase_v1200/bin:$PATH
```

In the C shell, add the following line to **$HOME/.cshrc**.

```
setenv PATH $HOME/phase_v1200/bin:$PATH
```

Also, add the bin directory of the MPI library to the PATH.
For the Bourne shell, add the following line to $HOME/.bashrc.

```
export PATH=$HOME/openmpi/bin:$PATH
```

In the C shell, add the following line to $HOME/.cshrc.

```
setenv PATH $HOME/openmpi/bin:$PATH
```

Now, mpirun and phase can be executed simply as follows:

```
$ mpirun -np 2 phase ne=1 nk=2
```

8.3 Notice for each platform

8.3.1 Linux

Make the FFTW3 interface of the Intel Math Kernel Library by using the Intel C++ compiler as follows.

```
cd /opt/intel/mkl/9.1/interfaces/fftw3xf
make lib32
```

In the EM64T environment, use libem64t instead of lib32 above. The library file **libfftw3xf_intel.a** will be generated in

```
/opt/intel/mkl/9.1/lib/32     (IA32 environment)
```

or in

```
/opt/intel/mkl/9.1/lib/em64t  (EM64T environment)
```

Note: if the MKL library was installed into a directory other than **/opt/intel**, the library file is also installed in that directory.

8.3.2 Windows XP

On a Windows platform, a Linux-compatible environment is required. Please install MSYS/MinGW or Cygwin. In case of Cygwin, the **make** command also needs to be installed.
If the MPI parallel version is necessary, install the MPI library for Windows DeinoMPI in advance.
Make the FFTW3 interface of the Intel Math Kernel Library using the **nmake** command with 'F=ms' and 'MKL_SUBVERS=serial' options (Use Microsoft C++ compiler).
The makefile is in

```
C:¥Program Files¥Intel¥MKL¥9.1¥interfaces¥fftw3xf
```

To correctly make the serial version of the library, remove the '/MT' option in the second-to-last line of this makefile. After modifying the makefile, open an MS-DOS command prompt and make the library as below:

```
C:¥Program Files¥Intel¥MKL¥9.1¥interfaces¥fftw3xf
nmake lib32 F=ms MKL_SUBVERS=serial
```

In the EM64T environment, use libem64t instead of lib32 above. The library file **fftw3xf_ms.lib** will be generated in

```
C:¥Program Files¥Intel¥MKL¥9.1¥lib¥_serial¥ia32¥lib  (IA32 environment)
```

or in

```
C:¥Program Files¥Intel¥MKL¥9.1¥lib¥_serial¥em64t¥lib  (EM64T environment)
```

8.3.3 Mac OS X (Intel ver.)

Use the Intel Fortran compiler ver. 10.
Make the FFTW3 interface of the Intel Math Kernel Library using the Intel C++ compiler as follows:

```
cd /Library/Frameworks/Intel_MKL.framework/Version/9.1/interfaces/fftw3xf
make lib32 MKL_SUBVERS=serial
```

In the EM64T environment, use libem64t instead of lib32 above. The library file **libfftw3xf_intel.a** will be generated in

```
/Library/Frameworks/Intel_MKL.framework/Version/9.1/lib_serial/32     (IA32)
```

or in

```
/Library/Frameworks/Intel_MKL.framework/Version/9.1/lib_serial/em64t (EM64T)
```

To make the MPI-parallelized version, install OpenMPI in advance.

9. Usage of programs and tools

9.1 Program phase

9.1.1 Executing phase

One can use PHASE to perform SCF calculations or MD simulations. Density of states (DOS) or band structures can also be calculated from charge-density distributions.

Prepare an input parameter file and pseudopotential files and put them into an execution directory. Put **file_names.data** into the execution directory as well, if it is needed.

When you run serial calculations using a single processor (1 core), execute the phase program as below. Here "../../phase_v1200/bin/" indicates the directory in which PHASE was installed.

```
% ../../phase_v1200/bin/phase
```

When you run PHASE in parallel, use the execution command in the MPI library. Usually, the mpirun or mpiexec command is used. For more details, see the manuals for your platform.

```
% mpirun -np NP ../../phase_v1200/bin/phase ne=NE nk=NK
```

Here NP is the number of MPI processes, while NE and NK are the degrees of parallelism for bands and k-points, respectively.

9.1.2 Options for parallel calculations

9.1.2.1 Parallelization over bands and parallelization over k-points

In parallel calculations, you need to specify the degree of parallelism for bands NE and for k-points NK. Note that NE × NK must equal NP. If ne and nk are omitted, NE = NP and NK = 1 are employed.

```
% mpirun -np NP ../../phase_v1200/bin/phase ne=NE nk=NK
```

Normally, parallelization over k-points is more effective than parallelization over bands. Therefore, it is usually better to maximize the number of k-points for parallelization. However, the number of k-points is normally reduced for large systems, and the actual number of k-points may not be divisible by the number of available processors. An error occurs if NK is larger than the number of actual k-points. Sufficient efficiency is not achieved if the number of k-points is not divisible by NK. Use band parallelization at the same time when needed.

9.1.2.2 Parallelization of replica method

"Parallelization of replica method" is available for some methods such as NEB, constrained dynamics, and metadynamics. To use replica parallelization, execute PHASE as below:

```
% mpirun -np NP ../../phase_v1200/bin/phase nr=NR ne=NE nk=NK
```

Here NR indicates the degree of parallelism for replicas. The relationship NP = NR × NE × NK must hold. Although parallelization of replicas is more effective than that of k-points, the most slowly converging replica can be a hindrance to the whole calculation.

9.1.3 Parallelization over G points (beta version)

PHASE supports not only band parallelization and k-point parallelization but also parallelization over G points for plane-wave functions. However, this function is still under testing, and the following restrictions are applied for the parallelization.

- G-point parallelization is not available with k-point parallelization
- Post-processing is not available with parallelization

The source codes supporting G-point parallelization are in the directory **src_phase_3d**. Go into the directory and execute the following command to generate the Makefile.

```
% sh configure
```

To generate the Makefile, the installer asks you questions similar to those described in the installation section. After generating the Makefile, modify the Makefile if needed and execute the make command as below:

```
% make
```

Before executing the calculation, prepare a file named **nml.lst** in the execution directory. The following is an example of this file.

```
&decomp3d
ng=NG
ne=NE
nk=1
/
```

Here NG gives the degree of parallelization over G points, and NE gives the degree of parallelization for bands. NG × NE must be equal to the number of MPI processes.

```
% mpirun -np NP ../../phase_v1200/bin/phase
```

9.2 Program ekcal

9.2.1 Executing ekcal

The program ekcal is used to calculate DOS or band structures from charge-density distributions obtained from SCF calculations done by PHASE.

First, copy the charge-density file **nfchgt.data** into an execution directory or specify this file by the keyword F_CHG in **file_names.data**.

For band-structure calculations, prepare the file for setting k-point sampling **kpoint.data**.

Execute the program ekcal as below. Here "phase_v1100/bin/" indicates the directory in which PHASE was installed.

```
% ../../phase_v1100/bin/ekcal
```

9.2.2 Options for ekcal

9.4 dos.pl: a tool for plotting DOS

PHASE generates data for the DOS in file dos.data. See another section of this manual or the tutorial for more details. The Perl script dos.pl can visualize the dos.data. The way to plot the DOS is described below. First, copy the dos.data in the example directory into the work directory.

```
$ cd PHASE_INST_DIR/samples/tools/work
$ cp ../example/dos.data .
```

Make sure that the dos.data file is copied in this directory by using the ls command.

```
$ ls dos.*
dos.data
```

To visualize the dos.data, execute the Perl script by the following command:

```
$ dos.pl dos.data -erange=-13,5 -color
```

This generates an EPS file **density_of_states.eps**. In UNIX, you can see this file by using ghostview or gv etc. as follows:

```
$ ghostview density_of_states.eps
```

or

```
$ gv density_of_states.eps
```

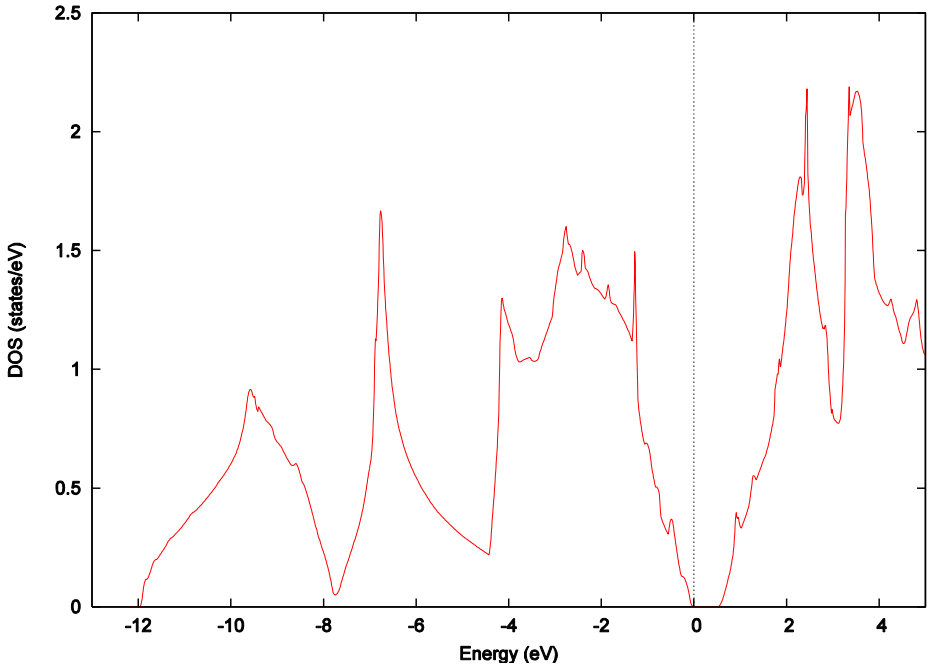


Figure 9.1 DOS of bulk Si

Here the –erange option gives the energy range to be plotted, and the –color option indicates that the figure is output in color.

9.4.1 Options for dos.pl

Usage of the dos.pl script is printed if it is executed without any arguments.

```
$ dos.pl
```

```
Version: 3.00
Usage: dos.pl DosData -erange=Emin,Emax -einc=dE -dosrange=DOSmin,DOSmax -dosinc=dDOS
-title=STRING -with_fermi -width=SIZE -font=SIZE -color -mode={total|layer|atom|projected}
```

```
-epsf={yes|no} -data={yes|no}
```

The DOS data file, normally dos.data, is given to the first argument DosData. The options for the dos.pl are listed below:

-erange=Emin,Emax	Specify the energy range to be plotted in units of eV. For example, -erange = -10,5 indicates that DOS is plotted from -10.0 eV to 5.0 eV. If the option is not given, the energy range is automatically determined by the minimum and maximum of the data given.
-einc=dE	Specify the scale resolution for the horizontal axis. For example, -einc=2 indicates that the scale interval is 2.0 eV.
-dosrange=DOSmin,DOSmax	Specify the range of DOS to be plotted. For example, -dosrange=0,12 indicates that the DOS is plotted from 0 states/eV to 12 states/eV.
-dosinc=dDOS	Specify the scale resolution for the vertical axis (DOS). For example, -dosinc=2 indicates that the scale interval is 2 states/eV.
-title=STRING	Give a title to the graph. For example, -title="Total DOS"
-with_fermi	If this option is given, a dotted line is drawn at the Fermi level for metals or at the level of the highest valence band for insulators and semiconductors.
-width=SIZE	Give the width of this figure. Defaults to 1.0. For example, -width=0.8
-font=SIZE	Specify the font size. Defaults to 14. For example, -font=28
-color	If this option is given, the graph is dumped in color.
-mode={total|layer|atom}	Options are as follows: total: plot total DOS (default) layer: plot layer-divided PDOS atom: plot atom-divided PDOS projected: plot atomic-orbital-divided PDOS
-epsf={yes|no}	If no, a postscript file is not generated. Defaults to yes.
-data={yes|no}	If yes, instead of generating an eps file, the PDOS data for each layer or atom are separately dumped into files.

9.5 band_kpoint.pl: a tool for generating k-points

To plot the band structure, many k-points along symmetrical lines are required. The program ekcal calculates eigenvalues at these k-points. The Perl script band_kpoint.pl generates these k-points, and the generated k-points data are dumped into **kpoint.data**, and then this file is given to ekcal. The format of the input file of this script is shown below.

```
dkv
b1x b2x b3x
b1y b2y b3y
b1z b2z b3z
n1 n2 n3 nd # Symbol
...
```

The variable dkv gives the interval between k-points; b1x, b1y, b1z indicate the x, y, z components of the reciprocal vector $\boldsymbol{b_1}$, and similarly for the reciprocal vectors $\boldsymbol{b_2}, \boldsymbol{b_3}$. The fifth line defines a special k-point and its symbol. Although the specification of the symbol is not required, it is used as a label when the band structure is plotted. The vectors of these k-points k are specified by integer numbers n_1, n_2, n_3, n_d as

$$k = \frac{n_1}{n_d} b_1 + \frac{n_2}{n_d} b_2 + \frac{n_3}{n_d} b_3$$

The symbols are written after the #. An example for a face-centered cubic lattice is shown below.

```
0.02                        <---- interval of k-points
-1.0  1.0  1.0
1.0 -1.0  1.0               <---- reciprocal lattice vector
1.0  1.0 -1.0
0 1 1 2 # X                 <---- n1 n2 n3 nd # Symbol
0 0 0 1 # {/Symbol G}
1 1 1 2 # L
5 2 5 8 # U
1 0 1 2 # X
```

The above input is in the example directory. Copy this file and execute band_kpoint.pl as below:

```
$ cd PHASE_INST_DIR/samples/tools/work
$ cp ../example/bandkpt_fcc_xglux.in .
$ band_kpoint.pl bandkpt_fcc_xglux.in > output
```

This generates kpoint.data, which contains k-points used for band structure calculation. Program ekcal calculates eigenenergies at these k-points by reading this file.

9.6 band.pl: a tool for plotting band structure

9.6.1 Executing band.pl

The script band.pl is a script that plot band structure. The output file from ekcal "nfenergy.data" and the input file for band_kpoint.pl are given to band.pl as input. The example directory contains the file nfenergy.data, which is obtained from the eigenenergy calculation of ekcal using the file **kpoint.data** generated in the previous section. To plot the band structure, copy the files nfenergy.data and bandkpt_fcc_xglux.in in the example directory to the work directory and execute band.pl as below:

```
$ cd PHASE_INST_DIR/samples/tools/work
$ cp ../example/nfenergy.data .
$ cp ../example/bandkpt_fcc_xglux.in .
$ band.pl nfenergy.data bandkpt_fcc_xglux.in -erange=-13,5 -color
```

This generates the EPS file band_structure.eps. This file can be displayed by ghostview or gv.

```
$ ghostview band_structure.eps
```

or

```
$ gv band_structure.eps
```

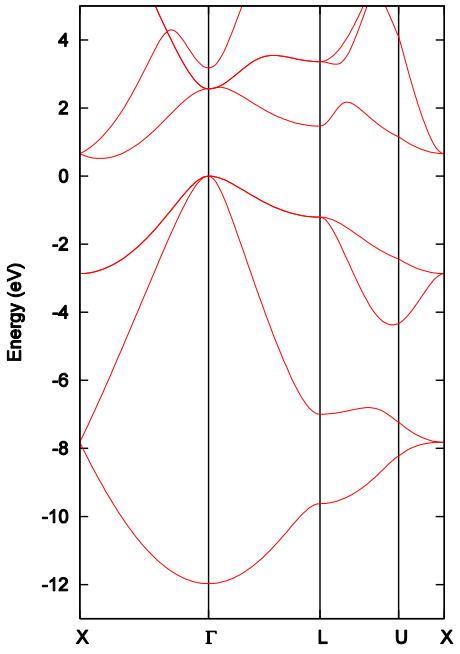


Figure 9.2 Band structure of bulk Si

Here the –erange option gives the energy range to be plotted, and the –color option indicates that the figure is dumped in color.

9.6.2 Options for band.pl

Usage of band.pl is printed if this script is executed without any arguments.

```
$ band.pl
```

```
Usage: band.pl EnergyDataFile KpointFile -erange=Emin,Emax
-einc=dE -ptype={solid_circles|lines} -with_fermi
```

-width=SIZE –color

The eigenenergy and k-point data files are given to the first and second arguments, EnergyDataFile, and KpointFile. The options for band.pl are listed below:

-erange=Emin,Emax	Specify the energy range to be plotted in units of eV. For example, -erange=-10,5 indicates that the DOS is plotted from -10.0 eV to 5.0 eV.
-einc=dE	Specify the scale resolution for the vertical axis. For example, -einc=2 indicates that the scale interval is 2.0 eV.
-ptype=TYPE	Specify plot type. Options are -ptype=solid_circles: display with black closed circles -ptype=lines: display with lines (default)
-with_fermi	If this option is given, a dotted line is drawn at the Fermi level for metals or at the level of the highest valence band for insulators and semiconductors.
-width=SIZE	Give the width of this figure. Defaults to 0.5. For example, -width=0.3
-color	If this option is given, the graph is dumped in color.

9.7 dynm2tr2.pl: a tool for converting to extended trajectory format

The Perl script dynm2tr2.pl converts "nfdym.dat," which contains atomic position data obtained by structure optimization or MD simulation, to the extended trajectory format.

The dynm2tr2.pl can be executed as below:

```
$ dynm2tr2.pl nfdynm.data
```

This command generates files dynm.tr2, which contains atomic positions, and grid.mol2, which defines cell vectors, etc. Here we convert the results from geometry optimization for two Si atoms in FCC primitive to the extended trajectory format and visualize below.

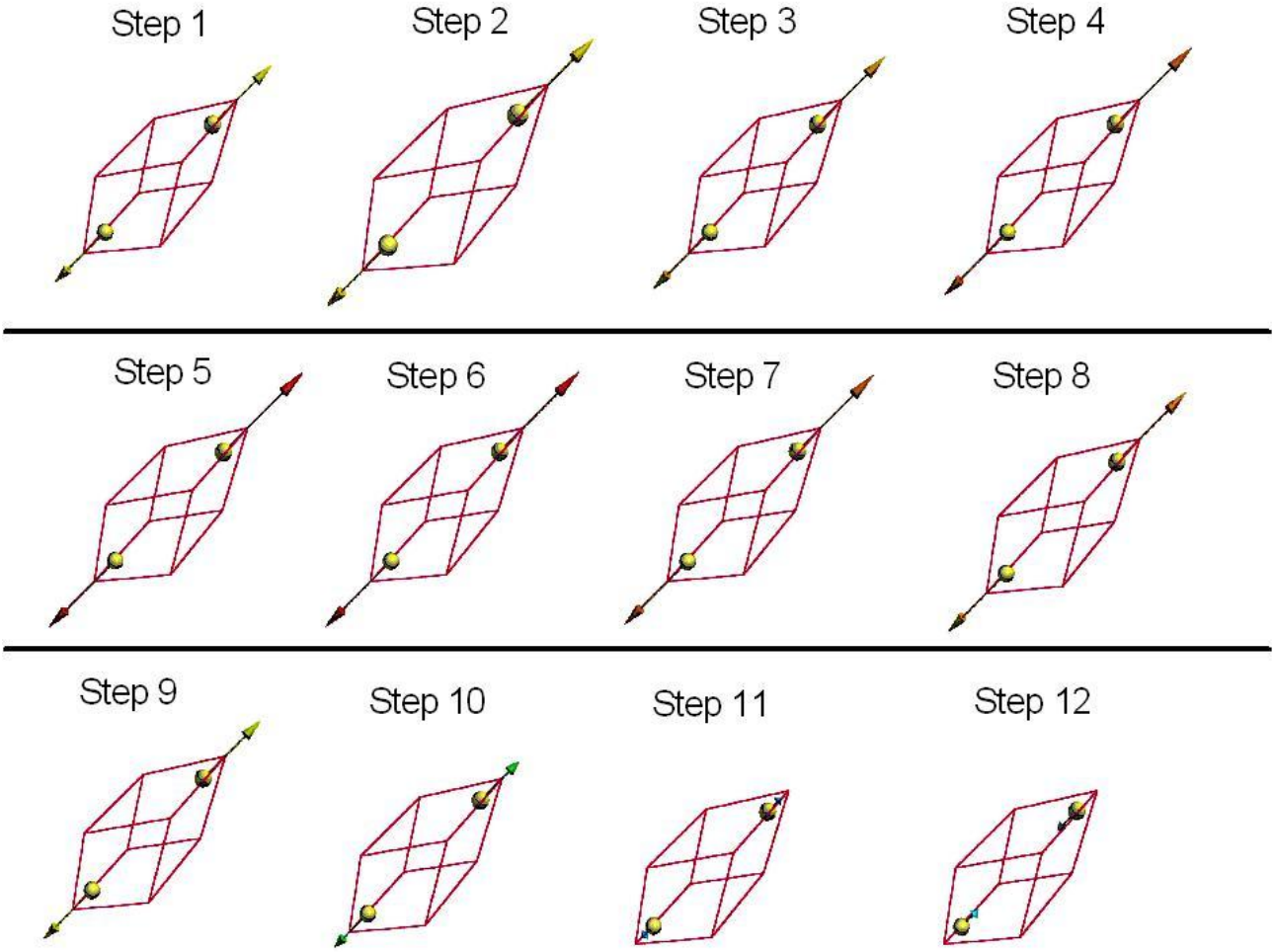


Figure 9.3 Visualized structure optimization progress for bulk Si

The arrows in エラー! 参照元が見つかりません。 represent forces acting on atoms. These arrows indicate that after the force was maximized, the force decreased and optimization converged. Although the primitive cell is displayed in エラー! 参照元が見つかりません。, changing the origin or cell vectors can be specified by making control.inp exemplified below.

```
origin  1.2825 1.2825 1.2825
vector1 10.26  0     0
vector2  0    10.26  0
vector3  0     0    10.26
```

If the above control.inp is used as below, the origin is set to (1.2825,1.2825,1.2825) Bohr, and cell vectors are

set to (10.26,0,0), (0,10.26,0), (0,0,10.26) Bohr.

```
$ dynm2tr2.pl nfdynm.data control.inp
```

エラー! 参照元が見つかりません。 shows step 10 of the optimization progress displayed with the Bravais cell given by control.inp.

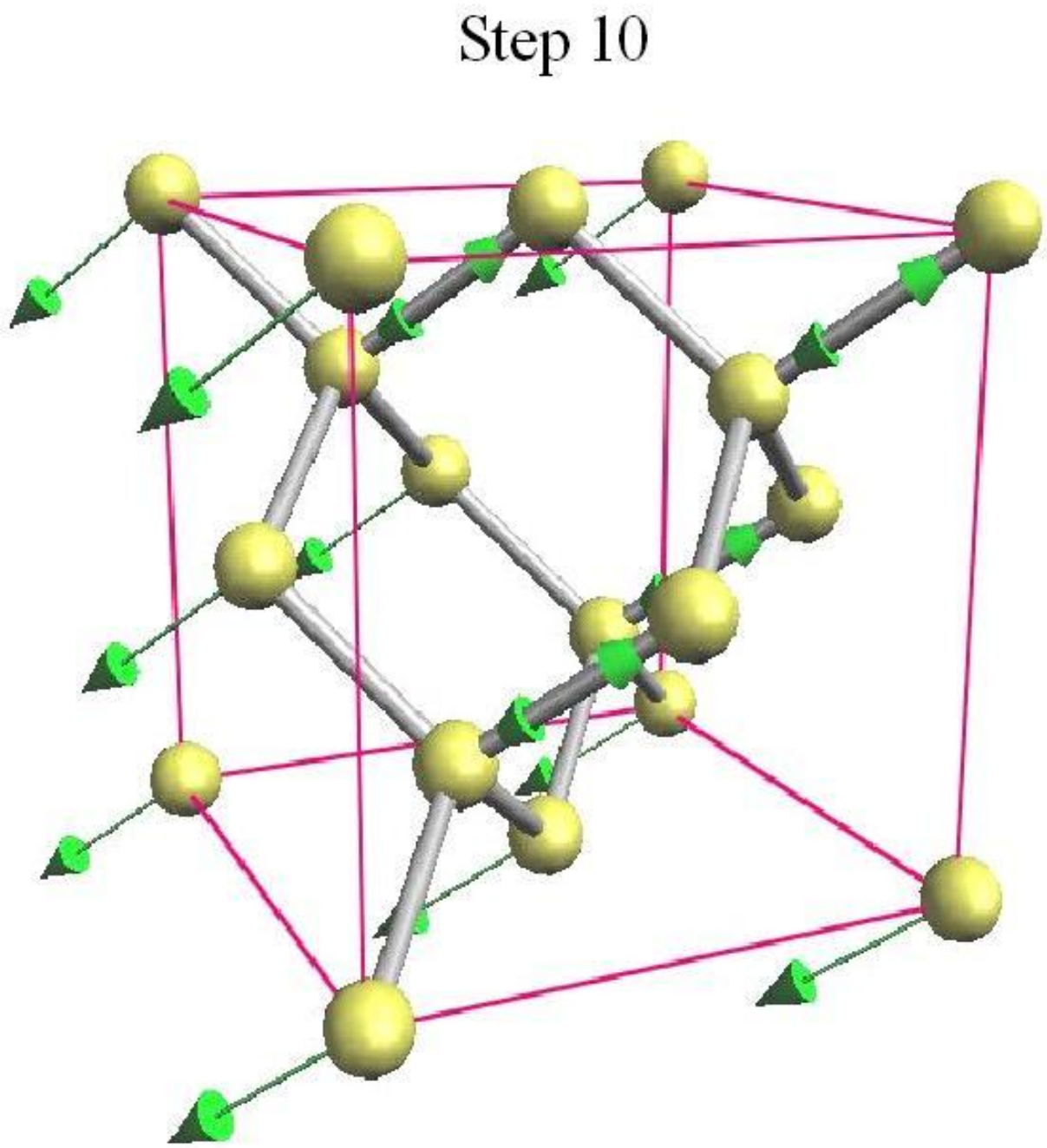


Figure 9.4 Step 10 of the optimization progress for Si atoms displayed with the Bravais cell

9.8 freq.pl: a tool for plotting frequency level diagrams

Frequencies and eigenvectors of the normal vibrational modes are obtained by vibrational analysis in PHASE. The results from a vibrational analysis are dumped into the file **mode.data**. The Perl script freq.pl extracts these results from mode.data and plots a frequency-level diagram. After executing freq.pl, an EPS file named freq.eps is generated.

```
$ freq.pl [options] mode.data
```

エラー! 参照元が見つかりません。 shows the frequency-level diagram for bulk Si.

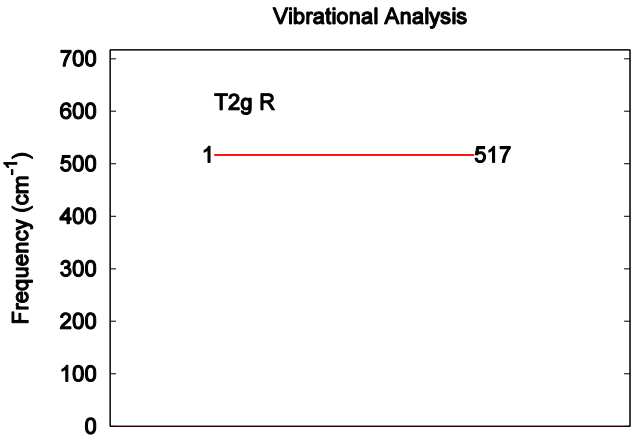


Figure 9.5 Frequency-level diagram for bulk Si

The horizontal lines that represent frequency levels are classified according to irreducible representations such as T2g. The irreducible representations and symbols representing their activity (IR, R, IR&R, and NON) are displayed by horizontal lines. Here IR represents infrared activity, R represents Raman activity, IR&R represents both infrared and Raman activity, and NON represents the silent mode. The number to the right of this line is the frequency in cm^{-1} units. The horizontal lines are numbered in order of frequency, and those numbers are displayed to the left of the line.

9.8.1 Options for freq.pl

Usage of freq.pl is printed if this script is executed without any arguments.

```
$ freq.pl
```

```
*** A visualization program for vibrational freqencies ***
Usage: freq.pl [-width=W] [-height=H] [-nrep=N] {-solid|-mol|-ignored_modes=LIST}
mode.data
```

The options for freq.pl are listed below:

-width=W	Give the width of this figure. Defaults to 1.0. For example, -width=0.3
-height=H	Give the height of this figure. Defaults to 1.0. For example, -height=2.5
-nrep=N	Specify the number of irreducible representations displayed in one diagram. If the number of irreducible representations obtained is larger than the value given by this option, multiple

	EPS files will be generated.
-solid	If this option is given (default), translational modes are not displayed.
-mol	If this option is given, translational and rotational modes are not displayed.
-ignored_modes=LIST	The modes specified by this option are not displayed. For example, -ignored_modes=1,2,3 hides the modes 1, 2, and 3.

9.9 animate.pl: a tool for converting normal modes to the extended trajectory format

The Perl script animate.pl reads eigenvectors of vibrational modes from mode.data and dumps the trajectory of normal vibrations to an extended trajectory formatted file.

The origin and cell vectors can be specified by the file control.inp exemplified below:

```
origin  1.27189 1.27189 1.27189
vector1 10.17512 0 0
vector2 0 10.17512 0
vector3 0 0 10.17512
```

In the above example, to display with the Bravais cell, the origin is set to (1.27189, 1.27189, 1.27189) Bohr, and cell vectors are specified as (10.17512, 0, 0), (0, 10.17512, 0), (0, 0, 10.17512) Bohr.

The animate.pl can be executed as follows:

```
$ animate.pl mode.data control.inp
```

Vibrational modes are dumped into the extended trajectory files mode_1.tr2, mode_2.tr2, …, mode_6.tr2, and cell vectors are dumped into the file grid.mol2. An extended trajectory file is generated for each vibrational mode.

エラー! 参照元が見つかりません。 shows mode_6.tr2, the sixth eigenvectors for the normal vibrations of bulk Si.

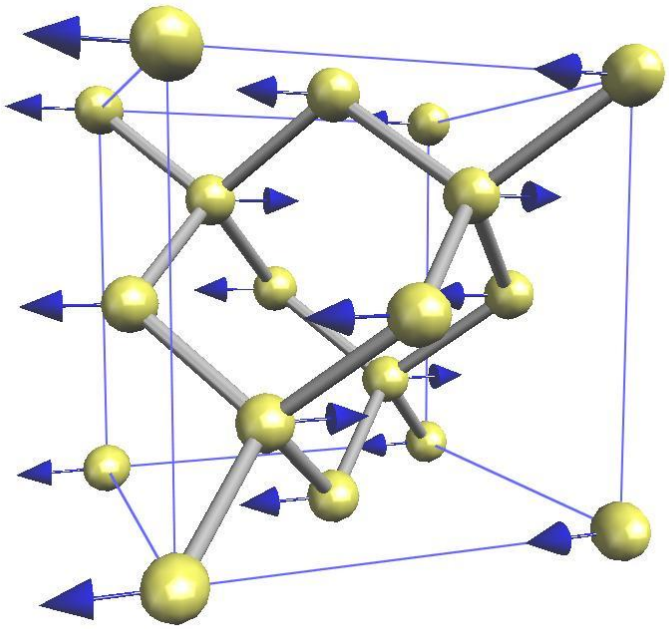


Figure 9.6 Eigenvectors of normal vibration for bulk Si

10. Input and output files

10.1 Input files

10.1.1 Input parameter file: nfinp.data

10.1.2 Pseudopotential files

Here we describe the format of the pseudopotential file.
The following shows a pseudopotential file for the Si atom.

```
   14   4   3   0   2  : zatom, ival, iloc, itpcc
ldapw91  : name
     2.160000    0.860000    1.605400   -0.605400  :   alp,cc
   1501   96.000000   60.000000  :   nmesh,  xh, rmax
VALL
 -0.1425006403755233 2E+07 -0.1410239247897529 1E+07 -0.1395625118175556 5E+07
 -0.1381162428840420 9E+07 -0.1366849610592247 1E+07 -0.1352685110365134 7E+07
 -0.1338667391198572 9E+07 -0.1324794932058984 6E+07 -0.1311066227674651 6E+07
 -0.1297479788372393 4E+07 -0.1284034139915911 6E+07 -0.1270727823345830 1E+07
 -0.1257559394821393 4E+07 -0.1244527425463785 9E+07 -0.1231630501201091 7E+07
 -0.1218867222614865 7E+07 -0.1206236204788271 3E+07 -0.1193736077155812 5E+07
 -0.1181365483354622 5E+07 -0.1169123081077276 3E+07 -0.1157007541926145 4E+07
 -0.1145017551269260 6E+07 -0.1133151808097655 2E+07 -0.1121409024884198 1E+07
 -0.1109787927443895 0E+07 -0.1098287254795615 5E+07 -0.1086905759025274 6E+07
 -0.1075642205150428 1E+07 -0.1064495370986257 2E+07 -0.1053464047012956 3E+07
 -0.1042547036244496 6E+07 -0.1031743154098732 2E+07 -0.1021051228268870 6E+07
 -0.1010470098596271 1E+07 -0.9999986169445488 5E+06 -0.9896356470749989 1E+06
  .....................................................................
  .....................................................................
  .....................................................................
```

You can insert comment lines beginning with # into the first lines. If the comment lines are inserted in the pseudopotential file, PHASE prints these comments to the standard output (oputput000). In the above example, the first four lines define the following parameters.

First line: natomn, ival, iloc, itpcc, igncpp
These variables represent the atomic number Z, the number of valence electrons Z_v, a number obtained by plus 1 to azimuthal quantum number of the localized orbital l_{loc}, a flag for core charge correction (1 or 0), and the format of the pseudopotential data (GNCPP1(=1), GNCPP2(=2)), respectively.

Second line: xctype
This line indicates the type of exchange-correlation energy. Options are LDAPW91 and GGAPBE.

Third line: alp1, alp2, cc1, cc2
These parameters $\alpha_1, \alpha_2, c_1, c_2$ are used in the following equation to calculate the pseudopotential of the core part:

$$V_{core} = -\frac{Z_v}{r}\{c_1 \mathrm{erf}(\sqrt{\alpha_1} r) + c_2 \mathrm{erf}(\sqrt{\alpha_2} r)\}$$

where $erf(\cdot)$ represents the Gaussian error function, and $c_1 + c_2 = 1$.

Fourth line: nmesh, xh, rmax
These parameters N_{mesh}, x_h, r_{max} are used in the following equation to generate the mesh in the radial direction:

$$r_i = r_{max} \exp\left((i - N_{mesh})/x_h\right) \ (i = 1, \cdots, N_{mesh})$$

where N_{mesh} represents the number of meshes in the radial direction.

In the above example, these four lines indicate that the pseudopotential file is for the Si atom of LDAPW91. The "VALL" in the fifth line is a symbol used to check the file in the PHASE program. The lines after the fifth contain actual pseudopotential data. The first block of this data represents screened all-electron potential, $V_{scr}^{AE}(r)$ and its data format is as follows:

```
do ir = 1, nmesh
 V_scr^AE(ir)
end do
```

The second block of this data contains the screened local potential, $V_{scr,l_{loc}}^{PP}(r, l)$ and its data format is as follows:

```
do ir = 1, nmesh
 V_scr,l_loc^PP(ir, iloc)
end do
```

The third block of this data contains $\rho_v(r)$, which is the product of the valence charge density $n_v(r)$ and the surface area of a sphere $4\pi r^2$ ($\rho_v(r) = 4\pi r^2 n_v(r)$). The data format for this block is as follows:

```
do ir = 1, nmesh
 ρ_v(r)
end do
```

After these three blocks, data for pseudo wave functions and pseudopotentials are dumped. The data format is completely different between norm-conserving and ultra-soft pseudopotentials. See the user manual of CIAO for more details.

10.2 Input/Output setting file: file_name.data

10.3 Input files (ekcal)

10.3.1 k-point data file: kpoint.data （F_KPOINT）

This file is mainly used for band calculations via ekcal. The k-points to be calculated are written to this file. Then, these k-points are read from this file when "file" is specified for the k-sampling method. This file is usually generated by the Perl script band_kpoint.pl. The following shows an example.

```
141 141          (a)
0 50 50 100 1    (b)
0 49 49 100 1
0 48 48 100 1
0 47 47 100 1
0 46 46 100 1
0 45 45 100 1
0 44 44 100 1
0 43 43 100 1
   ......
   ......
   ......
```

Each line represents the following:

(a) Number of k-points. This example has 141 k-points.

(b) These five integer numbers are the n_1, n_2, n_3, n_d, w in

$$\vec{k} = w \times \left(\frac{n_1}{n_d} \overrightarrow{b_1} + \frac{n_1}{n_d} \overrightarrow{b_2} + \frac{n_3}{n_d} \overrightarrow{b_3} \right)$$

where $\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}$ are reciprocal lattice vectors.

10.4 Output file

10.4.1 DOS file: dos.data (F_DOS)

Calculated DOS is dumped into a file designated by the F_DOS keyword. The default name of this file is **dos.data**.

If spin is not considered in the calculation, then the total DOS is dumped as follows:

```
  No.    E(hr.)         dos(hr.)          E(eV)          dos(eV)              sum
    6  -0.20528      0.0000000000    -11.949000      0.0000000000        0.0000000000
   16  -0.20491      0.0000000000    -11.939000      0.0000000000        0.0000000000
   26  -0.20455      0.0000000000    -11.929000      0.0000000000        0.0000000000
..............
..............
..............
END
```

Here No. (in the first column) represents the number assigned to each state, E(hr.) represents the energy in units of Hartree, dos(hr.) represents the DOS in units of states/Hartree, E(eV) represents the energy in units of eV, dos(eV) represents the DOS in units of states/eV, and sum represents the integrated DOS.

However, if spin is considered in the calculation, total DOS is dumped as follows:

```
No.  E(hr.)     dos_up(hr.)       dos_down(hr.)       E(eV)          dos_up(eV)         dos_down(eV)
      sum_up    sum_down sum_total
  1  -1.5451      0.0000000000      0.0000000000       -45.4403      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
 11  -1.5441      0.0000000000      0.0000000000       -45.4131      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
 21  -1.5431      0.0000000000      0.0000000000       -45.3859      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
 31  -1.5421      0.0000000000      0.0000000000       -45.3587      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
 41  -1.5411      0.0000000000      0.0000000000       -45.3315      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
 51  -1.5401      0.0000000000      0.0000000000       -45.3043      0.0000000000       0.0000000
000    0.0000    0.0000    0.0000
```

The dos_up and dos_down represents DOS for up-spin and down-spin; the sum_up and sum_down represents the integrated DOS for up-spin and down-spin; the sum_total is sum of sum_up and sum_down.If a layer-projected or an atom-projected DOS is calculated, its descriptor and data are also dumped after the data for total DOS.

- Atom-projected DOS

The following output is obtained for atom PDOS.

```
ALDOS      num_atom =        1
  No.    E(hr.)         dos(hr.)          E(eV)          dos(eV)              sum
    6  -0.84950      0.0000000000    -26.189850      0.0000000000        0.0000000
000
   16  -0.84850      0.0000000002    -26.162639      0.0000000000        0.0000000
000
                              ......................
                              ......................
                              ......................
END
ALDOS      num_atom =        2
                              ......................
```

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

Atom PDOS is dumped between the descriptor ALDOS and END line. The num_atoms = 2 indicates the serial number of atoms.

- LDOS for layers

LDOS for layers are dumped as follows:

```
LAYERDOS   num_layer =       1
  No.   E(hr.)        dos(hr.)         E(eV)          dos(eV)              sum
    6  -0.84950      0.0000000000    -26.189850      0.0000000000        0.0000000
000
   16  -0.84850      0.0000000002    -26.162639      0.0000000000        0.0000000
000
                              .......................
                              .......................
                              .......................
END
LAYERDOS     num_layer =       2
                              .......................
                              .......................
                              .......................
```

The data are dumped in same format as the atomic LDOS. The descriptor for layered LDOS is LAYERDOS, and num_layer indicates the number of layers defined in an input file.

10.4.2 Energy history file: nfefn.data (F_ENF)

Changing total energy in structure relaxation calculations or values of potential and kinetic energies in MD simulations are dumped into a file designated by the F_ENF keyword.

- Structure relaxation

The following shows a typical example of the F_ENF file for structure relaxation.

```
 iter_ion, iter_total, etotal, forcmx
     1      24     -108.4397629733        0.0086160410
     2      40     -108.4401764388        0.0076051917
     3      56     -108.4405310817        0.0068758156
     4      73     -108.4410640011        0.0065717365
     5      94     -108.4414256084        0.0099533097
     6     113     -108.4414317178        0.0094159378
                  ........
                  ........
                  ........
```

Each column represents

iter_ion	the number of iteration for updating ion positions
iter_total	the total number of SCF iterations
etotal	total energy in units of Hartree
forcmx	maximum value of the atomic force (Hartree/Bohr^3) Structure relaxation calculations continue until this value becomes lower than the given convergence criterion.

- MD simulations

The following shows a typical example of the E_ENF file for MD simulations.

```
      iter_ion, iter_total, etotal, ekina, econst, forcmx
     1      18    -7.8953179624     0.0000000000    -7.8953179624     0.0186964345
     2      30    -7.8953851218     0.0000665502    -7.8953185716     0.0183575425
     3      43    -7.8955768901     0.0002565396    -7.8953203505     0.0173392067
                        ........
                        ........
                        ........
```

In addition to columns like those in structure relaxation, the following columns are also printed.

ekina	kinetic energy for the system
econst	conserved quantity of the system (i.e., the total energy for a constant-energy MD simulation or the sum of the total energy and thermostat energy for a constant-temperature MD simulation)

10.4.3 Trajectory file: nfdynm.data（F_DYNM）

When structure relaxation calculations or MD simulations are performed, atomic coordinates and atomic forces are dumped into a file designated by the F_DYNM keyword. The following shows a typical example of the F_DYNM file. In this file, all values are printed in atomic units.

```
#
#   a_vector =         9.2863024980        0.0000000000        0.0000000000
#   b_vector =        -4.6431512490        8.0421738710        0.0000000000                (a)
#   c_vector =         0.0000000000        0.0000000000       10.2158587136
#   ntyp =        2 natm =        9                                                       (b)
# (natm->type)     1    1    1    1    1    1    2    2    2                            (c)
# (speciesname)     1 :   O                                                             (d)
# (speciesname)     2 :   Si
#
 cps and forc at (iter_ion, iter_total =     1      24 )                                 (e)
    1    3.161057370    1.169332082    1.214972077   -0.004058   -0.005565   -0.004966   (f)
    2    6.693102525    2.152889944    4.620258315    0.006945   -0.001028   -0.004994
    3    4.075293851    4.719951845    8.025544553   -0.002872    0.006394   -0.004796
    4   -1.482093879    6.872841789    5.595600399   -0.004362    0.005502    0.004993
    5   -0.567857398    3.322222026    9.000886637   -0.002792   -0.006296    0.004965
    6    2.049951276    5.889283925    2.190314161    0.006974    0.000708    0.004795
    7    4.921740324    0.000000000    3.405282833    0.001436    0.000122    0.000068
    8   -2.460870162    4.262352150    6.810569070   -0.000612    0.001305   -0.000066
    9    2.182281087    3.779821719   10.215855308   -0.000660   -0.001143    0.000001
 cps and forc at (iter_ion, iter_total =     2      40 )
    1    3.156999743    1.163767576    1.210005993   -0.002904   -0.005755   -0.003892
    2    6.700048015    2.151861938    4.615264365    0.006567    0.000186   -0.003832
    3    4.072421499    4.726345880    8.020748072   -0.003503    0.005487   -0.003829
    4   -1.486455954    6.878343743    5.600593135   -0.003122    0.005780    0.003831
    5   -0.570648922    3.315925959    9.005851266   -0.003532   -0.005392    0.003892
    6    2.056925355    5.889992076    2.195109289    0.006503   -0.000290    0.003828
    7    4.923176344    0.000121757    3.405351146    0.000397   -0.000013    0.000018
    8   -2.461482612    4.263656762    6.810503226   -0.000210    0.000337   -0.000017
    9    2.181621403    3.778679157   10.215856638   -0.000197   -0.000341    0.000000

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

(a) Cell vectors: a_vector, b_vector, and c_vector represent the vectors of the a-axis, b-axis, and c-axis, respectively.

(b) ntyp indicates the number of atomic species used in the simulation; natm indicates the number of atoms in the simulation.

(c) natom→type defines atomics species for atoms. In this example, atoms from No. 1 to No. 6 correspond to atomic species 1, atoms from No. 7 to No. 9 correspond to atomic species 2.

(d) speciesname defines atomic species and their identification numbers. In this example, 1 and 2 are assigned to oxygen and silicon atoms, respectively.

(e) Number of iterations for updating ions and the total number of SCF iterations

(f) Atomic positions and atomic forces. The first column gives the ID of atoms, the second to fourth columns contain the xyz coordinates of atoms, the fifth to seventh are the xyz components of atomic forces. If the print level for velocity is set to 2, velocities of atoms are printed in the seventh to ninth columns in atomic units.

10.4.4 Charge density file: nfchr.cube（F_CHR）

Charge density is dumped into a file designated by the F_CHR keyword. This file can be obtained in Gaussian cube format by assigning "cube" to the file_type variable. The Gaussian cube format is recommended because it can be visualized. The following shows an example of the Gaussian cube file.

```
 This is a title line for the bulk Si                                                    (a)
 SCF Total Density
     8    0.0000    0.0000    0.0000                                                    (b)
    20  0.513000  0.000000  0.000000                                                    (c)
    20  0.000000  0.513000  0.000000
    20  0.000000  0.000000  0.513000
    14  4.000000  1.282500  1.282500  1.282500                                          (d)
    14  4.000000  8.977500  8.977500  8.977500
    14  4.000000  1.282500  6.412500  6.412500
    14  4.000000  8.977500  3.847500  3.847500
    14  4.000000  6.412500  1.282500  6.412500
    14  4.000000  3.847500  8.977500  3.847500
    14  4.000000  6.412500  6.412500  1.282500
    14  4.000000  3.847500  3.847500  8.977500
  0.87897E-01  0.80457E-01  0.63811E-01  0.47743E-01  0.35993E-01  0.26628E-01          (e)
  0.18342E-01  0.12084E-01  0.83725E-02  0.65941E-02  0.60774E-02  0.65941E-02
  0.83725E-02  0.12084E-01  0.18342E-01  0.26628E-01  0.35993E-01  0.47743E-01
  0.63811E-01  0.80457E-01  0.80457E-01  0.76575E-01  0.63379E-01  0.51118E-01
  0.43367E-01  0.35993E-01  0.26413E-01  0.17302E-01  0.11265E-01  0.80672E-02
  0.65941E-02  0.62411E-02  0.68963E-02  0.88010E-02  0.12493E-01  0.18342E-01
  0.26413E-01  0.37600E-01  0.53180E-01  0.70418E-01  0.63811E-01  0.63379E-01
                                 ........
                                 ........
                                 ........
                                 ........
                                 ........
```

(a) Title and comment line

(b) Eight is the number of atoms. "0.0000 0.0000 0.0000" represents the origin. The origin is always (0,0,0) in PHASE.

(c) Grid box and number of meshes are defined here. For example, "20 0.513000 0.000000 0.000000" indicates that the number of mesh divisions for the first axis is 20, and the length of each mesh is 0.513,0.00,0.00. Unit of length is Bohr.

(d) First number indicates the atomic number. In this example, 14 identifies a silicon atom. The second number 4.00000 indicates the number of valence electrons. The third to fifth numbers correspond to xyz coordinates of the atom. Unit is Bohr.

(e) Charge density for each grid point is printed in the following order.

```
(1,1,1)  (1,1,2)  ......  (1,1,20)  (1,2,1)  (1,2,2)
......  (1,20,20)  (2,1,1)  ......
(20,20,19)  (20,20,20)
```

10.4.5 Restart file: continue.data （F_CNTN）

This file contains data used to restart calculations. You can edit this file to change parameters for the continued calculation; for example, you can change the convergence criterion of an SCF calculation. The following shows an example of this file.

```
 iteration, iteration_ionic, iteration_electronic
       19         1        19                                                    (a)
 Ionic System
  (natm)
         2                                                                       (b)
  (pos)
  0.1249999999999999D+00  0.1250000000000001D+00  0.1250000000000001D+00       (c)
  0.8749999999999994D+00  0.8749999999999994D+00  0.8749999999999994D+00
  (cps)
  0.1282864712563094D+01  0.1282864712563093D+01  0.1282864712563093D+01       (d)
  0.8980052987941646D+01  0.8980052987941646D+01  0.8980052987941646D+01
  (cpd)
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  1))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  2))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  (cpo(  3))
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
  0.0000000000000000D+00  0.0000000000000000D+00  0.0000000000000000D+00
 Total Energy
 -0.7851066208137508D+01 -0.7851066208137508D+01                               (e)
 isolver
        17
convergence
         2                                                                     (f)
edelta_ontheway
  0.1000000000000000D-07                                                       (g)
```

(a) Total number of iterations, the number of iterations for updating ion positions, the number of SCF iterations

(b) Number of atoms

(c) Atomic positions, referred to cell vectors

(d) Cartesian coordinates of atoms in units of Bohr

(e) Total energies for the previous step and the current step

(f) Convergence progress. Options are
0: not converged,
1: SCF converged, but structure relaxation is not converged,
2: converged
If the value is 2 and the calculation is restarted, post-processing starts immediately. If you want to change the convergence criterion and restart the calculation from an SCF calculation, set this value to 0.

(g) Convergence criterion for SCF. If you want to change the convergence criterion of SCF in the middle of a calculation, change this value also.

10.4.6 Eigenvalue data file: nfenergy.data（F_ENERG）

Eigenvalues calculated by ekcal are dumped into this file. The following shows a typical example of this file.

```
 num_kpoints =    117                                                    (a)
 num_bands   =      8                                                    (b)
 nspin       =      1                                                    (c)
 Valence band max   =   0.233846                                         (d)

 nk_converged =      117                                                 (e)
 ik =    1 (  0.500000  0.500000  0.000000 )
 ik =    2 (  0.487805  0.487805  0.000000 )
 ik =    3 (  0.475610  0.475610  0.000000 )
 ik =    4 (  0.463415  0.463415  0.000000 )
 ik =    5 (  0.451220  0.451220  0.000000 )
 ik =    6 (  0.439024  0.439024  0.000000 )
...
...
...

=== energy_eigen_values ===
 ik =    1 (  0.000000  0.500000  0.500000 )                            (f)
     -0.0484324576     -0.0484324576      0.1258094928      0.1258094928  (g)
      0.2619554301      0.2619554301      0.6015285208      0.6015285208
=== energy_eigen_values ===
 ik =    2 (  0.000000  0.490000  0.490000 )
     -0.0540717201     -0.0427149632      0.1258687739      0.1258687739
      0.2607026807      0.2633829927      0.6006243932      0.6006243932
                           ......
                           ......
                           ......
```

(a) Number of k-points. This example has 117 k-points.

(b) Number of bands. This example has eight bands.

(c) Spin degree of freedom: 1 or 2. In this example, the value 1 means that spin polarization is not considered in the calculation.

(d) Fermi energy. For semiconductor/insulator, the energy of the valence band edge is printed. The unit is Hartree.

(e) Calculated k-points.

(f) Eigenvalues are printed from here. This first line represents the k-point to which this eigenvalue corresponds. In this example, the first k-point corresponds to (0,0.5,0.5) of the reciprocal lattice vector.

(g) Eigenvalues for all bands are printed. The unit is Hartree.

If spin polarization is considered, the output of eigenenergies is almost same, but "UP" or "DOWN" is added to item (f). Eigenvalues corresponding to the major and minor spins are printed.

```
                           ......
                           ......
                           ......
=== energy_eigen_values ===
 ik =    1 (  0.000000  0.000000  0.000000)    UP
     -0.1998699758      0.0267639589      0.0267639589      0.0267639589
      0.0725171077      0.0725171077      1.0289118953      1.0289118953
      1.0289118953      1.1650173104      1.1650173104      1.1650173104
      1.2129026022      1.2129026022      1.2994754011      1.2994754011
      1.2994754011      1.6365336765      2.2629596795      2.2629596795
```

```
=== energy_eigen_values ===
 ik =    2 (  0.000000  0.000000  0.000000)  DOWN
     -0.1960420390      0.1062941746      0.1062941746      0.1062941746
      0.1799862148      0.1799862148      1.0183970612      1.0183970612
      1.0183970612      1.2174266166      1.2174266166      1.2192701193
      1.2192701193      1.2192701193      1.3289165100      1.3289165100
      1.3289165100      1.6910264603      2.2876818717      2.2876818717
                           ......
                           ......
                           ......
```

11. Dielectric function calculation program UVSOR

11.1 Linear-response time-dependent density functional theory（LR-TDDFT）

11.1.1 General features

11.1.1.1 introduction

In the independent particle approximation, excitation spectra of materials are obtained by calculating transitions between the ground-state eigenenergy levels of the Kohn–Sham equation. However, in experimentally observed excitation spectra, the transition energy and peak amplitudes differ from the spectra obtained with this approximation, indicating that interparticle interactions are not negligible. In the following, we explain one theoretical method by which interparticle interactions can be considered, at least in the linear response regime. This method is called linear-response time-dependent density functional theory (LR-TDDFT).

11.1.1.2 Application to solids

In the independent particle approximation, the response function χ^0 of the system on changing the external field is written as

$$\chi^0_{GG'}(q,\omega) = 2\int_{\mathrm{BZ}} \frac{d\boldsymbol{k}}{(2\pi)^3} \sum_{n,n'} \left(f_{n\boldsymbol{k}-\boldsymbol{q}} - f_{n'\,\boldsymbol{k}}\right) \frac{\rho^*_{n'\,n\boldsymbol{k}}(\boldsymbol{q},\boldsymbol{G})\rho_{n'\,n\boldsymbol{k}}\left(\boldsymbol{q},\boldsymbol{G}'\right)}{\omega - \left(\varepsilon_{n'\,\boldsymbol{k}} - \varepsilon_{n\boldsymbol{k}-\boldsymbol{q}}\right) + i\eta}$$

where

$$\rho_{n'\,n\boldsymbol{k}}(\boldsymbol{q},\boldsymbol{G}) = \left\langle n'\;\boldsymbol{k}\middle| e^{i(\boldsymbol{q}+\boldsymbol{G})\cdot\boldsymbol{r}} \middle| n\boldsymbol{k}-\boldsymbol{q}\right\rangle$$

The response function χ of the interacting system is given by the Dyson equation,

$$\chi = \chi^0 + \chi^0(v + f_{\mathrm{xc}})\chi$$

where v is the Coulomb interaction and f_{xc} is exchange-correlation interaction. The former is given by

$$v_{\boldsymbol{G}}(\boldsymbol{q}) = \frac{4\pi}{|\boldsymbol{q}+\boldsymbol{G}|^2}$$

but the latter is not well defined. We adopt the following two models for f_{xc}:

- Random-phase approximation (RPA)

$$f_{\mathrm{xc}} = 0$$

- Long-range correction (LRC)

$$f_{\mathrm{xc}} = -\frac{\alpha}{|\boldsymbol{q}+\boldsymbol{G}|^2}$$

The spectrum that is calculated by PHASE is the macroscopic dielectric function,

$$\varepsilon_M(\omega) = 1 - v_0\overline{\chi}_{\boldsymbol{G}=\boldsymbol{G}'\,=0}(\omega)$$

Here $\overline{\chi}$ is a response function similar to χ; their difference is caused by removing the G = 0 component of the Coulomb kernel.

$$\overline{\chi} = \chi_0 + \chi_0(\overline{v} + f_{\mathrm{xc}})\overline{\chi}$$

$$\overline{v}(\boldsymbol{q}) = \begin{cases} v_{\boldsymbol{G}}(\boldsymbol{q}) & \boldsymbol{G} \neq 0 (134) \\ 0 & \boldsymbol{G} = 0 (135) \end{cases}$$

11.1.1.3 Application to isolated systems

For isolated systems such as molecules, we adopt an alternative approach that is based on the Bethe–Salpeter equation. In this method, the electron-hole Green's function L^0 of the noninteracting system is defined by

$$\lim_{\boldsymbol{q}\to 0}\chi^0_{\boldsymbol{GG}'}\ (q,\omega) = -i\sum_{nn'}\sum_{\boldsymbol{k}}\lim_{\boldsymbol{q}\to 0}\left[\rho^*_{n'\ n\boldsymbol{k}}(\boldsymbol{q},\boldsymbol{G})\rho_{n'\ n\boldsymbol{k}}\left(\boldsymbol{q},\boldsymbol{G}'\right)\right]L^0_{nn'\ \boldsymbol{k}}left(\omega$$

In a similar manner, the electron-hole Green's function $\overline{L}$ of the interacting system is defined by

$$\lim_{\boldsymbol{q}\to 0}\overline{\chi}_{\boldsymbol{GG}'}\ (q,\omega) = -i\sum_{nn'}\sum_{mm'}\sum_{\boldsymbol{k},\boldsymbol{k}'}\lim_{\boldsymbol{q}\to 0}\left[\rho^*_{n'\ n\boldsymbol{k}}(\boldsymbol{q},\boldsymbol{G})\rho_{m'\ m\boldsymbol{k}'}\left(\boldsymbol{q},\boldsymbol{G}'\right)\right]\overline{L}_{nn'\ \boldsymbol{k},mm'\ \boldsymbol{k}'}(\omega)$$

These two Green's functions are related by the Bethe–Salpeter equation,

$$\overline{L}_{nn'\ \boldsymbol{k},m'\ m\boldsymbol{k}'}(\omega) = L^0_{nn'\ \boldsymbol{k}}(\omega)\left[\delta_{nm}\delta_{n'\ m'}\delta_{\boldsymbol{k}\boldsymbol{k}'} + i\sum_{ss'}\sum_{\boldsymbol{k}_1}\Xi_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1}\overline{L}_{ss'\ \boldsymbol{k}_1,mm'\ \boldsymbol{k}'}(\omega)\right]$$

where

$$\Xi_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1} = -V_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1} - K_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1}$$

$$V_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1} = \frac{1}{\Omega N_{\boldsymbol{k}}}\sum_{\boldsymbol{G}\neq 0}\rho_{nn'\ \boldsymbol{k}}(\boldsymbol{q}=0,\boldsymbol{G})\rho^*_{ss'\ \boldsymbol{k}_1}(\boldsymbol{q}=0,\boldsymbol{G})v(\boldsymbol{G})$$

$$K_{nn'\ \boldsymbol{k}ss'\ \boldsymbol{k}_1} = 2\int\int d\boldsymbol{r}d\boldsymbol{r}'\ \phi^*_{n\boldsymbol{k}}(\boldsymbol{r})\phi_{n'\ \boldsymbol{k}}(\boldsymbol{r})f_{\mathrm{xc}}\left(\boldsymbol{r},\boldsymbol{r}'\right)\phi_{s'\ \boldsymbol{k}'}\left(\boldsymbol{r}'\right)\phi^*_{s\boldsymbol{k}'}\left(\boldsymbol{r}'\right)$$

Here Ω is the system volume and $N_{\boldsymbol{k}}$ is the number of k-points sampled. In addition, we adopt the following model for f_{xc}:

- Adiabatic local density approximation (ALDA)

$$f_{\mathrm{xc}}\left(\boldsymbol{r},\boldsymbol{r}'\right) = \delta\left(\mathrm{r}-\mathrm{r}'\right)\frac{\partial v_{xc}(\rho(\boldsymbol{r}))}{\partial\rho}$$

The spectrum that is calculated by PHASE is the photoadsorption cross-section (PACS),

$$\sigma(\omega) = \frac{\Omega}{c}\omega\mathrm{Im}[\varepsilon_M(\omega)]$$

11.1.2 Input parameters

11.1.2.1 Control block

To use the LR-TDDFT method, the following steps are essential. First, in the "control" block, declare "condition = fixed_charge." This indicates that LR-TDDFT uses the charge density previously obtained from an SCF ground-state calculation. In addition, if you use the TM-type pseudopotential, in which the local potential is a specific orbital potential, set "use_additional_projector = on."

```
control{
   condition = fixed_charge
   cpumax = 1 day
   max_iteration = 600
   use_additional_projector=on
```

```
}
```

11.1.2.2 Accuracy block

In the "accuracy" block, specify parameters for the eigenvalues calculations.

```
accuracy{
…
    ek_convergence{
      num_extra_bands = 0
      num_max_iteration = 2000
      sw_eval_eig_diff = on
      delta_eigenvalue = 1.e-6 rydberg
      succession   = 3
    }
…
}
```

11.1.2.3 Structure block

In the "symmetry" block, indicate that all symmetry operations are neglected except the "E" symmetry.

```
structure{
…
  symmetry{
    method = manual
    tspace{
      lattice_system = primitive
      generators{
       !#tag rotation tx  ty  tz
                E     0   0   0
      }
    }
  }
…
}
```

11.1.2.4 Spectrum block

The "spectrum" block contains parameters concerning the calculations of excitation spectra. The parameters available and their meanings are explained below.

```
spectrum{
    type = optics
    momentum_transfer{
      deltaq = 1.0E-3
      nx = 1.1, ny = 1.2, nz = 0.9
      LongWaveApprox = ON
    }
    tddft{
      sw_tddft = ON
      solver{
        equation = DYSON
      }
      XC_Kernel{
        kernel_type = LRC
        LRC_alpha = 0.2
```

```
      }
      Coulomb_Kernel{
        sw_NLF = OFF
      }
      Expansion{
        NumGVec = 80
      }
    }
    energy{
      low = 0.0 eV
      high = 10.0 eV
      step = 0.05 eV
    }
    BZ_integration{
      width =  0.15 eV
    }
    band_gap_correction{
      scissor_operator = 0.6d0 eV
    }
}
```

type [OPTICS]	OPTICS and PACS are available. The former is for calculating the dielectric function of solids. The latter is for calculating the photoadsorption cross-section of isolated systems, such as molecules.
momentum_transfer	Name of a block in which the momentum transfer vector is specified.
deltaq [1.0E-3]	Length of the momentum transfer vector q is specified in units of $Å^{-1}$.
nx, ny, nz [0.0, 0.0, 1.0]	Direction of the momentum transfer vector q.
L LongWaveApprox [ON]	ON and OFF are available. The former is specified when you adopt the long-wave limit approximation ($q \to 0$).
tddft	Name of a block in which parameters concerning TDDFT are specified.
sw_tddft [OFF]	ON and OFF are available. The former is specified when you use LR-TDDFT.
solver	Name of a block in which you specify the solver.
equation [DYSON]	DYSON and BS are available. The former is specified when you use the DYSON equation. The latter is specified when you use the Bethe–Salpeter equation. Note that the BS equation is used for isolated systems such as molecules.
XC_Kernel	Name of a block in which parameters concerning the exchange-correlation kernel are specified.
kernel_type [RPA]	RPA, LRC, and ALDA-R are available. RPA is specified when you neglect the exchange-correlation kernel. LRC is specified when you consider the long-range interaction correction, which is meaningful in periodic systems such as solids. ALDA-R is used for isolated systems, such as molecules.
LRC_alpha [1.0]	Variable that is specified when you set "LRC" as "kernel_type."
Coulomb_Kernel	Name of a block in which the parameter concerning the Coulomb kernel is specified.
sw_NLF [OFF]	ON and OFF are available. The former is specified when you use an approximation in which the local field ($\|\boldsymbol{G}\| > 0$) is neglected.
Expansion	Name of a block in which the parameter concerning the G-vectors used in the plane-wave expansion is set.
NumGVec [100]	Number of G vectors is specified.
energy	Name of a block in which variables concerning the energy range used in the calculation of spectra are set.
low, high, step	Low (high): minimum (maximum) of the energy range. Step: interval of the quantized energy range.

BZ_Integration	Name of a block in which the parameter concerning the Brillouin zone integration is set.
width [1.0E-4 hartree]	Width of Lorentzian broadening.
band_gap_correction	Name of a block in which the parameter concerning the bandgap correction is set.
scissor_operator [0.0]	Band gap is artificially increased by this quantity.

11.1.3 Execution

Before executing LR-TDDFT, use the following command to perform the SCF calculation. Here NP indicates the number of MPI processes and BINDIR identifies the location of the executable program.

```
mpirun -np NP phase
```

Subsequently, execute the LR-TDDFT calculation by this command:

```
mpirun -np NP tdlrmain
```

11.1.4 output

The resulting spectrum data are printed to the file "spectrum.data," whose file format is explained below.

A. Case when "type" is set to "OPTICS"

```
#                 Optical spectrum
#                            NonInteracting              Interacting
#      Energy[eV]        Real        Imaginary         Real        Imaginary
     0.000000       8.626260       0.252860       9.678273       0.327540
     0.050000       8.627214       0.252961       9.679507       0.327682
     ..................................................................
```

The first column contains the energy of the excitation spectra. The second and third columns contain the real and imaginary parts of the dielectric function in the independent particle approximation, respectively. The fourth and fifth columns contain the dielectric function when the Coulomb and exchange-correlation kernels are considered, respectively.

B. Case when "type" is set to "PACS"

```
#                 Photo Absorption Cross Section
#      Energy[eV]   NonInteracting   Interacting
       0.000000       0.000000       0.000000
       0.050000       0.000034       0.000012
      ..............................................
```

The first column contains the energy of the excitation spectra. The second and third columns contain the photoadsorption cross-section in the independent particle approximation and in the interacting system, respectively.

11.1.5 Samples

11.1.5.1 Dielectric function of the Si crystal

The folder "sample/lr-tddft/ SiBulk" contains input files for calculating the dielectric function of bulk Si. First, enter the folder "scf" and perform an electronic-structure calculation to obtain the charge density of the ground state. Then, enter the folder "LRC" and perform a calculation of the excitation spectrum in which the "LRC" model is adopted as fxc.

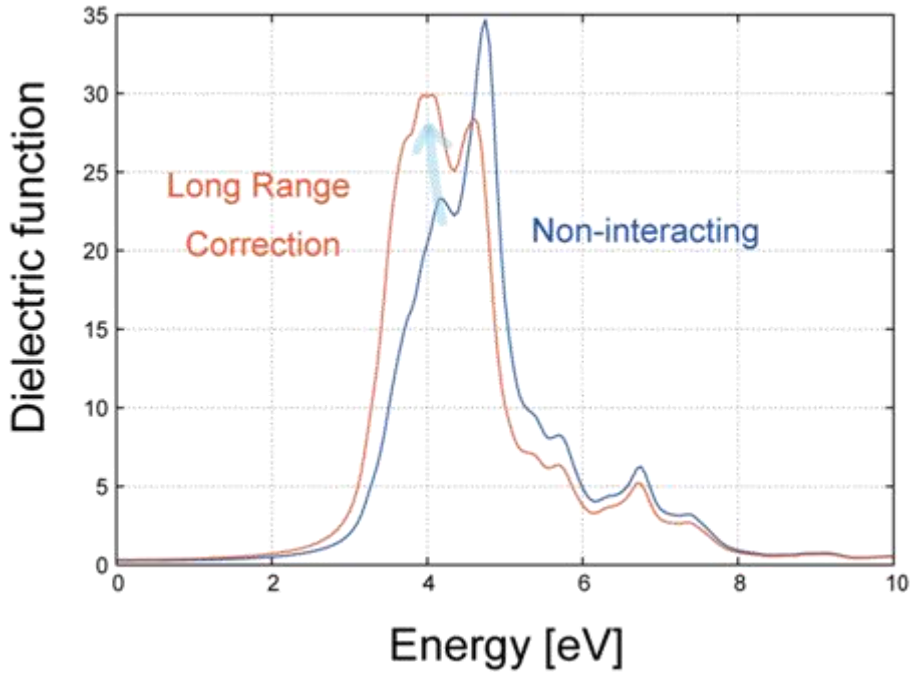


Figure 11.1 Excitation spectra of bulk Si in the LRC model. The blue curve is the result using the independent particle approximation.

The blue and red curves in Fig. 5 show the excitation spectra of bulk Si in the independent particle approximation and in the LRC model, respectively. These values will be found as the imaginary parts of the dielectric functions in the file "spectrum.data." The figure indicates that the long-range correction enhances the first peak. Note that the peak positions do not significantly change in TDDFT, indicating weak Coulomb interactions between the delocalized electrons in the crystal.

11.1.5.2 Photoadsorption cross-section of C_6H_6

The folder "sample/lr-tddft/C6H6" contains input files for calculating the photoadsorption spectra of an isolated C_6H_6 molecule. First, enter the folder "scf" and perform an electronic-structure calculation to obtain the charge density of the ground state. Then, enter the folder "ALDA" and perform a calculation of the excitation spectrum in which the "ALDA" model is adopted as fxc.

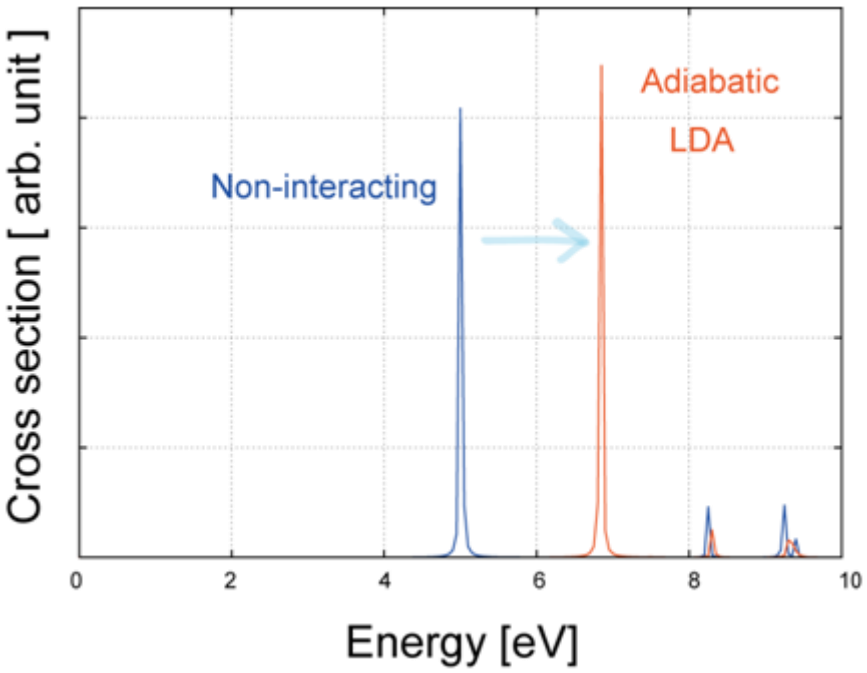


Figure 11.2 Photoadsorption cross-sections of an isolated C_6H_6 molecule. The blue curve is the result using the independent particle approximation.

The blue and red curves in Fig. 6 show the excitation spectra of the C_6H_6 molecule in the independent particle approximation and in the ALDA model, respectively. These values will be found in the file "spectrum.data." The figure indicates that the first peak position shifts to a higher energy value, suggesting that the gap is increased in TDDFT.

11.1.6 Notes

- Reduction of k-points using symmetry operations is not supported. Therefore, choose the symmetry operation "E" in the structure block.
- When "equation=BS," only nonmagnetic systems can be treated.