ABACUS

ABACUS

Jun 01, 2023

QUICK START

ABACUS (Atomic-orbital Based Ab-initio Computation at UStc) is an open-source computer code package based on density functional theory (DFT). The package utilizes both plane wave and numerical atomic basis sets with the usage of norm-conserving pseudopotentials to describe the interactions between nuclear ions and valence electrons. ABACUS supports LDA, GGA, meta-GGA, and hybrid functionals. Apart from single-point calculations, the package allows geometry optimizations and ab-initio molecular dynamics with various ensembles. The package also provides a variety of advanced functionalities for simulating materials, including the DFT+U, VdW corrections, and implicit solvation model, etc. In addition, ABACUS strives to provide a general infrastructure to facilitate the developments and applications of novel machine-learning-assisted DFT methods (DeePKS, DP-GEN, DeepH, etc.) in molecular and material simulations.

CHAPTER

ONE

EASY INSTALLATION

This guide helps you install ABACUS with basic features. **For DeePKS, DeePMD and Libxc support, or building with `make`, please refer to *the advanced installation guide*** after going through this page. We recommend building ABACUS with `cmake` to avoid dependency issues. We recommend compiling ABACUS(and possibly its requirements) from the source code using the latest compiler for the best performace. You can also deploy ABACUS **without building** by *Docker* or *conda*. Please note that ABACUS only supports Linux; for Windows users, please consider using WSL or docker.

1.1 Prerequisites

To compile ABACUS, please make sure that the following prerequisites are present:

- CMake >= 3.16 .
- C++ compiler, supporting C++11. You can use Intel® C++ compiler or GCC.

  GCC version 5 or later is always required. Intel compilers also use GCC headers and libraries(ref).
- MPI library. The recommended versions are Intel MPI, MPICH or Open MPI.
- Fortran compiler if you are building `BLAS`, `LAPACK`, `ScaLAPACK`, and `ELPA` from source file. You can use Intel® Fortran Compiler or GFortran.
- BLAS. You can use OpenBLAS.
- LAPACK.
- FFTW3.

These requirements support the calculation of plane-wave basis in ABACUS. For LCAO basis calculation, additional components are required:

- ScaLAPACK.
- CEREAL.
- ELPA >= 2017 (optional).

1.2 Install requirements

Some of these packages can be installed with popular package management system, such as `apt` and `yum`:

```
sudo apt update && sudo apt install -y libopenblas-openmp-dev liblapack-dev␣
↪libscalapack-mpi-dev libelpa-dev libfftw3-dev libcereal-dev libxc-dev g++ make␣
↪cmake bc git
```

> Installing ELPA by apt only matches requirements on Ubuntu 22.04. For earlier linux distributions, you should build ELPA from source.

We recommend Intel® oneAPI toolkit (former Intel® Parallel Studio) as toolchain. The Intel® oneAPI Base Toolkit contains Intel® oneAPI Math Kernel Library (aka `MKL`), including `BLAS`, `LAPACK`, `ScaLAPACK` and `FFTW3`. The Intel® oneAPI HPC Toolkit contains Intel® MPI Library, and C++ compiler(including MPI compiler).

> Please note that building `elpa` with a different MPI library may cause conflict. Don't forget to set environment variables before you start! `cmake` will use Intel MKL if the environment variable `MKLROOT` is set.

Please refer to our guide on installing requirements.

1.3 Get ABACUS source code

Of course a copy of ABACUS source code is required, which can be obtained via one of the following choices:

- Clone the whole repo with git: `git clone https://github.com/deepmodeling/abacus-develop.git`
- Clone the minimum required part of repo: `git clone https://github.com/deepmodeling/abacus-develop.git --depth=1`
- Download the latest source code without git: `wget https://github.com/deepmodeling/abacus-develop/archive/refs/heads/develop.zip`
- Get the source code of a stable version here
- If you have connection issues accessing GitHub, please try out our official Gitee repo: e.g. `git clone https://gitee.com/deepmodeling/abacus-develop.git`

1.4 Configure

The basic command synopsis is:

```
cd abacus-develop
cmake -B build [-D <var>=<value>] ...
```

Here, 'build' is the path for building ABACUS; and '-D' is used for setting up some variables for CMake indicating optional components or requirement positions.

- `CMAKE_INSTALL_PREFIX`: the path of ABACUS binary to install; `/usr/local/bin/abacus` by default
- Compilers
  - `CMAKE_CXX_COMPILER`: C++ compiler; usually `g++`(GNU C++ compiler) or `icpc`(Intel C++ compiler). Can also set from environment variable `CXX`. It is OK to use MPI compiler here.
  - `MPI_CXX_COMPILER`: MPI wrapper for C++ compiler; usually `mpicxx` or `mpiicpc`(for Intel MPI).

- Requirements: Unless indicated, CMake will try to find under default paths.
  - `MKLROOT`: If environment variable `MKLROOT` exists, `cmake` will take MKL as a preference, i.e. not using `LAPACK`, `ScaLAPACK` and `FFTW`. To disable MKL, unset environment variable `MKLROOT`, or pass `-DMKLROOT=OFF` to `cmake`.
  - `LAPACK_DIR`: Path to OpenBLAS library `libopenblas.so`(including BLAS and LAPACK)
  - `SCALAPACK_DIR`: Path to ScaLAPACK library `libscalapack.so`
  - `ELPA_DIR`: Path to ELPA install directory; should be the folder containing 'include' and 'lib'.

    Note: If you install ELPA from source, please add a symlink to avoid the additional include file folder with version name: `ln -s elpa/include/elpa-2021.05.002/elpa elpa/include/elpa`. This is a known behavior of ELPA.
  - `FFTW3_DIR`: Path to FFTW3.
  - `CEREAL_INCLUDE_DIR`: Path to the parent folder of `cereal/cereal.hpp`. Will download from GitHub if absent.
  - `Libxc_DIR`: (Optional) Path to Libxc.

    Note: Building Libxc from source with Makefile does NOT support using it in CMake here. Please compile Libxc with CMake instead.
- Components: The values of these variables should be 'ON', '1' or 'OFF', '0'. The default values are given below.
  - `ENABLE_LCAO=ON`: Enable LCAO calculation. If SCALAPACK, ELPA or CEREAL is absent and only require plane-wave calculations, the feature of calculating LCAO basis can be turned off.
  - `ENABLE_LIBXC=OFF`: *Enable Libxc* to suppport variety of functionals. If `Libxc_DIR` is defined, `ENABLE_LIBXC` will set to 'ON'.
  - `USE_OPENMP=ON`: Enable OpenMP support. Building ABACUS without OpenMP is not fully tested yet.
  - `BUILD_TESTING=OFF`: *Build unit tests*.
  - `ENABLE_COVERAGE=OFF`: Build ABACUS executable supporting *coverage analysis*. This feature has a drastic impact on performance.
  - `ENABLE_ASAN=OFF`: Build with Address Sanitizer. This feature would help detecting memory problems. Only supports GCC.
  - `USE_ELPA=ON`: Use ELPA library in LCAO calculations. If this value is set to OFF, ABACUS can be compiled without ELPA library.

Here is an example:

```
CXX=mpiicpc cmake -B build -DCMAKE_INSTALL_PREFIX=~/abacus -DELPA_DIR=~/elpa-2016.05.
↪004/build -DCEREAL_INCLUDE_DIR=~/cereal/include
```

1.5 Build and Install

After configuring, build and install by:

```
cmake --build build -j`nproc`
cmake --install build
```

You can change the number after `-j` on your need: set to the number of CPU cores(`nproc`) to reduce compilation time.

1.6 Run

If ABACUS is installed into a custom directory using `CMAKE_INSTALL_PREFIX`, please add it to your environment variable `PATH` to locate the correct executable.

```
export PATH=/my-install-dir/:$PATH
```

Please set OpenMP threads by setting environment variable:

```
export OMP_NUM_THREADS=1
```

Enter a directory containing a `INPUT` file. Please make sure structure, pseudo potential, or orbital files indicated by `INPUT` is at the correct location.

```
cd abacus-develop/examples/force/pw_Si2
```

Use 4 MPI processes to run, for example:

```
mpirun -n 4 abacus
```

> The total thread count(i.e. OpenMP per-process thread count * MPI process count) should not exceed the number of cores in your machine.

Please refer to *hands-on guide* for more instructions.

> Note: Some Intel CPU has a feature named Hyper-Threading(HT). This feature enables one physical core switch fastly between two logical threads. It would benefits from I/O bound tasks: when a thread is blocked by I/O, the CPU core can work on another thread. However, it helps little on CPU bound tasks, like ABACUS and many other scientific computing softwares. We recommend using the physical CPU core number. To determine if HT is turned on, execute `lscpu | grep 'per core'` and see if 'Thread(s) per core' is 2.

1.7 Container Deployment

> Please note that containers target at developing and testing, but not massively parallel computing for production. Docker has a bad support to MPI, which may cause performance degradation.

We've built a ready-for-use version of ABACUS with docker here. For a quick start: pull the image, prepare the data, run container. Instructions on using the image can be accessed in Dockerfile. A mirror is available by `docker pull registry.dp.tech/deepmodeling/abacus`.

We also offer a pre-built docker image containing all the requirements for development. Please refer to our Package Page.

The project is ready for VS Code development container. Please refer to Developing inside a Container. Choose `Open a Remote Window -> Clone a Repository in Container Volume` in VS Code command palette, and put the git address of `ABACUS` when prompted.

For online development environment, we support GitHub Codespaces: Create a new Codespace

We also support Gitpod: Open in Gitpod

1.8 Install by conda

Conda is a package management system with separated environment, not requiring system privileges. A pre-built ABACUS binary with all requirements is available at deepmodeling conda channel. Install ABACUS by the commands below:

```
# We recommend installing ABACUS in a new environment to avoid potential conflicts:
conda create -n abacus_env abacus -c deepmodeling -c conda-forge
conda activate abacus_env
# ABACUS is ready to go:
mpirun -n 4 abacus
```

For more details on building a conda package of ABACUS, please refer to the *conda recipe file* online.

CHAPTER

TWO

TWO QUICK EXAMPLES

2.1 Running SCF Calculation

2.1.1 A quick LCAO example

ABACUS is well known for its support of LCAO (Linear Combination of Atomic Orbital) basis set in calculating periodic condensed matter systems, so it's a good choice to start from a LCAO example of self-consistent field (SCF) calculation. Here, FCC MgO has been chosen as a quick start example. The default name of a structure file in ABACUS is `STRU`. The `STRU` file for FCC MgO in a LCAO calculation is shown below:

```
#This is the atom file containing all the information
#about the lattice structure.

ATOMIC_SPECIES
Mg 24.305  Mg_ONCV_PBE-1.0.upf  # element name, atomic mass, pseudopotential file
O  15.999 O_ONCV_PBE-1.0.upf

NUMERICAL_ORBITAL
Mg_gga_8au_100Ry_4s2p1d.orb
O_gga_8au_100Ry_2s2p1d.orb

LATTICE_CONSTANT
1.8897259886                 # 1.8897259886 Bohr =  1.0 Angstrom

LATTICE_VECTORS
4.25648 0.00000 0.00000
0.00000 4.25648 0.00000
0.00000 0.00000 4.25648

ATOMIC_POSITIONS
Direct                  #Cartesian(Unit is LATTICE_CONSTANT)
Mg                      #Name of element
0.0                     #Magnetic for this element.
4                       #Number of atoms
0.0  0.0  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
0.0  0.5  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.0  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.5  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
O                       #Name of element
0.0                     #Magnetic for this element.
4                       #Number of atoms
0.5  0.0  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.5  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
```

(continues on next page)

(continued from previous page)

```
0.0   0.0   0.5   0 0 0     #x,y,z, move_x, move_y, move_z
0.0   0.5   0.0   0 0 0     #x,y,z, move_x, move_y, move_z
```

Next, the `INPUT` file is required, which sets all key parameters to direct ABACUS how to calculte and what to output:

```
INPUT_PARAMETERS
suffix                  MgO
ntype                   2
pseudo_dir              ./
orbital_dir                ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-4                # Rydberg
basis_type              lcao
calculation             scf                   # this is the key parameter telling abacus␣
↪to do a scf calculation
```

The pseudopotential files of `Mg_ONCV_PBE-1.0.upf` and `O_ONCV_PBE-1.0.upf` should be provided under the directory of `pseudo_dir`, and the orbital files `Mg_gga_8au_100Ry_4s2p1d.orb` and `O_gga_8au_100Ry_2s2p1d.orb` under the directory of `orbital_dir`. The pseudopotential and orbital files can be downloaded from the ABACUS website.

The final mandatory input file is called `KPT`, which sets the reciprocal space k-mesh. Below is an example:

```
K_POINTS
0
Gamma
4 4 4 0 0 0
```

After all the above input files have been set, one should be able to run the first quick example. The simplest way is to use the command line, e.g.:

```
mpirun -np 2 abacus
```

The main output information is stored in the file `OUT.MgO/running_scf.log`, which starts with

```
                             WELCOME TO ABACUS v3.2

               'Atomic-orbital Based Ab-initio Computation at UStc'

                     Website: http://abacus.ustc.edu.cn/

    Version: Parallel, in development
    Processor Number is 2
    Start Time is Mon Oct 24 01:47:54 2022

 ------------------------------------------------------------------------------------

 READING GENERAL INFORMATION
                           global_out_dir = OUT.MgO/
                           global_in_card = INPUT
                               pseudo_dir = 
                              orbital_dir = 
                                    DRANK = 1
                                    DSIZE = 2
                                   DCOLOR = 1
                                    GRANK = 1
                                    GSIZE = 1
```

(continues on next page)

(continued from previous page)

```
 The esolver type has been set to : ksdft_lcao




 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 |                                                                     |
 | Reading atom information in unitcell:                               |
 | From the input file and the structure file we know the number of    |
 | different elments in this unitcell, then we list the detail         |
 | information for each element, especially the zeta and polar atomic |
 | orbital number for each element. The total atom number is counted. |
 | We calculate the nearest atom distance for each atom and show the   |
 | Cartesian and Direct coordinates for each atom. We list the file    |
 | address for atomic orbitals. The volume and the lattice vectors     |
 | in real and reciprocal space is also shown.                         |
 |                                                                     |
 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

......
```

If ABAUCS finishes successfully, the total energy will be output in `OUT.MgO/running_scf.log`:

```
 --------------------------------------------
 !FINAL_ETOT_IS -7663.897267807250 eV
 --------------------------------------------
```

2.1.2 A quick PW example

In order to run a SCF calculation with PW (Plane Wave) basis set, one has only to change the tag `basis_type` from `lcao` to `pw` in the `INPUT` file, and no longer needs to provide orbital files under `NUMERICAL_ORBITAL` in the `STRU` file.

The `INPUT` file follows as:

```
INPUT_PARAMETERS
suffix                  MgO
ntype                   2
pseudo_dir              ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-4                # Rydberg
basis_type              pw              # changes the type of basis set
calculation             scf                # this is the key parameter telling abacus␣
↪to do a scf calculation
```

And the `STRU` file will be:

```
#This is the atom file containing all the information
#about the lattice structure.

ATOMIC_SPECIES
Mg 24.305  Mg_ONCV_PBE-1.0.upf  # element name, atomic mass, pseudopotential file
O  15.999 O_ONCV_PBE-1.0.upf
```

(continues on next page)

(continued from previous page)

```
LATTICE_CONSTANT
1.8897259886                 # 1.8897259886 Bohr =  1.0 Angstrom

LATTICE_VECTORS
4.25648 0.00000 0.00000
0.00000 4.25648 0.00000
0.00000 0.00000 4.25648

ATOMIC_POSITIONS
Direct                  #Cartesian(Unit is LATTICE_CONSTANT)
Mg                      #Name of element
0.0                     #Magnetic for this element.
4                       #Number of atoms
0.0  0.0  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
0.0  0.5  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.0  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.5  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
O                       #Name of element
0.0                     #Magnetic for this element.
4                       #Number of atoms
0.5  0.0  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
0.5  0.5  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.0  0.0  0.5  0 0 0    #x,y,z, move_x, move_y, move_z
0.0  0.5  0.0  0 0 0    #x,y,z, move_x, move_y, move_z
```

Use the same pseudopotential and KPT files as the above LCAO example. The final total energy will be output:

```
 --------------------------------------------
 !FINAL_ETOT_IS -7665.688319476949 eV
 --------------------------------------------
```

2.2 Running Geometry Optimization

In order to run a full geometry optimization in ABACUS, the tag `calculation` in `INPUT` should be set to `cell-relax`. In addition, the convergence criteria for atomics force and cell stress can be set through the tags `force_thr_ev` and `stress_thr`, respectively. The maximum number of ionc steps is controlled by `relax_nmax`.

2.2.1 A quick LCAO example

The `INPUT` is provided as follows:

```
INPUT_PARAMETERS
suffix                  MgO
ntype                   2
nelec                   0.0
pseudo_dir              ./
orbital_dir             ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-4                # Rydberg
basis_type              lcao
calculation             cell-relax        # this is the key parameter telling abacus
↪to do a optimization calculation
```

(continues on next page)

(continued from previous page)

```
force_thr_ev                0.01              # the threshold of the force␣
↪convergence, in unit of eV/Angstrom
stress_thr                 5                # the threshold of the stress convergence,␣
↪in unit of kBar
relax_nmax                 100                # the maximal number of ionic iteration␣
↪steps
out_stru                  1
```

Use the same KPT, STRU, pseudopotential, and orbital files as in the above SCF-LCAO example. The final optimized structure can be found in STRU_NOW.cif and OUT.MgO/running_cell-relax.log.

2.2.2 A quick PW example

The INPUT is provided as follows:

```
INPUT_PARAMETERS
suffix                  MgO
ntype                   2
nelec                   0.0
pseudo_dir              ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-4                # Rydberg
basis_type              pw
calculation             cell-relax        # this is the key parameter telling abacus␣
↪to do a optimization calculation
force_thr_ev                0.01              # the threshold of the force␣
↪convergence, in unit of eV/Angstrom
stress_thr                 5                # the threshold of the stress convergence,␣
↪in unit of kBar
relax_nmax                 100                # the maximal number of ionic iteration␣
↪steps
out_stru                  1
```

Use the same KPT, STRU, and pseudopotential files as in the above SCF-PW examples. The final optimized structure can be found in STRU_NOW.cif and OUT.MgO/running_cell-relax.log.

CHAPTER

THREE

BRIEF INTRODUCTION OF THE INPUT FILES

The following files are the central input files for ABACUS. Before executing the program, please make sure these files are prepared and stored in the working directory. Here we give some simple descriptions XXX. For more details, users should consult the Advanced session.

3.1 *INPUT*

The INPUT file contains parameters that control the type of calculation as well as a variety of settings.

Below is an example INPUT file with some of the most important parameters that need to be set:

```
INPUT_PARAMETERS
suffix                  MgO
ntype                   2
pseudo_dir              ./
orbital_dir                ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-4                # Rydberg
basis_type              lcao
calculation             scf                 # this is the key parameter telling abacus␣
↪to do a scf calculation
out_chg                        True
```

The parameter list always starts with key word INPUT_PARAMETERS. Any content before INPUT_PARAMETERS will be ignored.

Any line starting with # or / will also be ignored.

Each parameter value is provided by specifying the name of the input variable and then putting the value after the name, separated by one or more blank characters(space or tab). The following characters ($\leq$ 150) in the same line will be neglected.

Depending on the input variable, the value may be an integer, a real number or a string. The parameters can be given in any order, but only one parameter should be given per line.

Furthermore, if a given parameter name appeared more than once in the input file, only the last value will be taken.

Note: if a parameter name is not recognized by the program, the program will stop with an error message.

In the above example, the meanings of the parameters are:

- suffix : the name of the system, default ABACUS
- ntype : how many types of elements in the unit cell
- pseudo_dir : the directory where pseudopotential files are provided

- `orbital_dir` : the directory where orbital files are provided
- `ecutwfc` : the plane-wave energy cutoff for the wave function expansion (UNIT: Rydberg)
- `scf_thr` : the threshold for the convergence of charge density (UNIT: Rydberg)
- `basis_type` : the type of basis set for expanding the electronic wave functions
- `calculation` : the type of calculation to be performed by ABACUS
- `out_chg` : if true, output thee charge density oon real space grid

For a complete list of input parameters, please consult this *instruction*.

> **Note:** Users cannot change the filename "INPUT" to other names. Boolean paramerters such as `out_chg` can be set by using `True` and `False`, `1` and `0`, or `T` and `F`. It is case insensitive so that other preferences such as `true` and `false`, `TRUE` and `FALSE`, and `t` and `f` for setting boolean values are also supported.

3.2 *STRU*

The structure file contains structural information about the system, e.g., lattice constant, lattice vectors, and positions of the atoms within a unit cell. The positions can be given either in direct or Cartesian coordinates.

An example of the `STRU` file is given as follows :

```
#This is the atom file containing all the information
#about the lattice structure.

ATOMIC_SPECIES
Mg 24.305  Mg_ONCV_PBE-1.0.upf  # element name, atomic mass, pseudopotential file
O  15.999 O_ONCV_PBE-1.0.upf

NUMERICAL_ORBITAL
Mg_gga_8au_100Ry_4s2p1d.orb
O_gga_8au_100Ry_2s2p1d.orb

LATTICE_CONSTANT
1.8897259886                 # 1.8897259886 Bohr =  1.0 Angstrom

LATTICE_VECTORS
4.25648 0.00000 0.00000
0.00000 4.25648 0.00000
0.00000 0.00000 4.25648

ATOMIC_POSITIONS
Direct                   #Cartesian(Unit is LATTICE_CONSTANT)
Mg                       #Name of element
0.0                      #Magnetic for this element.
4                        #Number of atoms
0.0  0.0  0.0  0 0 0     #x,y,z, move_x, move_y, move_z
0.0  0.5  0.5  0 0 0     #x,y,z, move_x, move_y, move_z
0.5  0.0  0.5  0 0 0     #x,y,z, move_x, move_y, move_z
0.5  0.5  0.0  0 0 0     #x,y,z, move_x, move_y, move_z
O                        #Name of element
0.0                      #Magnetic for this element.
4                        #Number of atoms
0.5  0.0  0.0  0 0 0     #x,y,z, move_x, move_y, move_z
0.5  0.5  0.5  0 0 0     #x,y,z, move_x, move_y, move_z
```

(continues on next page)

(continued from previous page)

```
0.0   0.0   0.5   0 0 0     #x,y,z, move_x, move_y, move_z
0.0   0.5   0.0   0 0 0     #x,y,z, move_x, move_y, move_z
```

Note: users may choose a different name for their structure file using the keyword `stru_file`. The order of the pseudopotential file list and the numerical orbital list (if LCAO is applied) MUST be consistent with that of the atomic type given in `ATOMIC_POSITIONS`.

For a more detailed description of STRU file, please consult *here*.

3.3 *KPT*

This file contains information of the kpoint grid setting for the Brillouin zone sampling.

An example of the `KPT` file is given below:

```
K_POINTS
0
Gamma
4 4 4 0 0 0
```

Note: users may choose a different name for their k-point file using keyword `kpoint_file`

For a more detailed description, please consult *here*.

- The pseudopotential files

  Norm-conserving pseudopotentials are used in ABACUS, in the UPF file format.The filename of each element's pseudopotential needs to be specified in the STRU file, together with the directory of the pseudopotential files unless they are already present in the working directory.

  More information on pseudopotentials is given *here*.

- The numerical orbital files

  This part is only required in LCAO calculations. The filename for each element's numerical orbital basis needs to be specified in the STRU file, together with the directory of the orbital files unless they are already present in the working directory. ABACUS provides numerical atomic basis sets of different accuracy levels for most elements commonly used. Users can download these basis sets from the website. Moreover, users can generate numerical atomic orbitals by themselves, and the procedure is provided in this *short introduction*.

CHAPTER

FOUR

ADVANCED INSTALLATION OPTIONS

This guide helps you install ABACUS with advanced features. Please make sure to read the *easy-installation guide* before.

4.1 Build with Libxc

ABACUS use exchange-correlation functionals by default. However, for some functionals (such as HSE hybrid functional), Libxc is required.

Dependency: Libxc >= 5.1.7 .

Note: Building Libxc from source with Makefile does NOT support using it in CMake here. Please compile Libxc with CMake instead.

If Libxc is not installed in standard path (i.e. installed with a custom prefix path), you can set `Libxc_DIR` to the corresponding directory.

```
cmake -B build -DLibxc_DIR=~/libxc
```

4.2 Build with DeePKS

If DeePKS feature is required for DeePKS-kit, the following prerequisites and steps are needed:

- C++ compiler, supporting **C++14** (GCC >= 5 is sufficient)
- CMake >= 3.18
- LibTorch with cxx11 ABI supporting CPU
- Libnpy

```
cmake -B build -DENABLE_DEEPKS=1 -DTorch_DIR=~/libtorch/share/cmake/Torch/ -Dlibnpy_
↪INCLUDE_DIR=~/libnpy/include
```

CMake will try to download Libnpy if it cannot be found locally.

4.3 Build with DeePMD-kit

> Note: This part is only required if you want to load a trained DeeP Potential and run molecular dynamics with that. To train the DeeP Potential with DP-GEN, no extra prerequisite is needed and please refer to this page for ABACUS interface with DP-GEN.

If the Deep Potential model is employed in Molecule Dynamics calculations, the following prerequisites and steps are needed:

- DeePMD-kit
- TensorFlow

```
cmake -B build -DDeePMD_DIR=~/deepmd-kit -DTensorFlow_DIR=~/tensorflow
```

> `deepmd_c`/`deepmd_cc` and `tensorflow_cc` libraries would be called according to `DeePMD_DIR` and `TensorFlow_DIR`, which is showed in detail in this page.

4.4 Build with LibRI and LibComm

The new EXX implementation depends on two external libraries:

- LibRI
- LibComm

These two libraries are added as submodules in the deps folder. Set `-DENABLE_LIBRI=ON` to build with these two libraries.

If you prefer using manually downloaded libraries, set `-DENABLE_LIBRI=ON, -DGIT_SUBMODULE=OFF`, and provide `-DLIBRI_DIR=${path to your LibRI folder} -DLIBCOMM_DIR=${path to your LibComm folder}`. Remember to make sure the commit numbers match with those in the deps folder.

4.5 Build Unit Tests

To build tests for ABACUS, define `BUILD_TESTING` flag. You can also specify path to local installation of Googletest by setting `GTEST_DIR` flags. If not found in local, the configuration process will try to download it automatically.

```
cmake -B build -DBUILD_TESTING=1
```

After building and installing, unit tests can be performed with `ctest`.

To run a subset of unit test, use `ctest -R <test-match-pattern>` to perform tests with name matched by given pattern.

4.6 Build with CUDA support

4.6.1 Extra prerequisites

- CUDA-Toolkit

To build NVIDIA GPU support for ABACUS, define `USE_CUDA` flag. You can also specify path to local installation of CUDA Toolkit by setting `CUDA_TOOLKIT_ROOT_DIR` flags.

```
cmake -B build -DUSE_CUDA=1
```

4.7 Build math library from source

> Note: This flag is **enabled by default**. It will get better performance than the standard implementation on `gcc` and `clang`. But it **will be disabled** when using `Intel Compiler` since the math functions will get wrong results and the performance is also unexpectly poor.

To build math functions from source code, instead of using c++ standard implementation, define `USE_ABACUS_LIBM` flag.

Currently supported math functions: `sin`, `cos`, `sincos`, `exp`, `cexp`

```
cmake -B build -DUSE_ABACUS_LIBM=1
```

4.8 Build ABACUS with make

> Note: We suggest using CMake to configure and compile.

To compile the ABACUS program using legacy `make`, users need to specify the location of the compilers, headers and libraries in `source/Makefile.vars`:

```
# This is the Makefile of ABACUS API
#======================================================================
# Users set
#======================================================================
CXX = mpiicpc
# mpiicpc:   compile intel parallel version
# icpc:      compile intel sequential version
# make: ELPA_DIR, ELPA_INCLUDE_DIR, CEREAL_DIR must also be set.
# make pw: nothing need to be set except LIBXC_DIR
#
# mpicxx:    compile gnu parallel version
# g++:       compile gnu sequential version
# make: FFTW_DIR, OPENBLAS_LIB_DIR, SCALAPACK_LIB_DIR, ELPA_DIR, ELPA_INCLUDE_DIR,
↪CEREAL_DIR must also be set.
# make pw: FFTW_DIR, OPENBLAS_LIB_DIR must be set.

# GPU = OFF  #We do not support GPU yet
# OFF:  do not use GPU
# CUDA: use CUDA
#======================================================================
```

(continues on next page)

(continued from previous page)

```
#---------------------  FOR INTEL COMPILER  ----------------------------
## ELPA_DIR           should contain an include folder and lib/libelpa.a
## CEREAL_DIR         should contain an include folder.
#----------------------------------------------------------------------

ELPA_DIR      = /usr/local/include/elpa-2021.05.002
ELPA_INCLUDE_DIR = ${ELPA_DIR}/elpa

CEREAL_DIR    = /usr/local/include/cereal


##-------------------  FOR GNU COMPILER  ------------------------------
## FFTW_DIR           should contain lib/libfftw3.a.
## OPENBLAS_LIB_DIR   should contain libopenblas.a.
## SCALAPACK_LIB_DIR should contain libscalapack.a
## All three above will only be used when CXX=mpicxx or g++
## ELPA_DIR           should contain an include folder and lib/libelpa.a
## CEREAL_DIR         should contain an include folder.
##---------------------------------------------------------------------

# FFTW_DIR = /public/soft/fftw_3.3.8
# OPENBLAS_LIB_DIR   = /public/soft/openblas/lib
# SCALAPACK_LIB_DIR  = /public/soft/openblas/lib

# ELPA_DIR      = /public/soft/elpa_21.05.002
# ELPA_INCLUDE_DIR = ${ELPA_DIR}/include/elpa-2021.05.002

# CEREAL_DIR    = /public/soft/cereal


##-------------------  OPTIONAL LIBS  ---------------------------------
## To use DEEPKS: set LIBTORCH_DIR and LIBNPY_DIR
## To use LIBXC:  set LIBXC_DIR which contains include and lib/libxc.a (>5.1.7)
## To use DeePMD: set DeePMD_DIR and TensorFlow_DIR
## To use LibRI:  set LIBRI_DIR and LIBCOMM_DIR
##---------------------------------------------------------------------

# LIBTORCH_DIR  = /usr/local
# LIBNPY_DIR    = /usr/local

# LIBXC_DIR                   = /public/soft/libxc

# DeePMD_DIR = ${deepmd_root}
# TensorFlow_DIR = ${tensorflow_root}

# LIBRI_DIR     = /public/software/LibRI
# LIBCOMM_DIR   = /public/software/LibComm

##---------------------------------------------------------------------
# NP = 14 # It is not supported. use make -j14 or make -j to parallelly compile
# DEBUG = OFF
# Only for developers
# ON:   use gnu compiler and check segmental defaults
# OFF:  nothing
#======================================================================
```

For example, below is a case where the Intel C++ compiler, Intel MPI and CEREAL are used, along with Intel MKL library. The file Makefile.vars can be set as follows:

```
CXX = mpiicpc #(or CXX = icpc)
ELPA_DIR      = /public/soft/elpa_21.05.002
ELPA_INCLUDE_DIR = ${ELPA_DIR}/include/elpa-2021.05.002
CEREAL_DIR    = /public/soft/cereal
```

When `CXX=mpiicpc`, a parallel version will be compiled. When `CXX=icpc`, a sequential version will be compiled.

Another example is where the Gnu C++ compiler, MPICH, OPENBLAS, ScaLAPACK, ELPA and CEREAL are used:

```
CXX = mpicxx/g++
FFTW_DIR = /public/soft/fftw_3.3.8
OPENBLAS_LIB_DIR   = /public/soft/openblas/lib
SCALAPACK_LIB_DIR  = /public/soft/openblas/lib
ELPA_DIR      = /public/soft/elpa_21.05.002
ELPA_INCLUDE_DIR = ${ELPA_DIR}/include/elpa-2021.05.002
CEREAL_DIR    = /public/soft/cereal
```

When `CXX=mpicxx`, a parallel version will be compiled. When `CXX=g++`, a sequential version will be compiled.

Except modifying `Makefile.vars`, you can also directly use

```
make CXX=mpiicpc ELPA_DIR=/public/soft/elpa_21.05.002 \
ELPA_INCLUDE_DIR=${ELPA_DIR}/include/elpa-2021.05.002 \
CEREAL_DIR=/public/soft/cereal
```

ABACUS now support full version and pw version. Use `make` or `make abacus` to compile full version which supports LCAO calculations. Use `make pw` to compile pw version which only supports pw calculations. For pw version, `make pw CXX=mpiicpc`, you do not need to provide any libs. For `make pw CXX=mpicxx`, you need provide `FFTW_DIR` and `OPENBLAS_LIB_DIR`.

Besides, libxc and deepks are optional libs to compile abacus. They will be used when `LIBXC_DIR` is defined like

```
LIBXC_DIR = /public/soft/libxc
```

or `LIBTORCH_DIR` and `LIBNPY_DIR` like

```
LIBTORCH_DIR  = /usr/local
LIBNPY_DIR    = /usr/local
```

After modifying the `Makefile.vars` file, execute `make` or `make -j12` to build the program.

After the compilation finishes without error messages (except perhaps for some warnings), an executable program `ABACUS.mpi` will be created in directory `bin/`.

4.8.1 Add Libxc Support

The program compiled using the above instructions do not link with LIBXC and use exchange-correlation functionals as written in the ABACUS program. However, for some functionals (such as HSE hybrid functional), LIBXC is required.

To compile ABACUS with LIBXC, you need to define `LIBXC_DIR` in the file `Makefile.vars` or use

```
make LIBXC_DIR=/pulic/soft/libxc
```

directly.

4.8.2 Add DeePKS Support

To compile ABACUS with DEEPKS, you need to define `LIBTORCH_DIR` and `LIBNPY_DIR` in the file `Makefile.vars` or use

```
make LIBTORCH_DIR=/opt/libtorch/ LIBNPY_DIR=/opt/libnpy/
```

directly.

4.8.3 Add DeePMD-kit Support

> Note: This part is only required if you want to load a trained DeeP Potential and run molecular dynamics with that. To train the DeeP Potential with DP-GEN, no extra prerequisite is needed and please refer to this page for ABACUS interface with DP-GEN.

To compile ABACUS with DeePMD-kit, you need to define `DeePMD_DIR` and `TensorFlow_DIR` in the file `Makefile.vars` or use

```
make DeePMD_DIR=~/deepmd-kit TensorFlow_DIR=~/tensorflow
```

directly.

> `deepmd_c/deepmd_cc` and `tensorflow_cc` libraries would be called according to `DeePMD_DIR` and `TensorFlow_DIR`, which is showed in detail in this page.

4.8.4 Add LibRI and LibComm Support

To use new EXX, you need two libraries: LibRI and LibComm and need to define `LIBRI_DIR` and `LIBCOMM_DIR` in the file `Makefile.vars` or use

```
make LIBRI_DIR=/public/software/LibRI LIBCOMM_DIR=/public/software/LibComm
```

directly.

CHAPTER

FIVE

RUNNING SCF

5.1 Initializing SCF

Good initializing would abate the number of iteration steps in SCF. Charge density should be initialed for constructing the initial hamiltonian operator.

In PW basis, wavefunction should be initialized for iterate diagonalization method. In LCAO basis, wavefunction can be read to calculate initial charge density. The wavefunction itself does not have to be initialized.

5.1.1 Charge Density

`init_chg` is used for choosing the method of charge density initialization.

- `atomic` : initial charge density by atomic charge density from pseudopotential file under keyword `PP_RHOATOM`
- `file` : initial charge density from files produced by previous calculations with *`out_chg 1`*.

5.1.2 Wave function

`init_wfc` is used for choosing the method of wavefunction coefficient initialization.

When `basis_type=pw`, setting of `random` and `atomic` are supported. Atomic wave function is read from pseudopotential file under keyword `PP_PSWFC`, if setting is `atomic` and number of band of atomic wavefunction less than `nbands` in INPUT file, the extra bands will be initialed by random.

When `basis_type=lcao`, we further support reading of initial wavefunction by setting `init_wfc` to `file`. In LCAO code, wave function is used to initialize density matrix and real-space charge density. For such purpose, a file containing wavefunction must be prepared. Such files can be generated from previous calculations with *`out_wfc_lcao 1`*.

5.2 Constructing the Hamiltonian

5.2.1 Exchange-Correlation Functionals

In our package, the XC functional can be set explicitly using the `dft_functional` keyword in `INPUT` file. If `dft_functional` is not specified, ABACUS will use the xc functional indicated in the pseudopotential file.

Several common functionals are implemented in ABACUS, such as PZ and PBE. Users can check out this file for a complete list of functionals implemented in ABACUS. Furthermore, if ABACUS is compiled with LIBXC, we also support all the LDA, GGA and meta-GGA functionals provided therein.

Here, we use a simple example calculation for illustration.

1. **Default setting:**

   In the original `INPUT` file, there is no specification of the `dft_functional` keyword. As a result, we use the default option, that is to use the xc functional in the pseudopotential file, `Si.pz-vbc.UPF`. We can take a look at the first few lines of the `<PP_HEADER>` section from the pseudopotential file:

   ```
   <PP_HEADER>
   0                   Version Number
   Si                   Element
   NC                  Norm - Conserving pseudopotential
       F                  Nonlinear Core Correction
   SLA  PZ   NOGX NOGC   PZ   Exchange-Correlation functional
   ```

   From the line commented 'Exchange-Correlation functional', we see that this pseudopotential is generated using PZ functional. As a result, if we run ABACUS with the original setting, PZ functional will be used.

   > Note : for systems with multiple elements, if no `dft_functional` is specified, users should make sure that all pseudopotentials are using the same functional. Otherwise, the type of xc functional should be specified explicitly.

2. **Using PBE**

   On the other hand, users might also explicitly specify the xc functional through `dft_functional` parameter. For example, to use PBE functional, add the following line to `INPUT` file and rerun the calculation:

   ```
   dft_functional PBE
   ```

3. **More functionals from LIBXC**

   ABACUS has its own implementation of the PBE functional as well as a few others, but our list is far from comprehensive. However, if ABACUS is compiled with LIBXC, we also support all the LDA, GGA and meta-GGA functionals provided therein.

   For this part, users should compile the ABACUS code with LIBXC linked (version 5.1.7 or higher).

   To use SCAN functional, make the following modification to the `INPUT` file:

   ```
   dft_functional SCAN
   ```

   Note that in the case of PBE and SCAN, we are using 'short-hand' names to represent the entire functional, which is made up of individual exchange and correlation components. A complete list of 'short-hand' expressions supported by ABACUS can be found in source code.

   Apart from the 'short-hand' names, ABACUS also allow supplying exchange-correlation functionals as combinations of LIBXC keywords for functional components, joined by plus sign, for example, setting:

   ```
   dft_functional LDA_X_YUKAWA+LDA_C_1D_CSC
   ```

   means we are using the short-range Yukawa attenuated exchange along with the Casula, Sorella & Senatore LDA correlation functional.

   The list of LIBXC keywords can be found on its website.

4. **Temperature-dependent functional**

   In ABACUS, we provide temperature-dependent functionals through LIBXC. For such functionals, the keyword `xc_temperature` (unit is Rydberg) is used to specify the temperature, such as the following:

   ```
   dft_functional LDA_XC_CORRKSDT
   xc_temperature 10
   ```

5. **Hybrid functional**

   ABACUS supports functionals with exact Hartree-Fock exchange in LCAO basis set only. The old INPUT parameter exx_hybrid_type for hybrid functionals has been absorbed into `dft_functional`. Options are `hf` (pure Hartree-Fock), `pbe0`(PBE0), `hse`, and `scan0`(SCAN0) (Note: in order to use HSE or SCAN0 functional, LIBXC is required). Note also that only HSE has been tested while other hybrid functionals have NOT been fully tested yet, and the maximum parallel cpus for running exx is Nx(N+1)/2, with N being the number of atoms. And forces for hybrid functionals are not supported yet.

   More information on the hybrid functional can be found from the section *Exact Exchange* in the list of input variables for more information.

   An example HSE calculation is provided in this directory. Apart from the input files (`INPUT`, `STRU`, `KPT`), we further provide two files: running_scf.log_ref and log_ref, which contains reference for running_scf.log and standard output from the program, respectively.

   In the log_ref file, you will see the repetitive appearance of a piece of warning message:

   ```
   The angular momentum larger than 4 (g orbitals) may be error about eggbox.
   Check file ./module_basis/module_ao/ORB_atomic_lm.cpp line 272
   ```

   This is normal and it will not affect the results of calculation.

5.2.2 DFT+*U*

Conventional functionals, e.g., L(S)DA and GGAs, encounter failures in strongly correlated systems, usually characterized by partially filled *d*/*f* shells. These include transition metals ™ and their oxides, rare-earth compounds, and actinides, to name a few, where L(S)DA/GGAs typically yield quantitatively or even qualitatively wrong results. To address this failure, an efficient and successful method named DFT+*U*, which inherits the efficiency of L(S)DA/GGA but gains the strength of the Hubbard model in describing the physics of strongly correlatedsystems, has been developed.

Now the DFT+*U* method is accessible in ABACUS. The details of the DFT+*U* method could be found in this paper. It should be noted that the DFT+*U* works only within the NAO scheme, which means that the value of the keyword `basis_type` must be lcao when DFT+*U* is called. To turn on DFT+*U*, users need to set the value of the `dft_plus_u` keyword in the `INPUT` file to be 1. All relevant parmeters used in DFT+*U* calculations are listed in the *DFT+U correction* part of the *list of keywords*.

Examples of DFT+*U* calculations are provided in this directory.

5.3 Solving the Hamiltonian

5.3.1 Explicit Diagonalization

Method of explicit solving KS-equation can be chosen by variable “ks_solver” in INPUT file.

When “basis_type = pw”, `ks_solver` can be `cg` or `dav`. The default setting `cg` is recommended, which is band-by-band conjugate gradient diagonalization method. There is a large probability that the use of setting of `dav` , which is block Davidson diagonalization method, can be tried to improve performance.

When “basis_type = lcao”, `ks_solver` can be `genelpa` or `scalapack_gvx`. The default setting `genelpa` is recommended, which is based on ELPA (EIGENVALUE SOLVERS FOR PETAFLOP APPLICATIONS) (https://elpa.mpcdf.mpg.de/) and the kernel is auto choosed by GENELPA(https://github.com/pplab/GenELPA), usually faster than the setting of “scalapack_gvx”, which is based on ScaLAPACK(Scalable Linear Algebra PACKage)

5.3.2 Stochasic DFT

We support stochastic DFT calculation (SDFT) or mixed stochastic-deterministic DFT (MDFT) with plane-wave basis [Phys. Rev. B 106, 125132 (2022)]. Different from traditional KSDFT with the explicit diagonalization method, SDFT and MDFT calculate physical quantities with trace of the corresponding operators. The advantages of SDFT and MDFT compared to the traditional KSDFT are the ability to simulate larger sizes and higher temperatures. In our package, SDFT and MDFT can be used by setting the `esolver_type` parameter to `sdft` for SCF calculations or MD calculations. To start with, you can refer to two examples and an explanation of the *input variables*.

When we have a hamiltonian, the electronic density can be calculated with:

$\rho(\mathbf{r}) = \mathrm{Tr}[f(\hat{H})\mathbf{r}\mathbf{r}]$,

where the Fermi-Dirac function $f(\hat{H}) = \frac{1}{1+\exp(\frac{\hat{H}-\mu}{kT})}$ and it can be calculated with the Chebyshev expansion. Here we only support the "fd" or "fermi-dirac" `smearing_method`, the parameter `smearing_sigma` is equal the temperature T (in Ry) and `nche_sto` represents the order of the expansion.

For physical quantities represented by operator $\hat{O}$, SDFT calculates its trace with:

$\mathrm{Tr}[\hat{O}] = \sum_{i=1}^{N_\chi} \chi_i \hat{O} \chi_i$,

while MDFT calculates the trace as:

$\mathrm{Tr}[\hat{O}] = \sum_{n=1}^{N_\phi} \phi_n \hat{O} \phi_n + \sum_{i=1}^{N_\chi} \tilde{\chi}_i \hat{O} \tilde{\chi}_i$,

where $\{\tilde{\chi}_i\}$ are obtaiend by projecting stochastic orbitals onto the subspace orthogonal to KS orbitals $\{\phi_n\}$:

$\tilde{\chi}_i = \chi_i - \sum_{n=1}^{N_\phi} \phi_n | \chi_i \phi_n$.

Here the number of KS orbitals N_ϕ is controlled by the parameter `nbands` while the number of stochastic orbitals N_χ is controlled by `nbands_sto`.

Besides, although SDFT does not diagonalize the hamiltonian, it can also caluclate DOS and electronic conductivities with parameters `out_dos` and `cal_cond` separately.

5.4 Converging SCF

As in any non-linear systems, numerical instabilities during SCF iterations may lead to nonconvergence. In ABACUS, we offer the following options to facilitate SCF convergence.

5.4.1 Charge Mixing

By mixing the electron density with that obtained from previous steps, numerical instabilities can be ameliorated. ABACUS offers several mixing schemes, and users may make a selection by adjusting the *mixing_type* keyword in INPUT file.

For each of the mixing types, we also provide variables for controlling relevant parameters, including `mixing_beta`, `mixing_ndim`, and `mixing_gg0`.

The default choice is `pulay`, which should work fine in most cases. If convergence issue arises in metallic systems, inclusion of Kerker preconditioning may be helpful, which can be achieved by setting *mixing_gg0* to be a positive number. For the default pulay method, a choice of 1.5 might be a good start.

A large `mixing_beta` means a larger change in electron density for each SCF step. For well-behaved systems, a larger `mixing_beta` leads to faster convergence. However, for some difficult cases, a smaller `mixing_beta` is preferred to avoid numerical instabilities.

An example showcasing different charge mixing methods can be found in our repository. Four INPUT files are provided, with description given in README.

5.4.2 Smearing

Thermal smearing is an efficient tool for accelerating SCF convergence by allowing fractional occupation of molecular orbitals near the band edge. It is important for metallic systems.

In ABACUS, we provide a few smearing methods, which can be controlled using the keyword *smearing_method*. We also provide keyword `smearing_sigma` or `smearing_sigma_temp` to control the energy range of smearing. A larger value of smearing sigma leads to a more diffused occupation curve.

> Note : The two keywords `smearing_sigma` and `smearing_sigma_temp` should not be used concurrently.

We provide an example showing the importance of smearing in our repository. Two INPUT fiels rae provided, with description given in README.

5.5 Accelerating the Calculation

In ABACUS, we provide a few methods for accelerating the calculation. The parameters are usually set as default for calculations where there is not extreme concern for efficiency, as some of them may produce numerical issues under certain circumstances. In short, methods in this section should be used with care. It is better to calibrate the results against the default setting.

5.5.1 K-point Parallelization

In ABACUS, we offer k-point parallelization for calculations with PW basis, which should increase the efficiency when a large k-point mesh is used.

To use k-point parallelization, users may set keyword *kpar* to be larger than 1.

> Note: It has been observed that k-point parallelization cannot work in conjunction with Davidson diagonalization.

5.5.2 K-point Symmetry

Inclusion of k-point symmetry helps increasing the efficiency of calculations by reducing the effective number of k-points used. To turn on k-point symmetry, users may set keyword *symmetry* to be 1.

> Note: In ABACUS we only support point-group symmetry but not space-group symmetry.

5.5.3 Accelerating Grid Integration

For LCAO calculation, the matrix elements of the local potential is evaluated using grid integration. In grid integration, we group real-space FFT grid points into boxes of dimension bx * by * bz, and then proceed with the boxes as the basis unit of calculation.

Setting *bx, by, bz* to be values other than default might help with the efficiency of grid integration.

> Note: the choice of bx, by, bz should be integer factors of the dimension of the real space FFT grid in each direction.

5.5.4 Low Dimension Materials

In grid integration, we chose to parallelize the grid points along the z direction. Therefore, when using LCAO calculation for low dimension materials, it is recommended to put the material more evenly in z direction to avoid imbalanced workload on different MPI threads.

Namely, when calculating 2D materials, it is better to put the material in xz or yz direction; while for 1D materials, it is better to align the material with the z direction.

5.6 SCF in Complex Environments

5.6.1 Implicit Solvation Model

Solid-liquid interfaces are ubiquitous in nature and frequently encountered and employed in materials simulation. The solvation effect should be taken into account in first-principles calculations of such systems so as to obtain accurate predictions.

Implicit solvation model is a well-developed method to deal with solvation effects, which has been widely used in finite and periodic systems. This approach treats the solvent as a continuous medium instead of individual "explicit" solvent molecules, which means that the solute embedded in an implicit solvent, and the average over the solvent degrees of freedom becomes implicit in the properties of the solvent bath. Compared to the "explicit" method, such implicit solvation model can provide qualitatively correct results with much less computational cost, which is particularly suited for large and complex systems. The implicit solvation model implemented in ABACUS follows the methodology developed by Mathew, Sundararaman, Letchworth-Weaver, Arias, and Hennig in 2014.

Input parameters that control the implicit solvation model are listed as follows with detailed explaination and recommended values provided on this webpage:

```
INPUT_PARAMETERS
imp_sol                 1
eb_k                    80
tau                     0.000010798
sigma_k                 0.6
nc_k                    0.00037
```

Example of running DFT calculation with the implicit solvation model is provided in this directory.

5.6.2 External Electric Field

A saw-like potential simulating an electric field can be added to the bare ionic potential, which is a simplified simulation to the field-effect measurements, in which the system is separated from the gate electrode by a dielectric such as silicon oxide.

Whether to apply the external field is controlled via the keyword `efield_flag` in `INPUT` (setting to 1 to turn on the field). Related keywords that control the external field are listed as follows with detailed explaination provided here:

```
INPUT_PARAMETERS
efield_flag        1
efield_dir         2
efield_pos_max     0.5
efield_pos_dec     0.1
efield_amp         0.001
```

Example of running DFT calculation with added external electric field is provided in this directory.

5.6.3 Dipole Correction

A dipole correction can be added to the bare ionic potential, which can compensate for the artificial dipole field within the context of a periodic supercell calculation. The dipole correction implemented in ABACUS follows the methodology proposed by Bengtsson in 1999. This correction must be used ONLY in a slab geometry, for surface calculations, with the discontinuity FALLING IN THE EMPTY SPACE. Note that the common input parameters shared between the external electric field and dipole correction, with detailed explaination provided here. The following keywords settings add dipole correction only without applying any external electric field:

```
INPUT_PARAMETERS
efield_flag        1
dip_cor_flag       1
efield_dir         2
efield_pos_max     0.5
efield_pos_dec     0.1
efield_amp         0
```

While The external electric field and dipole correction can also be added together to the bare ionic potential as follows:

```
INPUT_PARAMETERS
efield_flag        1
dip_cor_flag       1
efield_dir         2
efield_pos_max     0.5
efield_pos_dec     0.1
efield_amp         0.001
```

Examples of running DFT calculations with dipole correction are provided in this directory. There are two input files, where `INPUT1` considers only the dipole correction without no applied external field, while `INPUT2` considers the dipole correction under an applied external field.

To run any of the two cases, users may enter the directory, copy the corresponding input file to `INPUT`, and run ABACUS.

5.6.4 Compensating Charge

Modeling a constant-potential electronchemcial surface reaction requires adjustment of electron numbers in a simulation cell. At the mean time, we need to maintain the supercell's neutrality due to the periodic boundary condition. A distribution of compensating charge thus needs to be implemented in the vacuum region of surface models when extra electrons are added/extracted from the system.

The compensating charge implemented in ABACUS follows the methodology developed by Brumme, Calandra, and Mauri in 2014. Input parameters that control the compensating charge are listed as follows with detailed explaination provided here:

```
INPUT_PARAMETERS
gate_field          1
efield_dir          2
zgate               0.5
block               1
block_down          0.45
block_up            0.55
block_height        0.1
```

Example of running DFT calculation with the compensating charge is provided in this directory.

5.6.5 Van-der-Waals Correction

Conventional DFT functionals often suffer from an inadequate treatment of long-range dispersion, or Van der Waals (VdW) interactions. In order to describe materials where VdW interactions are prominent, one simple and popular approach is to add a Lennard-Jones type term. The resulting VdW-corrected DFT has been proved to be a very effective method for description of both short-range chemical bonding and long-range dispersive interactions.

Currently ABACUS provides three Grimme DFT-D methods, including D2, D3(0) and D3(BJ), to describe Van der Waals interactions. Among them, the D3 method has been implemented in ABACUS based on the dftd3 program written by Stefan Grimme, Stephan Ehrlich and Helge Krieg.

To use VdW-correction, users need to supply value to the `vdw_method` keyword in the `INPUT` file:

- (Default) none: no VdW correction
- d2: DFT-D2 method
- d3_0: DFT-D3(0) method
- d3_bj: DFT-D3(BJ) method

Furthermore, ABACUS also provides a *list of keywords* to control relevant parmeters used in calculating the VdW correction, such as the scale factor (s6) term. Recommended values of such parameters can be found on the webpage. The default values of the parameters in ABACUS are set to be the recommended values for PBE.

Examples of VdW-corrected DFT calculations are provided in this directory. There are two input files, where `INPUT1` shows how to apply D2 correction with user-specified C_6 parameter, and `INPUT2` shows how to apply D3(BJ) correction with default VdW parameters.

To run any of the two cases, users may enter the directory, copy the corresponding input file to `INPUT`, and run ABACUS.

5.7 Spin-polarization and SOC

5.7.1 Non-spin-polarized Calculations

Setting of **"nspin 1"** in INPUT file means calculation with non-polarized spin. In this case, electrons with spin up and spin down have same occupations at every energy states, weights of bands per k point would be double.

5.7.2 Collinear Spin Polarized Calculations

Setting of **"nspin 2"** in INPUT file means calculation with polarized spin along z-axis. In this case, electrons with spin up and spin down will be calculated respectively, number of k points would be doubled. Potential of electron and charge density will separate to spin-up case and spin-down case.

Magnetic moment Settings in *STRU files* are not ignored until **"nspin 2"** is set in INPUT file

When **"nspin 2"** is set, the screen output file will contain magnetic moment information. e.g.

```
ITER    TMAG       AMAG       ETOT(eV)        EDIFF(eV)       DRHO        TIME(s)
GE1     4.16e+00   4.36e+00   -6.440173e+03   0.000000e+00    6.516e-02   1.973e+01
```

where "TMAG" refers to total magnetization and "AMAG" refers to average magnetization. For more detailed orbital magnetic moment information, please use *Mulliken charge analysis*.

Constraint DFT for collinear spin polarized calculations

For some special need, there are two method to constrain electronic spin.

1. **"ocp"** and **"ocp_set"** If **"ocp=1"** and **"ocp_set"** is set in INPUT file, the occupations of states would be fixed by **"ocp_set"**, this method is often used for excited states calculation. Be careful that: when **"nspin=1"**, spin-up and spin-down electrons will both be set, and when **"nspin=2"**, you should set spin-up and spin-down respectively.
2. **"nupdown"** If **"nupdown"** is set to non-zero, number of spin-up and spin-down electrons will be fixed, and Fermi energy level will split to E_Fermi_up and E_Fermi_down. By the way, total magnetization will also be fixed, and will be the value of **"nupdown"**.

5.7.3 Noncollinear Spin Polarized Calculations

The spin non-collinear polarization calculation corresponds to setting **"noncolin 1"**, in which case the coupling between spin up and spin down will be taken into account. In this case, nspin is automatically set to 4, which is usually not required to be specified manually. The weight of each band will not change, but the number of occupied states will be double. If the nbands parameter is set manually, it is generally set to twice what it would be when nspin<4.

In general, non-collinear magnetic moment settings are often used in calculations considering *SOC effects*. When **"lspinorb 1"** in INPUT file, "nspin" is also automatically set to 4. Note: different settings for "noncolin" and "lspinorb" correspond to different calculations:

- noncolin=0 lspinorb=0 nspin<4 : Non-collinear magnetic moments and SOC effects are not considered.
- noncolin=0 lspinorb=0 nspin=4 : Actualy same as the above setting, but the calculation will be larger. So the setting is not recommended.
- noncolin=1 lspinorb=0 : Non-collinear magnetic moments are considered but SOC effects are not considered
- noncolin=0 lspinorb=1 : The SOC effect is considered but the magnetic moment is limited to the Z direction
- noncolin=1 lspinorb=1 : The SOC effect and non-collinear magnetic moment are both calculated.

5.7.4 For the continuation job

- Continuation job for "nspin 1" need file "SPIN1_CHG.cube" which is generated by setting "out_chg=1" in task before. By setting "init_chg file" in new job's INPUT file, charge density will start from file but not atomic.
- Continuation job for "nspin 2" need files "SPIN1_CHG.cube" and "SPIN2_CHG.cube" which are generated by "out_chg 1" with "nspin 2", and refer to spin-up and spin-down charge densities respectively. It should be note that reading "SPIN1_CHG.cube" only for the continuation target magnetic moment job is not supported now.
- Continuation job for "nspin 4" need files "SPIN%s_CHG.cube", where %s in {1,2,3,4}, which are generated by "out_chg 1" with any variable setting leading to 'nspin'=4, and refer to charge densities in Pauli spin matrixes. It should be note that reading charge density files printing by 'nspin'=2 case is supported, which means only σ_{tot} and σ_z are read.

5.8 SOC Effects

5.8.1 SOC

`lspinorb` is used for control whether or not SOC(spin-orbit coupling) effects should be considered.

Both `basis_type=pw` and `basis_type=lcao` support `scf` and `nscf` calculation with SOC effects.

Atomic forces and cell stresses can not be calculated with SOC effects yet.

5.8.2 Pseudopotentials and Numerical Atomic Orbitals

For Norm-Conserving pseudopotentials, there are differences between SOC version and non-SOC version.

Please check your pseudopotential files before calculating. In `PP_HEADER` part, keyword `has_so=1` and `relativistic="full"` refer to SOC effects have been considered, which would lead to different `PP_NONLOCAL` and `PP_PSWFC` parts. Please be careful that `relativistic="full"` version can be used for SOC or non-SOC calculation, but `relativistic="scalar"` version only can be used for non-SOC calculation. When full-relativistic pseudopotential is used for non-SOC calculation, ABACUS will automatically transform it to scalar-relativistic version.

Numerical atomic orbitals in ABACUS are unrelated with spin, and same orbital file can be used for SOC and non-SOC calculation.

5.8.3 Partial-relativistic SOC Effect

Sometimes, for some real materials, both scalar-relativistic and full-relativistic can not describe the exact spin-orbit coupling. Artificial modulation can help for these cases.

`soc_lambda`, which has value range [0.0, 1.0] , is used for modulate SOC effect. In particular, `soc_lambda 0.0` refers to scalar-relativistic case and `soc_lambda 1.0` refers to full-relativistic case.

CHAPTER SIX

BASIS SET AND PSEUDOPOTENTIALS

6.1 Basis Set

ABACUS supports both PW and LCAO basis set, controlled by keyword *basis_type* in INPUT file.

The default value of basis_type is pw. The size of pw basis set is controlled by imposing an upper bound for the *kinetic energy cutoff* of the plane wave.

When choosing lcao basis set, users need to prepare a set of atomic orbitals. Such files may be downloaded from the official website. For more information, also check the `NUMERICAL_ORBITAL` section in the specification of the *STRU file*.

6.2 Generating atomic orbital bases

Users may also choose to generate their own atomic obitals. In ABACUS, the atomic orbital bases are generated using a scheme developed in the paper. A detailed description of the procedure for generating orbitals will be provided later.

6.3 BSSE Correction

For treating BSSE(Basis Set Superposition Error), we allow for the inclusion of "empty" or "ghost" atoms in the calculation. Namely, when expanding the Hamiltonian, basis sets on the atoms are used, while the ionic potentials on those atoms are not included when constructing the Hamiltonian.

An empty atom is defined in the `STRU` file when an element name contains the "empty" suffix, such as "H_empty", "O_empty" and so on. Here we provide an example of calculating the molecular formation energy of H_2O with BSSE correction.

In the example, we provide four STRU files:

- STRU_0 : used along with ntype = 2;normal calculation of water molecule ($E(\mathrm{H_2O})$)

  obtained total energy of -466.4838149140513 eV
- STRU_1 : used along with ntype = 2;calculation of single O atom (E_O)

  obtained total energy of -427.9084406198214 eV
- STRU_2 : used along with ntype = 3;calculation of 1st H atom (E_{H1})

  obtained total energy of -12.59853381731160 eV
- STRU_3 : used along with ntype = 3;calculation of 2nd H atom (E_{H2})

  obtained total energy of -12.59853378720844 eV

Note : Remember to adjust the parameter `ntype` in INPUT file

Thus, the formation energy is given by:

$$\Delta E(\mathrm{H_2O}) = E(\mathrm{H_2O}) - E(\mathrm{O}) - E(\mathrm{H^1}) - E(\mathrm{H^2}) \approx -13.38eV$$

6.4 Pseudopotentials

In ABACUS, we only support norm-conserving pseudopotentials. We support four different formats of the pseudopotential files: UPF, UPF2, VWR, and BLPS. For more information, check the `ATOMIC_SPECIES` section in the specification of the *STRU file*.

Here we list some common sources of the pseudopotential files:

1. Quantum ESPRESSO.
2. SG15-ONCV.
3. DOJO.
4. BLPS.

CHAPTER

SEVEN

GEOMETRY OPTIMIZATION

By setting `calculation` to be `relax` or `cell-relax`, ABACUS supports structural relaxation and variable-cell relaxation.

Current implementation of variable-cell relaxation in ABACUS now follows a nested procedure: fixed cell structural relaxation will be performed, followed by an update of the cell parameters, and the process is repeated until convergence is achieved.

An example of the variable cell relaxation can be found in our repository, which is provided with the reference output file log.ref. Note that in log.ref, each ionic step is labelled in the following manner:

```
-------------------------------------------
RELAX CELL : 3
RELAX IONS : 1 (in total: 15)
-------------------------------------------
```

indicating that this is the first ionic step of the 3rd cell configuration, and it is the 15-th ionic step in total.

7.1 Optimization Algorithms

In the nested procedure mentioned above, we used CG method to perform cell relaxation, while offering four different algorithms for doing fixed-cell structural relaxation: BFGS, SD(steepest descent), CG(conjugate gradient), as well as a mixed CG-BFGS method. The optimziation algorithm can be selected using keyword *relax_method*. We also provide a *list of keywords* for controlling the relaxation process.

7.1.1 BFGS method

The BFGS method is a quasi-Newton method for solving nonlinear optimization problem. It belongs to the class of quasi-Newton method where the Hessian matrix is approximated during the optimization process. If the initial point is not far from the extrema, BFGS tends to work better than gradient-based methods.

In ABACUS, we implemented the BFGS method for doing fixed-cell structural relaxation.

7.1.2 SD method

The SD (steepest descent) method is one of the simplest first-order optimization methods, where in each step the motion is along the direction of the gradient, where the function descents the fastest.

In practice, SD method may take many iterations to converge, and is generally not used.

7.1.3 CG method

The CG (conjugate gradient) method is one of the most widely used methods for solving optimization problems.

In ABACUS, we implemented the CG method for doing fixed-cell structural relaxation as well as the optimization of cell parameters.

7.2 Constrained Optimization

Apart from conventional optimization where all degrees of freedom are allowed to move, we also offer the option of doing constrained optimization in ABACUS.

7.2.1 Fixing Atomic Positions

Users may note that in the above-mentioned example, the atomic positions in STRU file are given along with three integers:

```
Al
0.0
4
0.00 0.00 0.00 1 1 1
0.53 0.50 0.00 1 1 1
0.50 0.00 0.52 1 1 1
0.00 0.50 0.50 1 1 1
```

For relaxation calculations, the three integers denote whether the corresponding degree of freedom is allowed to move. For example, if we replace the STRU file by:

```
Al
0.0
4
0.00 0.00 0.00 1 1 0
0.53 0.50 0.00 1 1 1
0.50 0.00 0.52 1 1 1
0.00 0.50 0.50 1 1 1
```

then the first Al atom will not be allowed to move in z direction.

Fixing atomic position is sometimes helpful during relaxation of isolated molecule/cluster, to prevent the system from drifting in space.

7.2.2 Fixing Cell Parameters

Sometimes we want to do variable-cell relaxation with some of the cell degrees of freedom fixed. This is achieved by keywords such as *fixed_axes*, *fixed_ibrav* and *fixed_atoms*. Specifically, if users are familiar with the `ISIF` option from VASP, then we offer the following correspondence:

- ISIF = 0 : calculation = "relax"
- ISIF = 1, 2 : calculation = "relax", cal_stress = 1
- ISIF = 3 : calculation = "cell-relax"
- ISIF = 4 : calculation = "cell-relax", fixed_axes = "volume"
- ISIF = 5 : calculation = "cell-relax", fixed_axes = "volume", fixed_atoms = True
- ISIF = 6 : calculation = "cell-relax", fixed_atoms = True
- ISIF = 7 : calculation = "cell-realx", fixed_axes = "shape", fixed_atoms = True

CHAPTER

EIGHT

MOLECULAR DYNAMICS

Molecular dynamics (MD) is a computer simulation method for analyzing the physical movements of atoms and molecules. The atoms and molecules are allowed to interact for a fixed period of time, giving a view of the dynamic “evolution” of the system. In the most common version, the trajectories of atoms and molecules are determined by numerically solving Newton’s equations of motion for a system of interacting particles, where forces between the particles and their potential energies are calculated using first-principles calculations (first-principles molecular dynamics, FPMD), or interatomic potentials and molecular mechanics force fields (classical molecular dynamics, CMD).

By setting `calculation` to be `md`, ABACUS currently provides several different MD evolution methods, which is specified by keyword `md_type` in the `INPUT` file:

- -1: FIRE method
- 0: velocity Verlet algorithm (default: NVE ensemble)
- 1: Nose-Hoover style non-Hamiltonian equations of motion
- 2: NVT ensemble with Langevin thermostat
- 4: MSST method

When `md_type` is set to 0, `md_thermostat` is used to specify the thermostat based on the velocity Verlet algorithm.

- nve: NVE ensemble
- anderson: NVT ensemble with Anderson thermostat
- berendsen: NVT ensemble with Berendsen thermostat
- rescaling: NVT ensemble with velocity Rescaling method 1
- rescale_v: NVT ensemble with velocity Rescaling method 2

When `md_type` is set to 1, `md_pmode` is used to specify the NVT or NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.

- none: NVT ensemble
- iso: NPT ensemble with isotropic cetl fluctuations
- aniso: NPT ensemble with anisotropic cetl fluctuations
- tri: NPT ensemble with non-orthogonal (triclinic) simulation box

Furthermore, ABACUS also provides a *list of keywords* to control relevant parmeters used in MD simulations.

To employ CMD calculations, `esolver_type` should be set to be `lj` or `dp`. If DP model is selected, the filename of DP model is specified by keyword `pot_file`.

The MD output information will be written into the file `MD_dump�` in which the atomic forces, atomic velocities, and lattice virial are controlled by keyword `dump_force`, `dump_vel`, and `dump_virial`, respectively.

Examples of MD simulations are also provided. There are eight INPUT files corresponding to eight different MD evolution methods in the directory. For examlpe, `INPUT_0` shows how to employ the NVE simulation.

To run any of the fix cases, users may enter the directory, copy the corresponding input file to `INPUT`, and run ABACUS.

8.1 FIRE

FIRE (fast inertial relaxation engine) is a MD-based minimization algorithm. It is based on conventional molecular dynamics with additional velocity modifications and adaptive time steps. The MD trajectory will descend to an energy-minimum.

8.2 NVE

NVE ensemble (i. e. microcanonical ensemble) is a statistical ensemble that represents the possible states of a mechanical system whose total energy is exactly specified. The system is assumed to be isolated in the sense that it cannot exchange energy or particles with its environment, so that the energy of the system does not change with time.

The primary macroscopic variables of the microcanonical ensemble are the total number of particles in the system (symbol: N), the system's volume (symbol: V), as well as the total energy in the system (symbol: E). Each of these is assumed to be constant in the ensemble.

Currently NVE ensemble in ABACUS is implemented based on the velocity verlet algorithm.

8.3 Nose Hoover Chain

NVT ensemble (i. e. canonical ensemble) is the statistical ensemble that represents the possible states of a mechanical system in thermal equilibrium with a heat bath at a fixed temperature. The system can exchange energy with the heat bath, so that the states of the system will differ in total energy.

The principal thermodynamic variable of the canonical ensemble, determining the probability distribution of states, is the absolute temperature (symbol: T). The ensemble typically also depends on mechanical variables such as the number of particles in the system (symbol: N) and the system's volume (symbol: V), each of which influence the nature of the system's internal states. An ensemble with these three parameters is sometimes called the NVT ensemble.

The isothermal–isobaric ensemble (constant temperature and constant pressure ensemble), also called NPT ensemble, is a statistical mechanical ensemble that maintains the number of particles N, constant temperature T, and constant pressure P. This ensemble plays an important role in chemistry as chemical reactions are usually carried out under constant pressure condition. The NPT ensemble is also useful for measuring the equation of state of model systems whose virial expansion for pressure cannot be evaluated, or systems near first-order phase transitions.

ABACUS perform time integration on Nose-Hoover style non-Hamiltonian equations of motion which are designed to generate positions and velocities sampled from NVT and NPT ensemble.

8.4 Langevin

Langevin thermostat can be used for molecular dynamics equations by assuming that the atoms being simulated are embedded in a sea of much smaller fictional particles. In many instances of solute-solvent systems, the behavior of the solute is desired, and the behavior of the solvent is non-interesting(e.g. proteins, DNA, nanoparticles in solution). In these cases, the solvent influences the dynamics of the solute(typically nanoparticles) via random collisions, and by imposing a frictional drag force on the motion of the nanoparticle in the solvent. The damping factor and the random force combine to give the correct NVT ensemble.

8.5 Anderson

Anderson thermostat couples the system to a heat bath that imposes the desired temperature to simulate the NVT ensemble. The coupling to a heat bath is represented by stochastic collision that act occasionally on randomly selected particles.

8.6 Berendsen

Reset the temperature of a group of atoms by using a Berendsen thermostat, which rescales their velocities every timestep. In this scheme, the system is weakly coupled to a heat bath with some temperature. Though the thermostat does not generate a correct canonical ensemble (especially for small systems), for large systems on the order of hundreds or thousands of atoms/molecules, the approximation yields roughly correct results for most calculated properties.

8.7 Rescaling

Reset the temperature of a group of atoms by explicitly rescaling their velocities. Velocities are rescaled if the current and target temperature differ more than `md_tolerance` (Kelvin).

8.8 Rescale_v

Reset the temperature of a group of atoms by explicitly rescaling their velocities. Every `md_nraise` steps the current temperature is rescaled to target temperature.

8.9 MSST

ABACUS performs the Multi-Scale Shock Technique (MSST) integration to update positions and velocities each timestep to mimic a compressive shock wave passing over the system. The MSST varies the cell volume and temperature in such a way as to restrain the system to the shock Hugoniot and the Rayleigh line. These restraints correspond to the macroscopic conservation laws dictated by a shock front.

CHAPTER

NINE

ACCELERATE PERFORMANCE

This section describes various methods for improving ABACUS performance for different classes of problems running on different kinds of devices.

Accelerated versions of CUDA GPU implementations have been added to ABACUS, which will typically run faster than the standard non-accelerated versions. This requires appropriate hardware to be present on your system, e.g. NVIDIA GPUs.

9.1 CUDA GPU Implementations

In ABACUS, we provide the option to use the GPU devices to accelerate the performance. And it has the following general features:

- **Full gpu implementations**: During the SCF progress, `Psi`, `Hamilt`, `Hsolver`, `DiagCG`, and `DiagoDavid` classes are stored or calculated by the GPU devices.
- **Electronic state data**: (e.g. electronic density) are moved from the GPU to the CPU(s) every scf step.
- **Acclerated by the NVIDIA libraries**: `cuBLAS` for common linear algebra calculations, `cuSolver` for eigen values/vectors, and `cuFFT` for the conversions between the real and recip spaces.
- **Multi GPU supprted**: Using multiple MPI tasks will often give the best performance. Note each MPI task will be bind to a GPU device with automatically computing load balancing.
- **Parallel strategy**: K point parallel.

9.1.1 Required hardware/software

To compile and use ABACUS in CUDA mode, you currently need to have an NVIDIA GPU and install the corresponding NVIDIA CUDA toolkit software on your system (this is only tested on Linux and unsupported on Windows):

- Check if you have an NVIDIA GPU: cat /proc/driver/nvidia/gpus/*/information
- Go to https://developer.nvidia.com/cuda-downloads
- Install a driver and toolkit appropriate for your system (SDK is not necessary)

9.1.2 Building ABACUS with the GPU support:

Check the Advanced Installation Options for the installation of CUDA version support.

9.1.3 Run with the GPU support by editing the INPUT script:

In `INPUT` file we need to set the value keyword *device* to be `gpu`.

9.1.4 Examples

We provides examples of gpu calculations.

9.1.5 Known limitations

- CG and Davidson methods are supported, so the input keyword `ks_solver` can take the values `cg` or `dav`,
- Only PW basis is supported, so the input keyword `basis_type` can only take the value `pw`,
- Only k point parallelization is supported, so the input keyword `kpar` will be set to match the number of MPI tasks automatically.
- Supported CUDA architectures:
  - 60 # P100, 1080ti
  - 70 # V100
  - 75 # T4
  - 80 # A100, 3090

9.1.6 FAQ

```
Q: Does the GPU implementations support atomic orbital basis sets?
A: Currently no.
```

CHAPTER

TEN

ELECTRONIC PROPERTIES AND OUTPUTS

10.1 Extracting Band Structure

ABACUS can calculate the energy band structure, and the examples can be found in examples/band. Similar to the DOS case, we first, do a ground-state energy calculation ***with one additional keyword "out_chg" in the INPUT file***:

```
out_chg              1
```

This will produce the converged charge density, which is contained in the file SPIN1_CHG.cube. Then, use the same `STRU` file, pseudopotential file and atomic orbital file (and the local density matrix file onsite.dm if DFT+U is used) to do a non-self-consistent calculation. In this example, the potential is constructed from the ground-state charge density from the proceeding calculation. Now the INPUT file is like:

```
INPUT_PARAMETERS
#Parameters (General)
ntype 1
nbands 8
calculation nscf
basis_type lcao
read_file_dir   ./

#Parameters (Accuracy)
ecutwfc 60
scf_nmax 50
scf_thr 1.0e-9
pw_diag_thr 1.0e-7

#Parameters (File)
init_chg file
out_band 1
out_proj_band 1

#Parameters (Smearing)
smearing_method gaussian
smearing_sigma 0.02
```

Here the the relevant k-point file KPT looks like,

```
K_POINTS # keyword for start
6 # number of high symmetry lines
Line # line-mode
0.5 0.0 0.5 20 # X
0.0 0.0 0.0 20 # G
```

(continues on next page)

(continued from previous page)

```
0.5 0.5 0.5 20 # L
0.5 0.25 0.75 20 # W
0.375 0.375 0.75 20 # K
0.0 0.0 0.0 1 # G
```

This means we are using:

- 6 number of k points, here means 6 k points: (0.5, 0.0, 0.5) (0.0, 0.0, 0.0) (0.5, 0.5, 0.5) (0.5, 0.25, 0.75) (0.375, 0.375, 0.75) (0.0, 0.0, 0.0)
- 20/1 number of k points along the segment line, which is constructed by two adjacent k points.

Run the program, and you will see a file named BANDS_1.dat in the output directory. Plot it to get energy band structure.

If "out_proj_band" set 1, it will also produce the projected band structure in a file called PBAND_1 in xml format.

The PBAND_1 file starts with number of atomic orbitals in the system, the text contents of element `<band structure>` is the same as data in the BANDS_1.dat file, such as:

```
<pband>
<nspin>1</nspin>
<norbitals>153</norbitals>
<band_structure nkpoints="96" nbands="50" units="eV">
...
```

The rest of the files arranged in sections, each section with a header such as below:

```
<orbital
 index="                                      1"
 atom_index="                                      1"
 species="Si"
 l="                                      0"
 m="                                      0"
 z="                                      1"
>
<data>
...
</data>
```

The shape of text contents of element `<data>` is (Number of k-points, Number of bands)

10.2 Calculating DOS and PDOS

10.2.1 DOS

ABACUS can calculate the density of states (DOS) of the system, and the examples can be found in examples/dos. We first, do a ground-state energy calculation ***with one additional keyword "out_chg" in the INPUT file***:

```
out_chg              1
```

this will produce the converged charge density, which is contained in the file SPIN1_CHG.cube. Then, use the same `STRU` file, pseudopotential file and atomic orbital file (and the local density matrix file onsite.dm if DFT+U is used) to do a non-self-consistent calculation. In this example, the potential is constructed from the ground-state charge density from the proceeding calculation. Now the INPUT file is like:

```
INPUT_PARAMETERS
#Parameters (General)
suffix Si2_diamond
ntype 1
nbands 8
calculation nscf
basis_type lcao
read_file_dir   ./

#Parameters (Accuracy)
ecutwfc 60
symmetry 1
scf_nmax 50
scf_thr 1.0e-9
pw_diag_thr 1.0e-7

#Parameters (File)
init_chg file
out_dos 1
dos_sigma 0.07

#Parameters (Smearing)
smearing_method gaussian
smearing_sigma 0.02
```

Some parameters in the INPUT file are explained:

- calculation

  choose which kind of calculation: scf calculation, nscf calculation, structure relaxation or Molecular Dynamics. Now we need to do one step of nscf calculation. Attention: This is a main variable of ABACUS, and for its more information please see the here.

- pw_diag_thr

  threshold for the CG method which diagonalizes the Hamiltonian to get eigenvalues and eigen wave functions. If one wants to do nscf calculation, pw_diag_thr needs to be changed to a smaller account, typically smaller than 1.0e-3. Note that this parameter only apply to plane-wave calculations that employ the CG or Davidson method to diagonalize the Hamiltonian. For its more information please see the here.

  For LCAO calculations, this parameter will be neglected !

- init_chg

  the type of starting density. When doing scf calculation, this variable can be set ”atomic”. When doing nscf calculation, the charge density already exists(eg. in SPIN1_CHG.cube), and the variable should be set as ”file”. It means the density will be read from the existing file SPIN1_CHG.cube. For its more information please see the here.

- out_dos

  output density of state(DOS). The unit of DOS is `(number of states)/(eV * unitcell)`. For its more information please see the here.

- dos_sigma

  the gaussian smearing parameter(DOS), in unit of eV. For its more information please see the here.

- read_file_dir

  the location of electron density file. For its more information please see the here.

To have an accurate DOS, one needs to have a denser k-point mesh. For example, the KPT file can be set as:

```
K_POINTS
0
Gamma
8 8 8 0 0 0
```

Run the program, and you will see a file named DOS1_smearing.dat in the output directory. The first two columns in the file are the energy and DOS, respectively, and the third column is the sum of DOS. Plot file DOS1_smearing.dat with graphing software, and you'll get the DOS.

```
            -5.49311           0.0518133           0.0518133
            -5.48311           0.0641955            0.116009
            -5.47311           0.0779299            0.193939
            -5.46311           0.0926918             0.28663
            -5.45311            0.108023            0.394653
            -5.44311            0.123346            0.517999
            ...
```

10.2.2 PDOS

Along with the DOS1_smearing.dat file, we also produce the projected density of states (PDOS) in a file called PDOS.

The PDOS file starts with number of atomic orbitals in the system, then a list of energy values, such as:

```
<pdos>
<nspin>1</nspin>
<norbitals>26</norbitals>
<energy_values units="eV">
            -5.50311
            -5.49311
            -5.48311
            -5.47311
...
```

The rest of the fileis arranged in sections, each section with a header such as below:

```
<orbital
 index="                                      1"
 atom_index="                                      1"
 species="Si"
 l="                                      0"
 m="                                      0"
 z="                                      1"
>
<data>
...
</data>
```

which tells the atom and symmetry of the current atomic orbital, and followed by the PDOS values. The values can thus be plotted against the energies. The unit of PDOS is also `(number of states)/(eV * unitcell)`.

10.3 Mulliken Charge Analysis

From version 2.1.0, ABACUS has the function of Mulliken population analysis. The example can be found in examples/mulliken.
To use this function, set 'out_mul' to '1' in the INPUT file. After calculation, there will be an output file named 'mulliken.txt' in the output directory. In the file, there are contents like (`nspin 1`):

```
CALCULATE THE MULLIkEN ANALYSIS FOR EACH ATOM
 Total charge of spin 1:         8
 Total charge:         8
Decomposed Mulliken populations
0                 Zeta of Si                        Spin 1
s                        0                       1.2553358
  sum over m                                     1.2553358
s                        1                    -0.030782972
  sum over m                                  -0.030782972
  sum over m+zeta                                1.2245529
px                        0                       0.85945806
py                        0                       0.85945806
pz                        0                       0.85945806
  sum over m                                     2.5783742
px                        1                     0.0065801228
py                        1                     0.0065801228
pz                        1                     0.0065801228
  sum over m                                   0.019740368
  sum over m+zeta                                2.5981145
d3z^2-r^2                        0                       0.0189287
dxy                        0                       0.046491729
dxz                        0                       0.046491729
dx^2-y^2                        0                       0.0189287
dyz                        0                       0.046491729
  sum over m                                    0.17733259
  sum over m+zeta                               0.17733259
Total Charge on atom:  Si                  4
 ...
```

The file gives Mulliken charge in turn according to the order of atoms in the system. For example, the following block is for the first atom in system (`nspin 2`),

```
0             Zeta of Si                 Spin 1                   Spin 2                    Sum ␣
↪               Diff
...
Total Charge on atom:  Si                  4
Total Magnetism on atom:  Si      -1.2739809e-14
```

And the next block is for the second atom in system, and so on.

```
1             Zeta of Si                 Spin 1                   Spin 2                    Sum ␣
↪               Diff
...
```

For each atom, the file gives detailed Mulliken population analysis at different levels,

- magnetic quantum number level: such as lines beigin with 's,px,py,pz,…'
- azimuthal quantum number level: such as lines begin with 'sum over m'.
- principal quantum number level: such as lines begin with 'sum over m+zeta'. Here 'zeta' equals 'zeta' in the file,

which means how many radial atomic orbitals there are for a given orbital angular momentum.

- atomic level: such as lines begin with 'Total Charge on atom'.

More orbital information can be found in 'Orbital' file output with 'mulliken.txt' when `out_mul 1`

10.4 Extracting Electrostatic Potential

From version 2.1.0, ABACUS has the function of outputing electrostatic potential, which consists of Hartree potential and the local pseudopotential. To use this function, set 'out_pot' to '2' in the INPUT file. Here is an example for the Si-111 surface, and the INPUT file is:

```
INPUT_PARAMETERS
#Parameters (1.General)
calculation scf
ntype 1
nbands 100
gamma_only 0

#Parameters (2.Iteration)
ecutwfc 50
scf_thr 1e-8
scf_nmax 200

#Parameters (3.Basis)
basis_type lcao
ks_solver genelpa

#Parameters (4.Smearing)
smearing_method gaussian
smearing_sigma 0.01

#Parameters (5.Mixing)
mixing_type pulay
mixing_beta 0.4
out_pot 2
```

The STRU file is:

```
ATOMIC_SPECIES
Si 1.000 Si_ONCV_PBE-1.0.upf

NUMERICAL_ORBITAL
Si_gga_8au_60Ry_2s2p1d.orb

LATTICE_CONSTANT
1.8897162

LATTICE_VECTORS
7.6800298691 0.0000000000 0.0000000000
-3.8400149345 6.6511009684 0.0000000000
0.0000000000 0.0000000000 65.6767997742

ATOMIC_POSITIONS
Cartesian
Si
```

(continues on next page)

(continued from previous page)

```
0.0
40
3.840018749 2.217031479 2.351520061 0 0 0
3.840014935 0.000000000 3.135360003 0 0 0
3.840018749 2.217031479 5.486879826 0 0 0
3.840014935 0.000000000 6.270720005 0 0 0
3.840018749 2.217031479 8.622240067 0 0 0
3.840014935 0.000000000 9.406080246 0 0 0
3.840018749 2.217031479 11.757599831 0 0 0
3.840014935 0.000000000 12.541440010 0 0 0
3.840018749 2.217031479 14.892959595 0 0 0
3.840014935 0.000000000 0.000000000 0 0 0
1.920011044 5.542582035 2.351520061 0 0 0
1.920007467 3.325550556 3.135360003 0 0 0
1.920011044 5.542582035 5.486879826 0 0 0
1.920007467 3.325550556 6.270720005 0 0 0
1.920011044 5.542582035 8.622240067 0 0 0
1.920007467 3.325550556 9.406080246 0 0 0
1.920011044 5.542582035 11.757599831 0 0 0
1.920007467 3.325550556 12.541440010 0 0 0
1.920011044 5.542582035 14.892959595 0 0 0
1.920007467 3.325550556 0.000000000 0 0 0
0.000003815 2.217031479 2.351520061 0 0 0
0.000000000 0.000000000 3.135360003 0 0 0
0.000003815 2.217031479 5.486879826 0 0 0
0.000000000 0.000000000 6.270720005 0 0 0
0.000003815 2.217031479 8.622240067 0 0 0
0.000000000 0.000000000 9.406080246 0 0 0
0.000003815 2.217031479 11.757599831 0 0 0
0.000000000 0.000000000 12.541440010 0 0 0
0.000003815 2.217031479 14.892959595 0 0 0
0.000000000 0.000000000 0.000000000 0 0 0
-1.920003772 5.542582035 2.351520061 0 0 0
-1.920007467 3.325550556 3.135360003 0 0 0
-1.920003772 5.542582035 5.486879826 0 0 0
-1.920007467 3.325550556 6.270720005 0 0 0
-1.920003772 5.542582035 8.622240067 0 0 0
-1.920007467 3.325550556 9.406080246 0 0 0
-1.920003772 5.542582035 11.757599831 0 0 0
-1.920007467 3.325550556 12.541440010 0 0 0
-1.920003772 5.542582035 14.892959595 0 0 0
-1.920007467 3.325550556 0.000000000 0 0 0
```

the KPT file is:

```
K_POINTS
0
Gamma
4 4 2 0 0 0
```

Run the program, and you will see the following two files in the output directory,

- ElecStaticPot: contains electrostatic potential (unit: Rydberg) in realspace. This file can be visually viewed by the software of VESTA.
- ElecStaticPot_AVE: contains electrostatic potential (unit: Rydberg) along the z-axis (here z-axis is the default direction of vacuum layer) in realspace.

10.5 Extracting Wave Functions

ABACUS is able to output electron wave functions in both PW and LCAO basis calculations. One can find the examples in examples/wfc.

10.5.1 wave function in G space

For the wave function in G space, one only needs to do a ground-state energy calculation with one additional keyword in the INPUT file: '*out_wfc_pw*' for PW basis calculation, and '*out_wfc_lcao*' for LCAO basis calculation. In the PW basis case, the wave function is output in a file called `WAVEFUNC${k}.txt`, where `${k}` is the index of K point.
In the LCAO basis case, several `LOWF_K_${k}.dat` files will be output in multi-k calculation and `LOWF_GAMMA_S1.dat` in gamma-only calculation.

10.5.2 wave function in real space

One can also choose to output real-space wave function in PW basis calculation with the key word ***out_wfc_r***.

After calculation, an additional directory named `wfc_realspace` will appear in the `OUT.${system}` directory.

Notice: when the ***basis_type*** is `lcao`, only `ienvelope` ***calculation*** is effective. An example is examples/wfc/lcao_ienvelope_Si2.

10.6 Extracting Charge Density

ABACUS can output the charge density by adding the keyword out_chg in INPUT file:

```
out_chg              1
```

After finishing the calculation, the information of the charge density is stroed in files `OUT.${suffix}/SPIN${spin}_CHG.cube`, which can be used to do visualization. The SPIN${spin}_CHG.cube file looks like:

```
Cubefile created from ABACUS SCF calculation
2 (nspin) 0.914047 (fermi energy, in Ry)
2 0.0 0.0 0.0
27 0.222222 0 0
27 0 0.222222 0
27 0 0 0.222222
 26 16 0 0 0
 26 16 3 3 3
 6.63594288898e-01 8.42344790519e-01 1.16349621677e+00 1.18407505276e+00 8.
↪04461725175e-01 3.77164277045e-01
 1.43308127341e-01 5.93894932356e-02 3.23036576611e-02 2.08414809212e-02 1.
↪51271068218e-02 1.27012859512e-02
 1.15620162933e-02 1.08593210023e-02 1.08593210023e-02 1.15620162933e-02 1.
↪27012859512e-02 1.51271068218e-02
 2.08414809212e-02 3.23036576611e-02 5.93894932356e-02 1.43308127341e-01 3.
↪77164277045e-01 8.04461725175e-01
 1.18407505276e+00 1.16349621677e+00 8.42344790519e-01
 8.42344790519e-01 9.86194056340e-01 1.21545550606e+00 1.14987597026e+00 7.
↪50033272229e-01 3.46047149862e-01
 1.32713411550e-01 5.65432381171e-02 3.13971442033e-02 2.04281058891e-02 1.
↪49536046293e-02 1.26489807288e-02
```

(continues on next page)

(continued from previous page)

```
1.15432695307e-02 1.08422207044e-02 1.08422207044e-02 1.15432695307e-02 1.
↪26489807288e-02 1.49536046293e-02
2.04281058891e-02 3.13971442033e-02 5.65432381171e-02 1.32713411550e-01 3.
↪46047149862e-01 7.50033272229e-01
1.14987597026e+00 1.21545550606e+00 9.86194056340e-01
...
```

The first line is a brief description.
The second line contains NSPIN and Fermi energy.
The following 4 lines are the informations of lattice, in order:
  total number of atoms, the coordinate of original point.
  the number of lattice points along lattice vector a1 (nx), a1/nx, in Bohr.
  the number of lattice points along lattice vector a2 (ny), a2/ny, in Bohr.
  the number of lattice points along lattice vector a3 (nz), a3/nz, in Bohr.
The following lines are about the elements and coordinates, in order: the atom number of each atoms, the electron number in pseudopotential, the Cartesian coordinates, in Bohr.
The rest lines are the value of charge density at each grid. Note that the inner loop is z index, followed by y index, x index in turn.
The examples can be found in examples/charge_density

10.7 Extracting Hamiltonian and Overlap Matrices

In ABACUS, we provide the option to write the Hamiltonian and Overlap matrices to files after SCF calculation.

For periodic systems, there are two ways to represent the matrices, the first is to write the entire square matrices for each k point, namely $H(k)$ and $S(k)$; the second is the R space representation, $H(R)$ and $S(R)$, where R is the lattice vector. The two representations are connected by Fourier transform:

- $H(k) = \sum_R H(R)e^{-ikR}$

and

- $S(k) = \sum_R S(R)e^{-ikR}$

10.7.1 out_mat_hs

Users may set the keyword `out_mat_hs` to 1 for outputting the k-space matrices. It is available for both gamma_only and multi-k and calculations. Detailed description of the naming and formats of the output files are given *here*.

10.7.2 out_mat_hs2

The output of R-space matrices is controlled by the keyword `out_mat_hs2`. This functionality is not available for gamma_only calculations. To generate such matrices for gamma only calculations, users should turn off *gamma_only*, and explicitly specify that gamma point is the only k point in the KPT file.

For a more detailed description of the naming and format of the matrices, refer to this *instruction*.

10.7.3 get_S

We also offer the option of only calculating the overlap matrix without running SCF. For that purpose, in `INPUT` file we need to set the value keyword *calculation* to be `get_S`.

A file named `SR.csr` will be generated in the working directory, which contains the overlap matrix.

10.7.4 examples

We provide examples of outputting the matrices. There are four examples:

- out_hs2_multik : writing H® and S® for multi-k calculation
- out_hs_gammaonly : writing H(k) and S(k) for gamma-only calculation
- out_hs_multik : writing H(k) and S(k) for multi-k calculation
- out_s_multik : running get_S for multi-k calculation

Reference output files are provided in each directory.

10.8 Extracting Density Matrices

ABACUS can output the density matrix by adding the keyword "out_dm" in INPUT file:

```
out_dm              1
```

After finishing the calculation, the information of the density matrix is stroed in files `OUT.${suffix}/SPIN${spin}_DM`, which looks like:

```
test
 5.39761
 0.5 0.5 0
 0.5 0 0.5
 0 0.5 0.5
 Si
 2
Direct
 0 0 0
 0.25 0.25 0.25

 1
 0.570336288801065 (fermi energy)
  26 26

 3.904e-01 1.114e-02 2.050e-14 1.655e-13 1.517e-13 -7.492e-15 -1.729e-14 5.915e-15
 -9.099e-15 2.744e-14 3.146e-14 6.631e-15 2.594e-15 3.904e-01 1.114e-02 -7.395e-15
 ...
```

The first 5 lines are the informations of lattice, in order:
  lattice name (if keyword latname is not specified in INPUT, this will be "test"),
  lattice constance with unit in angstrom,
  lattice vector a,
  lattice vector b,
  lattice vector c.
The following lines are about the elements and coordinates, in order: all elements, the atom number of each elements, the

type of coordinate, the coordinates.
After a blank line, the output is the values of NSPIN and fermi energy.
The following line is dimension of the density matrix, and the rest lines are the value of each matrix element.

The examples can be found in examples/density_matrix

- Note: now this function is valid only for LCAO gamma only calcualtion.

10.9 Berry Phase Calculation

From version 2.0.0, ABACUS is capable of calculating macroscopic polarization of insulators by using the Berry phase method, known as the "modern theory of polarization". To calculate the polarization, you need first to do a self-consistent calculation to get the converged charge density. Then, do a non-self-consistent calculation with berry_phase setting to 1. You need also to specify the direction of the polarization you want to calculate. An example is given in the directory examples/berryphase/lcao_PbTiO3.

To run this example, first do a self-consistent calculation:

```
cp INPUT-scf INPUT
cp KPT-scf KPT
mpirun -np 4 abacus
```

Then run a non-self-consistent berry-phase calculation:

```
cp INPUT-nscf-c INPUT
cp KPT-nscf-c KPT
mpirun -np 4 abacus
```

In this example, we calculate the electric polarization along c axis for PbTiO~3~, and below are the INPUT file (nscf) and KPT file (nscf):

```
INPUT_PARAMETERS
pseudo_dir      ../../../tests/PP_ORB  //the path to locate the pesudopotential files
orbital_dir     ../../../tests/PP_ORB  //the path to locate the numerical orbital␣
↪files
ntype          3
ecutwfc        50 // Ry
symmetry       0 // turn off symmetry
calculation    nscf // non-self-consistent calculation
basis_type     lcao // atomic basis
init_chg   file // read charge from files
berry_phase    1 // calculate Berry phase
gdir           3 // calculate polarization along c axis
```

Note: You need to turn off the symmetry when do Berry phase calculations. Currently, ABACUS support Berry phase calculation with nspin=1 and nspin=2. The Berry phase can be calculated in both pw and lcao bases.

- *berry_phase* : 1, calculate berry phase; 0, no calculate berry phase.
- *gdir* : 1, 2, 3, the lattice vector direction of the polarization you want to calculate.

The KPT file need to be modified according to gdir in the INPUT file. Generally, you need denser k points along this direction. For example, in the following KPT file, 4 k-points are taken along the a and b axes, and 8 k-points are taken along the c-axis. You should check the convergence of the k points when calculating the polarization.

```
K_POINTS
0
```

(continues on next page)

(continued from previous page)

```
Gamma
4 4 8 0 0 0
```

The results of the berry phase calculation are written in the “running_nscf.log” in the OUT folder. You may search for these results by searching for keywords “POLARIZATION CALCULATION”.

The results are shown as follows:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                    |
| POLARIZATION CALCULATION:                                          |
|                  Modern Theory of Polarization                     |
| calculate the Macroscopic polarization of a crystalline insulator  |
| by using Berry Phase method.                                       |
|                                                                    |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

VALUES OF POLARIZATION

 The Ionic Phase:   -0.10600
Electronic Phase:    0.92508

The calculated polarization direction is in R3 direction

P =     7.4095194  (mod   18.0922373)  (    0.0000000,    0.0000000,    7.4095194) (e/
↪Omega).bohr

P =     0.0155792  (mod    0.0380407)  (    0.0000000,    0.0000000,    0.0155792) e/
↪bohr^2

P =     0.8906925  (mod    2.1748536)  (    0.0000000,    0.0000000,    0.8906925) C/m^2
```

The electric polarization **P** is multivalued, which modulo a quantum e**R**/V~cell~. Note: the values in parentheses are the components of the **P** along the c axis in the x, y, z Cartesian coordinates when set gdir = 3 in INPUT file.

CHAPTER

ELEVEN

INTERFACES TO OTHER SOFTWARES

11.1 DeePKS

DeePKS is a machine-learning aided density funcitonal model that fits the energy difference between highly accurate but computationally demanding method and effcient but less accurate method via neural-network. As such, the trained DeePKS model can provide highly accurate energetics (and forces) with relatively low computational cost, and can therefore act as a bridge to connect expensive quantum mechanic data and machine-learning-based potentials. While the original framework of DeePKS is for molecular systems, please refer to this reference for the application of DeePKS in periodic systems.

Detailed instructions on installing and running DeePKS can be found on this website. An example for training DeePKS model with ABACUS is also provided. The DeePKS-related keywords in `INPUT` file can be found here.

Note: Use the LCAO basis for DeePKS-related calculations

11.2 DP-GEN

DP-GEN, the deep potential generator, is a package designed to generate deep learning based model of interatomic potential energy and force fields (Yuzhi Zhang, Haidi Wang, Weijie Chen, Jinzhe Zeng, Linfeng Zhang, Han Wang, and Weinan E, DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models, Computer Physics Communications, 2020, 107206). ABACUS can now interface with DP-GEN to generate deep potentials and perform autotests. The minimum recommended version is ABACUS 3.0, dpdata 0.2.8, and dpgen 0.10.7 . In the following part, we take the FCC aluminum as an example.

11.2.1 init_bulk and run

This example can be found in examples/dpgen-example/init_and_run directory.

Firstly, one needs to prepare input files for ABACUS calculation, e.g., "INPUT", "INPUT.md", "KPT", "Al.STRU", "Al_ONCV_PBE-1.0.upf", which are the main input file containing input tags, k-point mesh, crystal structure and pseudoptential, respectively. "INPUT" is for scf calculation, and "INPUT.md" is for AIMD (ab-initio molecular dynamic) calculation.

Secondly, for the "dpgen init_bulk" step, an `init.json` file should be provided:

```
{
    "init_fp_style":     "ABACUS",    # abacus interface
    "stages":            [1,2,3,4],
    "cell_type":         "fcc",
    "super_cell":        [2, 1, 1],
```

(continues on next page)

(continued from previous page)

```
    "elements":          ["Al"],
    "from_poscar":       true,
    "from_poscar_path":  "./Al.STRU",
    "potcars":           ["Al_ONCV_PBE-1.0.upf"],
    "relax_incar":       "INPUT",
    "relax_kpt":         "KPT",
    "md_incar" :         "INPUT.md",
    "md_kpt" :           "KPT",
    "skip_relax":        false,
    "scale":             [1.00],
    "pert_numb":         10,
    "pert_box":          0.01,
    "pert_atom":         0.01,
    "coll_ndata":        10,
    "_comment":          "that's all"
}
```

Next, for the "dpgen run" step, the following `run_param.json` should be provided.

```
{
    "type_map": [
        "Al"
    ],
    "mass_map": [
        26.9815
    ],
    "init_data_prefix": "./",
    "init_data_sys": [
        "Al.STRU.01x01x01/02.md/sys-0004/deepmd"
    ],
    "sys_format": "abacus/stru",   # the initial structures are in ABACUS/STRU formate
    "sys_configs_prefix": "./",
    "sys_configs": [
        [
            "Al.STRU.01x01x01/01.scale_pert/sys-0004/scale*/00000*/STRU"
        ],
        [
            "Al.STRU.01x01x01/01.scale_pert/sys-0004/scale*/000010/STRU"
        ]
    ],
    "_comment": " 00.train ",
    "numb_models": 4,
    "default_training_param": {
        "model": {
            "type_map": [
                "Al"
            ],
            "descriptor": {
                "type": "se_e2_a",
                "sel": [
                    16
                ],
                "rcut_smth": 0.5,
                "rcut": 5.0,
                "neuron": [
                    10,
                    20,
```

(continues on next page)

(continued from previous page)

```
                    40
                ],
                "resnet_dt": true,
                "axis_neuron": 12,
                "seed": 1
            },
            "fitting_net": {
                "neuron": [
                    25,
                    50,
                    100
                ],
                "resnet_dt": false,
                "seed": 1
            }
        },
        "learning_rate": {
            "type": "exp",
            "start_lr": 0.001,
            "decay_steps": 100
        },
        "loss": {
            "start_pref_e": 0.02,
            "limit_pref_e": 2,
            "start_pref_f": 1000,
            "limit_pref_f": 1,
            "start_pref_v": 0.0,
            "limit_pref_v": 0.0
        },
        "training": {
            "stop_batch": 20000,
            "disp_file": "lcurve.out",
            "disp_freq": 1000,
            "numb_test": 4,
            "save_freq": 1000,
            "save_ckpt": "model.ckpt",
            "disp_training": true,
            "time_training": true,
            "profiling": false,
            "profiling_file": "timeline.json",
            "_comment": "that's all"
        }
    },
    "_comment": "01.model_devi ",
    "model_devi_dt": 0.002,
    "model_devi_skip": 0,
    "model_devi_f_trust_lo": 0.05,
    "model_devi_f_trust_hi": 0.15,
    "model_devi_clean_traj": false,
    "model_devi_jobs": [
        {
            "sys_idx": [
                0
            ],
            "temps": [
                50
            ],
```

(continues on next page)

(continued from previous page)

```
                "press": [
                    1.0
                ],
                "trj_freq": 10,
                "nsteps": 300,
                "ensemble": "nvt",
                "_idx": "00"
            },
            {
                "sys_idx": [
                    1
                ],
                "temps": [
                    50
                ],
                "press": [
                    1.0
                ],
                "trj_freq": 10,
                "nsteps": 3000,
                "ensemble": "nvt",
                "_idx": "01"
            }
        ],
        "fp_style": "abacus/scf",
        "shuffle_poscar": false,
        "fp_task_max": 20,
        "fp_task_min": 5,
        "fp_pp_path": "./",
        "fp_pp_files": ["Al_ONCV_PBE-1.0.upf"],   # the pseudopotential file
        "fp_orb_files": ["Al_gga_9au_100Ry_4s4p1d.orb"],  # the orbital file (use only in␣
↪LCAO calculation)
        "k_points":[2, 2, 2, 0, 0, 0],  # k-mesh setting
        "user_fp_params":{  # All the ABACUS input paramters are defined here
        "ntype": 1,         # defining input parameters from INPUT files is not supported␣
↪yet.
        "ecutwfc": 80,
        "mixing_type": "pulay",
        "mixing_beta": 0.8,
        "symmetry": 1,
        "nspin": 1,
        "ks_solver": "cg",
        "smearing_method": "mp",
        "smearing_sigma": 0.002,
        "scf_thr":1e-8,
        "cal_force":1,        # calculate force must be set to 1 in dpgen calculation
        "kspacing": 0.01  # when KSPACING is set, the above k_points setting becomes␣
↪invalid.
        }
}
```

11.2.2 autotest

This example can be found in examples/dpgen-example/autotest directory.

`dpgen autotest` supports to perform `relaxation`,`eos` (equation of state),`elastic`,`surface`,`vacancy`, and `interstitial` calculations with ABACUS. A `property.json` and `machine.json` file need to be provided. For example,

`property.json`:

```
{
    "structures":    ["confs/"],
    "interaction": {
        "type":          "abacus",
        "incar":         "./INPUT",
        "potcar_prefix":"./",
        "potcars":       {"Al": "Al.PD04.PBE.UPF"},
        "orb_files": {"Al":"Al_gga_10au_100Ry_3s3p2d.orb"}
    },
    "_relaxation": {
            "cal_type": "relaxation",
            "cal_setting":{
                    "input_prop": "./INPUT.rlx"
            }
     },
    "properties": [
        {
         "type":          "eos",
         "vol_start":     0.85,
         "vol_end":       1.15,
         "vol_step":      0.01,
         "cal_setting": {
                         "relax_pos": true,
                         "relax_shape": true,
                         "relax_vol": false,
                         "overwrite_interaction":{
                                     "type": "abacus",
                                     "incar": "./INPUT",
                                     "potcar_prefix":"./",
                                     "orb_files": {"Al":"Al_gga_10au_100Ry_3s3p2d.orb
↪"},
                                     "potcars": {"Al": "Al.PD04.PBE.UPF"} }
                        }
        },
         {
         "type":          "elastic",
         "skip":          false,
         "norm_deform":    1e-2,
         "shear_deform":   1e-2
        },
        {
         "type":          "vacancy",
         "skip":          false,
         "supercell":     [2, 2, 2]
        },
        {
         "type":            "surface",
         "skip":          true,
```

(continues on next page)

(continued from previous page)

```
        "min_slab_size":  15,
        "min_vacuum_size":11,
        "pert_xz":        0.01,
        "max_miller":     3,
        "cal_type":       "static"
       }
       ]
}
```

machine.json

```
{
  "api_version": "1.0",
  "deepmd_version": "2.1.0",
  "train" :[
    {
      "command": "dp",
      "machine": {
        "batch_type": "DpCloudServer",
        "context_type": "DpCloudServerContext",
        "local_root" : "./",
        "remote_profile":{
          "email": "xxx@xxx.xxx",
          "password": "xxx",
          "program_id": 000,
            "input_data":{
                "api_version":2,
                "job_type": "indicate",
                "log_file": "00*/train.log",
                "grouped":true,
                "job_name": "Al-train-VASP",
                "disk_size": 100,
                "scass_type":"c8_m32_1 * NVIDIA V100",
                "platform": "ali",
                "image_name":"LBG_DeePMD-kit_2.1.0_v1",
                "on_demand":0
            }
        }
      },
      "resources": {
        "number_node":123473334635,
        "local_root":"./",
        "cpu_per_node": 4,
        "gpu_per_node": 1,
        "queue_name": "GPU",
        "group_size": 1
      }
    }],
  "model_devi":
    [{
      "command": "lmp -i input.lammps -v restart 0",
      "machine": {
        "batch_type": "DpCloudServer",
        "context_type": "DpCloudServerContext",
        "local_root" : "./",
        "remote_profile":{
          "email": "xxx@xxx.xxx",
```

(continues on next page)

(continued from previous page)

```
          "password": "xxx",
          "program_id": 000,
            "input_data":{
              "api_version":2,
              "job_type": "indicate",
              "log_file": "*/model_devi.log",
              "grouped":true,
              "job_name": "Al-devia-ABACUS",
              "disk_size": 200,
              "scass_type":"c8_m32_1 * NVIDIA V100",
              "platform": "ali",
              "image_name":"LBG_DeePMD-kit_2.1.0_v1",
              "on_demand":0
            }
        }
      },
      "resources": {
        "number_node": 28348383,
        "local_root":"./",
        "cpu_per_node": 4,
        "gpu_per_node": 1,
        "queue_name": "GPU",
        "group_size": 100
      }
    }],
  "fp":
    [{
      "command": "OMP_NUM_THREADS=1 mpirun -np 16 abacus",
      "machine": {
        "batch_type": "DpCloudServer",
        "context_type": "DpCloudServerContext",
        "local_root" : "./",
        "remote_profile":{
          "email": "xxx@xxx.xxx",
          "password": "xxx",
         "program_id": 000,
            "input_data":{
              "api_version":2,
              "job_type": "indicate",
              "log_file": "task*/fp.log",
              "grouped":true,
              "job_name": "al-DFT-test",
              "disk_size": 100,
              "scass_type":"c32_m128_cpu",
              "platform": "ali",
              "image_name":"XXXXX",
              "on_demand":0
            }
        }
      },
      "resources": {
        "number_node": 712254638889,
        "cpu_per_node": 32,
        "gpu_per_node": 0,
        "queue_name": "CPU",
        "group_size": 2,
        "local_root":"./",
```

(continues on next page)

(continued from previous page)

```
        "source_list": ["/opt/intel/oneapi/setvars.sh"]
      }
    }
  ]
}
```

For each property, the command `dpgen autotest make property.json` will generate the input files, `dpgen autotest run property.json machine.json` will run the corresponding tasks, and `dpgen autotest post property.json` will collect the final results.

Notes:

- The ABACUS-DPGEN interface can be used in both pw and lcao basis.

11.3 DeepH

DeepH applies meaching learning to predict the Hamiltonian in atomic basis representation. For such purpose, DeepH uses the Hamiltonian and overlap matrices from DFT calculations. Here we introduce how to extract relevant information from ABACUS for the purpose of DeepH training and prediction.

Detailed instructions on installing and running DeepH can be found on its official website. An example for using DeepH with ABACUS is also provided.

Here I intend not to repeat information from the above sources, but to add some minor details related to the setting of ABACUS `INPUT` files.

Note: Use the LCAO basis for DeepH-related calculations

As mentioned in the README.md file in the above-mentioned example, there are two stages where users need to run ABACUS calculations.

The first stage is during the data preparation phase, where we need to run a series of SCF calculations and output the Hamiltonian and overlap matrices. For such purpose, one needs to add the following line in the `INPUT` file:

```
out_mat_hs2 1
```

Files named data-HR-sparse_SPIN`${x}`.csr and data-SR-sparse_SPIN`${x}`.csr will be generated, which contain the Hamiltonian and overlap matrices respectively in csr format. `${x}` takes value of 0 or 1, based on the spin component. More details on this keyword can be found in the *list of input keywords*.

The second stage is during the inference phase. After DeepH training completes, we can apply the model to predict the Hamiltonian on other systems. For that purpose, we also need the overlap matrices from the new systems, but no SCF calculation is required.

For that purpose, in `INPUT` file we need to make the following specification of the keyword `calculation`:

```
calculation get_S
```

A file named `SR.csr` will be generated in the working directory, which contains the overlap matrix.

11.4 Hefei-NAMD

Hefei-NAMD Non-adiabatic molecular dynamics applies surface hopping to incorporate quantum mechanical effects into molecular dynamics simulations. Surface hopping partially incorporates the non-adiabatic effects by including excited adiabatic surfaces in the calculations, and allowing for 'hops' between these surfaces.

Detailed instructions on installing and running Hefei-NAMD can be found on its official website.

ABACUS provides results of molecular dynamics simulations for Hefei-NAMD to do non-adiabatic molecular dynamics simulations.

The steps are as follows :

1. Add output parameters in INPUT when running MD using ABACUS .

```
out_wfc_lcao     1
out_mat_hs       1
```

Then we obtain output files of hamiltonian matrix, overlap matrix, and wavefunction to do NAMD simulation.

2. Clone Hefei-NAMD codes optimized for ABACUS from website.
3. Modify parameters in `Args.py` including directory of ABACUS output files and NAMD parameters. We can see detailed explanation for all parameters in `Args.py`.
4. Run `NAC.py` to prepare related files for NAMD simulations.

```
sbatch sub_nac
```

5. Run `SurfHop.py` to perform NAMD simulations.

```
sbatch sub_sh
```

And results are under directory namddir in `Args.py`.

11.5 Phonopy

Phonopy (Note: please use the `develop` branch, rather than the `master` branch until the abacus interface has been merged into phonopy's `master` branch.) is a powerful package to calculate phonon and related properties. It has provided interface with ABACUS. In the following, we take the FCC aluminum as an example:

1. Prepare a 'setting.conf' with following tags:

```
DIM=2 2 2
ATOM_NAME = Al
```

- when three integers are specified after `DIM =`, a supercell elongated along axes of unit cell is created
- chemical symbols are specified after `ATOM_NAME =`, number of symbols should match `ntype` in ABACUS INPUT file

2. To obtain supercells ($2 \times 2 \times 2$) with displacements, run phonopy:

```
phonopy setting.conf --abacus -d
```

3. Calculate forces on atoms in the supercells with displacements. For each SCF calculation, you should specify `stru_file` with `STRU-{number}` and `cal_force=1` in INPUT in order to calculate force using ABACUS. Be careful not to relax the structures

```
echo 'stru_file ./STRU-001' >> INPUT
```

4. Then create ‘FORCE_SETS’ file using ABACUS inteface:

```
phonopy -f ./disp-{number}/OUT*/running*.log
```

5. Calculate the phonon dispersion:

```
phonopy band.conf --abacus
```

using the following `band.conf` file:

```
ATOM_NAME = Al
DIM = 2 2 2
MESH = 8 8 8
PRIMITIVE_AXES = 0 1/2 1/2  1/2 0 1/2  1/2 1/2 0
BAND= 1 1 1  1/2 1/2 1  3/8 3/8 3/4  0 0 0   1/2 1/2 1/2
BAND_POINTS = 21
BAND_CONNECTION = .TRUE.
```

11.6 Wannier90

Wannier90 is a useful package to generating the maximally-localized Wannier functions (MLWFs), which can be used to compute advanced electronic properties. Some post-processing tools (such as WannierTools, etc.) will use MLWFs for further analysis and calculations.

Currently ABACUS provides an interface to Wannier90 package. The users are assumed to be familiar with the use of Wannier90. The ABACUS-Wannier90 interface is only suitable for nspin=1 or 2, not for nspin=4 or spin-orbit coupling (SOC).

To construct the MLWFs using the wave functions of ABACUS generally requires four steps. Here we use the diamond as an example which can be found in examples/interface_wannier90/.

1. Enter the `ABACUS_towannier90/` folder, prepare a Wannier90 input file `diamond.win`, which is the main input file for Wannier90. Then To generate `diamond.nnkp` file by running Wannier90, which ABACUS will read later:

   ```
   wannier90 -pp diamond.win
   ```

   The content of `diamond.win` is as follows:

   ```
   num_wann        =  4
   num_iter        = 20

   wannier_plot=.true.
   wannier_plot_supercell = 3
   wvfn_formatted = .true.

   begin atoms_frac
   C   -0.12500  -0.1250    -0.125000
   C    0.12500   0.1250     0.125000
   end atoms_frac

   begin projections
   f=0.0,0.0,0.0:s
   ```

(continues on next page)

(continued from previous page)

```
f=0.0,0.0,0.5:s
f=0.0,0.5,0.0:s
f=0.5,0.0,0.0:s
end projections

begin unit_cell_cart
-1.613990   0.000000   1.613990
 0.000000   1.613990   1.613990
-1.613990   1.613990   0.000000
end unit_cell_cart

mp_grid : 4 4 4

begin kpoints
0.0000  0.0000     0.0000
0.0000  0.2500     0.0000
0.0000  0.5000     0.0000
0.0000  0.7500     0.0000
...
end kpoints
```

2. Do a self-consistent calculation and get the converged charge density:

```
cp INPUT-scf INPUT
cp KPT-scf KPT
mpirun -np 4 abacus
```

3. Do a non-self-consistent calculation:

```
cp INPUT-nscf INPUT
cp KPT-nscf KPT
mpirun -np 4 abacus
```

below are the INPUT file (nscf):

```
INPUT_PARAMETERS

ntype                   1
ecutwfc                 50
nbands                  4
calculation             nscf
scf_nmax                50
pw_diag_thr             1.0e-12
scf_thr                 1.0e-15
init_chg                file
symmetry                0
towannier90             1
nnkpfile                diamond.nnkp
```

There are three interface-related parameters in the `INPUT` file:

- *towannier90*: `1`, generate files for wannier90 code; `0`, do not generate.
- *nnkpfile* : the name of the file generated by running "wannier90 -pp …".
- *wannier_spin*: If you use nspin=2, `up`: calculate the Wannier functions for the spin up components ; `down`: calculate the Wannier functions spin down components.

Note: You need to turn off the symmetry during the entire nscf calculation.

To setup the KPT file according to the diamond.win file, which is similar to "begin kpoints ..." in the diamond.win file:

```
K_POINTS
64
Direct
0.0000 0.0000 0.0000 0.0156250
0.0000 0.2500 0.0000 0.0156250
0.0000 0.5000 0.0000 0.0156250
0.0000 0.7500 0.0000 0.0156250
...
```

After the nscf calculation, ABACUS will generate diamond.amn, diamond.mmn, diamond.eig, UNK files in the OUT. folder which are input files needed by Wannier90 code.

4. Copy .amn, .mmn, .eig, UNK file to wannier/ folder, to get the MLWFs by running Wannier90:

```
wannier90 diamond.win
```

Notes:

- The ABACUS-wannier90 interface can be used in both PW and LCAO basis.
- If you want to plot the Wannier function, you must set wvfn_formatted = .true. in diamond.win, otherwise Wannier90 code cannot read files generated by ABACUS because these files are not binary files.

11.7 ASE

11.7.1 Introduction

ASE (Atomic Simulation Environment) provides a set of Python tools for setting, running, and analysing atomic simulations. We have developed an ABACUS calculator (ase-abacus) to be used together with the ASE tools, which exists as an external project with respect to ASE and is maintained by ABACUS developers.

11.7.2 Installation

```
git clone https://gitlab.com/1041176461/ase-abacus.git
cd ase-abacus
python3 setup install
```

11.7.3 Environment variables

ABACUS supports two types of basis sets: PW, LCAO. The path of pseudopotential and numerical orbital files can be set throught the environment variables ABACUS_PP_PATH and ABACUS_ORBITAL_PATH, respectively, e.g.:

```
  PP=${HOME}/pseudopotentials
  ORB=${HOME}/orbitals
  export ABACUS_PP_PATH=${PP}
  export ABACUS_ORBITAL_PATH=${ORB}
```

For PW calculations, only ABACUS_PP_PATH is needed. For LCAO calculations, both ABACUS_PP_PATH and ABACUS_ORBITAL_PATH should be set.

11.7.4 ABACUS Calculator

The default initialization command for the ABACUS calculator is

```
from ase.calculators.abacus import Abacus
```

In order to run a calculation, you have to ensure that at least the following parameters are specified, either in the initialization or as environment variables:

key-word	description
pp	dict of pseudopotentials for involved elememts, such as `pp={'Al':'Al_ONCV_PBE-1.0.upf',...}`.
pseu	directory where the pseudopotential are located, Can also be specified with the `ABACUS_PP_PATH` environment variable. Default: `pseudo_dir=./`.
ba-sis	dict of orbital files for involved elememts, such as `basis={'Al':'Al_gga_10au_100Ry_4s4p1d.orb'}`. It must be set if you want to do LCAO calculations. But for pw calculations, it can be omitted.
ba-sis_	directory where the orbital files are located, Can also be specified with the `ABACUS_ORBITAL_PATH` environment variable. Default: `basis_dir=./`.
xc	which exchange-correlation functional is used. An alternative way to set this parameter is via seting `dft_functional` which is an ABACUS parameter used to specify exchange-correlation functional
kpts	a tuple (or list) of 3 integers `kpts=(int, int, int)`, it is interpreted as the dimensions of a Monkhorst-Pack grid, when `kmode` is `Gamma` or `MP`. It is interpreted as k-points, when `kmode` is `Direct`, `Cartesian` or `Line`, and `knumber` should also be set in these modes to denote the number of k-points. Some other parameters for k-grid settings: including `koffset` and `kspacing`.

For more information on pseudopotentials and numerical orbitals, please visit [ABACUS]. The elaboration of input parameters can be found *here*.

The input parameters can be set like::

```
  calc = Abacus(profile=profile, ntype=1, ecutwfc=50, scf_nmax=50, smearing_method=
↪'gaussian', smearing_sigma=0.01, basis_type='pw', ks_solver='cg', calculation='scf'␣
↪pp=pp, basis=basis, kpts=kpts)
```

The command to run jobs can be set by specifying `AbacusProfile`::

```
  from ase.calculators.abacus import AbacusProfile
  abacus = '/usr/local/bin/abacus'
  profile = AbacusProfile(argv=['mpirun','-n','2',abacus])
```

in which `abacus` sets the absolute path of the `abacus` executable.

11.7.5 MD Analysis

After molecular dynamics calculations, the log file `running_md.log` can be read. If the ‘STRU_MD_*’ files are not continuous (e.g. ‘STRU_MD_0’, ‘STRU_MD_5’, ‘STRU_MD_10’…), the index parameter of read should be as a slice object. For example, when using the command `read('running_md.log', index=slice(0, 15, 5), format='abacus-out')` to parse ‘running_md.log’, ‘STRU_MD_0’, ‘STRU_MD_5’ and ‘STRU_MD_10’ will be read.

11.7.6 SPAP Analysis

SPAP (Structure Prototype Analysis Package) is written by Dr. Chuanxun Su to analyze symmetry and compare similarity of large amount of atomic structures. The coordination characterization function (CCF) is used to measure structural similarity. An unique and advanced clustering method is developed to automatically classify structures into groups.

If you use this program and method in your research, please read and cite the publication:

```
Su C, Lv J, Li Q, Wang H, Zhang L, Wang Y, Ma Y. Construction of crystal structure
prototype database: methods and applications. J Phys Condens Matter. 2017 Apr
26;29(16):165901.
```

and you should install it first with command `pip install spap`.

CHAPTER

TWELVE

DETAILED INTRODUCTION OF THE INPUT FILES

12.1 Full List of INPUT Keywords

- *System variables*

  suffix | *ntype* | *calculation* | *esolver_type* | *symmetry* | *kpar* | *bndpar* | *latname* | *init_wfc* | *init_chg* | *init_vel* | *nelec* | *nupdown* | *dft_functional* | *xc_temperature* | *pseudo_rcut* | *pseudo_mesh* | *mem_saver* | *diago_proc* | *nbspline* | *kspacing* | *min_dist_coef* | *symmetry_prec* | *device*

- *Variables related to input files*

  stru_file | *kpoint_file* | *pseudo_dir* | *orbital_dir* | *read_file_dir* | *wannier_card*

- *Plane wave related variables*

  ecutwfc | *nx,ny,nz* | *pw_seed* | *pw_diag_thr* | *pw_diag_nmax* | *pw_diag_ndim*

- *Numerical atomic orbitals related variables*

  nb2d | *lmaxmax* | *lcao_ecut* | *lcao_dk* | *lcao_dr* | *lcao_rmax* | *search_radius* | *search_pbc* | *bx,by,bz*

- *Electronic structure*

  basis_type | *ks_solver* | *nbands* | *nbands_istate* | *nspin* | *smearing_method* | *smearing_sigma* | *smearing_sigma_temp* | *mixing_type* | *mixing_beta* | *mixing_ndim* | *mixing_gg0* | *mixing_tau* | *mixing_dftu* | *gamma_only* | *printe* | *scf_nmax* | *scf_thr* | *chg_extrap* | *lspinorb* | *noncolin* | *soc_lambda*

- *Electronic structure (SDFT)*

  method_sto | *nbands_sto* | *nche_sto* | *emin_sto* | *emax_sto* | *seed_sto* | *initsto_freq* | *npart_sto*

- *Geometry relaxation*

  relax_nmax | *relax_method* | *relax_cg_thr* | *relax_bfgs_w1* | *relax_bfgs_w2* | *relax_bfgs_rmax* | *relax_bfgs_rmin* | *relax_bfgs_init* | *cal_force* | *force_thr* | *force_thr_ev* | *force_thr_ev2* | *cal_stress* | *stress_thr* | *press1, press2, press3* | *fixed_axes* | *cell_factor* | *fixed_ibrav* | *relax_new* | *relax_scale_force*

- *Variables related to output information*

  out_mul | *out_freq_elec* | *out_freq_ion* | *out_chg* | *out_pot* | *out_dm* | *out_wfc_pw* | *out_wfc_r* | *out_wfc_lcao* | *out_dos* | *out_band* | *out_proj_band* | *out_stru* | *out_bandgap* | *out_level* | *out_alllog* | *out_mat_hs* | *out_mat_r* | *out_mat_hs2* | *out_mat_t* | *out_mat_dh* | *out_app_flag* | *out_hs2_interval* | *out_element_info* | *restart_save* | *restart_load* | *dft_plus_dmft* | *rpa*

- *Density of states*

  dos_edelta_ev | *dos_sigma* | *dos_scale* | *dos_emin_ev* | *dos_emax_ev* | *dos_nche*

- *Exact exchange* (Under tests)

  exx_hybrid_alpha | *exx_hse_omega* | *exx_separate_loop* | *exx_hybrid_step* | *exx_lambda* | *exx_pca_threshold* | *exx_c_threshold* | *exx_v_threshold* | *exx_c_grad_threshold* | *exx_v_grad_threshold* | *exx_dm_threshold* | *exx_schwarz_threshold* | *exx_cauchy_threshold* | *exx_cauchy_grad_threshold* | *exx_ccp_threshold* | *exx_ccp_rmesh_times* | *exx_distribute_type* | *exx_opt_orb_lmax* | *exx_opt_orb_ecut* | *exx_opt_orb_tolerence* | *exx_real_number*

- *Molecular dynamics*

  md_type | *md_thermostat* | *md_nstep* | *md_restart* | *md_dt* | *md_tfirst, md_tlast* | *md_dumpfreq* | *md_restartfreq* | *md_seed* | *md_prec_level* | *ref_cell_factor* | *md_tfreq* | *md_tchain* | *md_pmode* | *md_pcouple* | *md_pfirst, md_plast* | *md_pfreq* | *md_pchain* | *dump_force* | *dump_vel* | *dump_virial* | *lj_rcut* | *lj_epsilon* | *lj_sigma* | *pot_file* | *msst_direction* | *msst_vel* | *msst_vis* | *msst_tscale* | *msst_qmass* | *md_damp* | *md_tolerance* | *md_nraise*

- *vdW correction*

  vdw_method | *vdw_s6* | *vdw_s8* | *vdw_a1* | *vdw_a2* | *vdw_d* | *vdw_abc* | *vdw_C6_file* | *vdw_C6_unit* | *vdw_R0_file* | *vdw_R0_unit* | *vdw_cutoff_type* | *vdw_cutoff_radius* | *vdw_radius_unit* | *vdw_cutoff_period* | *vdw_cn_thr* | *vdw_cn_thr_unit*

- *Berry phase and wannier90 interface*

  berry_phase | *gdir* | *towannier90* | *nnkpfile* | *wannier_spin*

- *TDDFT: time dependent density functional theory* (Under tests)

  td_force_dt | *td_vext* | *td_vext_dire* | *td_stype* | *td_ttype* | *td_tstart* | *td_tend* | *td_lcut1* | *td_lcut2* | *td_gauss_freq* | *td_guass_phase* | *td_gauss_sigma* | *td_gauss_t0*| *td_gauss_amp* | *td_trape_freq* | *td_trape_phase* | *td_trape_t1* | *td_trape_t2* | *td_trape_t3* | *td_trape_amp* | *td_trigo_freq1* | *td_trigo_freq2* | *td_trigo_phase1* | *td_trigo_phase2* | *td_trigo_amp* | *td_heavi_t0* | *td_heavi_amp* | *out_dipole* |*out_efield*| *ocp* | *ocp_set* | *td_val_elec_01* | *td_val_elec_02* |*td_val_elec_03*

- *DFT+U correction* (Under development)

  dft_plus_u | *orbital_corr* | *hubbard_u* | *yukawa_potential* | *yukawa_lambda* | *omc*

- *Variables useful for debugging*

  nurse | *t_in_h* | *vl_in_h* | *vnl_in_h* | *vh_in_h* | *vion_in_h* | *test_force* | *test_stress* | *colour* | *test_skip_ewald*

- *NAOs*

  bessel_nao_ecut | *bessel_nao_tolerence* | *bessel_nao_rcut* | *bessel_nao_smooth* | *bessel_nao_sigma*

- *DeePKS*

  deepks_out_labels | *deepks_scf* | *deepks_model* | *bessel_descriptor_lmax* | *bessel_descriptor_ecut* | *bessel_descriptor_tolerence* | *bessel_descriptor_rcut* | *bessel_descriptor_smooth* | *bessel_descriptor_sigma* | *deepks_bandgap* | *deepks_out_unittest*

- *OFDFT: orbital free density functional theory*

  of_kinetic | *of_method* | *of_conv* | *of_tole* | *of_tolp* | *of_tf_weight* | *of_vw_weight* | *of_wt_alpha* | *of_wt_beta* | *of_wt_rho0* | *of_hold_rho0* | *of_read_kernel* | *of_kernel_file* | *of_full_pw* | *of_full_pw_dim*

- *Electric field and dipole correction*

  efield_flag | *dip_cor_flag* | *efield_dir* | *efield_pos_max* | *efield_pos_dec* | *efield_amp*

- *Gate field (compensating charge)*

  gate_flag | *zgate* | *block* | *block_down* | *block_up* | *block_height*

- *Electronic conductivities*

  cal_cond | *cond_nche* | *cond_dw* | *cond_wcut* | *cond_dt* | *cond_dtbatch* | *cond_fwhm* | *cond_nonlocal*
- *Implicit solvation model*

  imp_sol | *eb_k* | *tau* | *sigma_k* | *nc_k*

back to top

12.1.1 System variables

These variables are used to control general system parameters.

suffix

- **Type**: String
- **Description**: In each run, ABACUS will generate a subdirectory in the working directory. This subdirectory contains all the information of the run. The subdirectory name has the format: OUT.suffix, where the `suffix` is the name you can pick up for your convenience.
- **Default**: ABACUS

ntype

- **Type**: Integer
- **Description**: Number of different atom species in this calculation. If this value is not equal to the atom species in the STRU file, ABACUS will stop and quit. If not set or set to 0, ABACUS will automatically set it to the atom species in the STRU file.
- **Default**: 0

calculation

- **Type**: String
- **Description**: Specify the type of calculation.
  - *scf*: do self-consistent electronic structure calculation
  - *relax*: do structure relaxation calculation, one can use `relax_nmax` to decide how many ionic relaxations you want.
  - *cell-relax*: do variable-cell relaxation calculation.
  - *nscf*: do the non self-consistent electronic structure calculations. For this option, you need a charge density file. For nscf calculations with planewave basis set, pw_diag_thr should be <= 1e-3.
  - *istate*: For LCAO basis. Please see the explanation for variable `nbands_istate`.
  - *ienvelope*: Envelope function for LCAO basis. Please see the explanation for variable `nbands_istate`.
  - *md*: molecular dynamics
  - *test_memory* : checks memory required for the calculation. The number is not quite reliable, please use it with care
  - *test_neighbour* : only performs neighbouring atom search

  - *gen_bessel* : generates projectors (a series of Bessel functions) for DeePKS; see also keywords bessel_descriptor_lmax, bessel_descriptor_rcut and bessel_descriptor_tolerence. A file named `jle.orb` will be generated which contains the projectors. An example is provided in examples/H2O-deepks-pw.
  - *get_S* : only works for multi-k calculation with LCAO basis. Generates and writes the overlap matrix to a file named `SR.csr` in the working directory. The format of the file will be the same as that generated by *out_mat_hs2*.
- **Default**: scf

esolver_type

- **Type**: String
- **Description**: choose the energy solver.
  - ksdft: Kohn-Sham density functional theory;
  - ofdft: orbital-free density functional theory;
  - sdft: *stochastic density functional theory*;
  - tddft: real-time time-dependent density functional theory (TDDFT);
  - lj: Leonard Jones potential;
  - dp: DeeP potential;
- **Default**: ksdft

symmetry

- **Type**: Integer
- **Description**: takes value 1, 0 or -1.
  - if set to 1, symmetry analysis will be performed to determine the type of Bravais lattice and associated symmetry operations. (point groups only)
  - if set to 0, only time reversal symmetry would be considered in symmetry operations, which implied k point and -k point would be treated as a single k point with twice the weight.
  - if set to -1, no symmetry will be considered.
- **Default**: 0

kpar

- **Type**: Integer
- **Description**: divide all processors into kpar groups, and k points will be distributed among each group. The value taken should be less than or equal to the number of k points as well as the number of MPI threads.
- **Default**: 1

bndpar

- **Type**: Integer
- **Description**: divide all processors into bndpar groups, and bands (only stochastic orbitals now) will be distributed among each group. It should be larger than 0.
- **Default**: 1

latname

- **Type**: String
- **Description**: Specifies the type of Bravias lattice. When set to `none`, the three lattice vectors are supplied explicitly in STRU file. When set to a certain Bravais lattice type, there is no need to provide lattice vector, but a few lattice parameters might be required. For more information regarding this parameter, consult the *page on STRU file*. Available options are (correspondence with ibrav in QE is given in parenthesis):
  - `none`: free structure.
  - `sc`: simple cubic. (1)
  - `fcc`: face-centered cubic. (2)
  - `bcc`: body-centered cubic. (3)
  - `hexagonal`: hexagonal. (4)
  - `trigonal`: trigonal. (5)
  - `st`: simple tetragonal. (6)
  - `bct`: body-centered tetragonal. (7)
  - `so`: orthorhombic. (8)
  - `baco`: base-centered orthorhombic. (9)
  - `fco`: face-centered orthorhombic. (10)
  - `bco`: body-centered orthorhombic. (11)
  - `sm`: simple monoclinic. (12)
  - `bacm`: base-centered monoclinic. (13)
  - `triclinic`: triclinic. (14)
- **Default**: `none`

init_wfc

- **Type**: String
- **Description**: Only useful for plane wave basis only now. It is the name of the starting wave functions. In the future. we should also make this variable available for localized orbitals set. Available options are:
  - `atomic`: from atomic pseudo wave functions. If they are not enough, other wave functions are initialized with random numbers.
  - `atomic+random`: add small random numbers on atomic pseudo-wavefunctions
  - `file`: from file

  - `random`: random numbers
- **Default**:`atomic`

init_chg

- **Type**: String
- **Description**: This variable is used for both plane wave set and localized orbitals set. It indicates the type of starting density. If set to `atomic`, the density is starting from the summation of the atomic density of single atoms. If set this to `file`, the density will be read in from a file. Besides, when you do `nspin=1` calculation, you only need the density file SPIN1_CHG.cube. However, if you do `nspin=2` calculation, you also need the density file SPIN2_CHG.cube. The density file should be output with these names if you set out_chg = 1 in INPUT file.
- **Default**: atomic

init_vel

- **Type**: Boolean
- **Description**: Read the atom velocity from the atom file (STRU) if set to true.
- **Default**: false

nelec

- **Type**: Real
- **Description**: If >0.0, this denotes the total number of electrons in the system. Must be less than 2*nbands. If set to 0.0, the total number of electrons will be calculated by the sum of valence electrons (i.e. assuming neutral system).
- **Default**: 0.0

nupdown

- **Type**: Real
- **Description**: If >0.0, this denotes the difference number of electrons between spin-up and spin-down in the system. The range of value must in [-nelec ~ nelec]. It is one method of constraint DFT, the fermi energy level will separate to E_Fermi_up and E_Fermi_down. If set to 0.0, no constrain apply to system.
- **Default**: 0.0

dft_functional

- **Type**: String
- **Description**: In our package, the XC functional can either be set explicitly using the `dft_functional` keyword in `INPUT` file. If `dft_functional` is not specified, ABACUS will use the xc functional indicated in the pseudopotential file. On the other hand, if dft_functional is specified, it will overwrite the functional from pseudopotentials and performs calculation with whichever functional the user prefers. We further offer two ways of supplying exchange-correlation functional. The first is using 'short-hand' names such as 'LDA', 'PBE', 'SCAN'. A complete list of 'short-hand' expressions can be found in the source code. The

other way is only available when ***compiling with LIBXC***, and it allows for supplying exchange-correlation functionals as combinations of LIBXC keywords for functional components, joined by a plus sign, for example, 'dft_functional='LDA_X_1D_EXPONENTIAL+LDA_C_1D_CSC'. The list of LIBXC keywords can be found on its website. In this way, **we support all the LDA,GGA and mGGA functionals provided by LIBXC**.

Furthermore, the old INPUT parameter exx_hybrid_type for hybrid functionals has been absorbed into dft_functional. Options are `hf` (pure Hartree-Fock), `pbe0`(PBE0), `hse` (Note: in order to use HSE functional, LIBXC is required). Note also that HSE has been tested while PBE0 has NOT been fully tested yet, and the maximum CPU cores for running exx in parallel is $N(N+1)/2$, with N being the number of atoms. And forces for hybrid functionals are not supported yet.

If set to `opt_orb`, the program will not perform hybrid functional calculation. Instead, it is going to generate opt-ABFs as discussed in this article.

- **Default**: same as UPF file.

xc_temperature

- **Type**: Real
- **Description**: specifies temperature when using temperature-dependent XC functionals (KSDT and so on); unit in Rydberg
- **Default** : 0.0

pseudo_rcut

- **Type**: Real
- **Description**: Cut-off of radial integration for pseudopotentials, in Bohr.
- **Default**: 15

pseudo_mesh

- **Type**: Integer
- **Description**: If set to 0, then use our own mesh for radial integration of pseudopotentials; if set to 1, then use the mesh that is consistent with quantum espresso.
- **Default**: 0

mem_saver

- **Type**: Boolean
- **Description**: Used only for nscf calculations. If set to 1, then a memory saving technique will be used for many k point calculations.
- **Default**: 0

diago_proc

- **Type**: Integer
- **Description**: If set to a positive number, then it specifies the number of threads used for carrying out diagonalization. Must be less than or equal to total number of MPI threads. Also, when cg diagonalization is used, diago_proc must be the same as the total number of MPI threads. If set to 0, then it will be set to the number of MPI threads. Normally, it is fine just leave it to the default value. Only used for pw base.
- **Default**: 0

nbspline

- **Type**: Integer
- **Description**: If set to a natural number, a Cardinal B-spline interpolation will be used to calculate Structure Factor. `nbspline` represents the order of B-spline basis and a larger one can get more accurate results but cost more. It is turned off by default.
- **Default**: -1

kspacing

- **Type**: Real
- **Description**: Set the smallest allowed spacing between k points, unit in 1/bohr. It should be larger than 0.0, and suggest smaller than 0.25. When you have set this value > 0.0, then the KPT file is unnecessary, and the number of K points nk_i = max(1, int(|b_i|/KSPACING)+1), where b_i is the reciprocal lattice vector. The default value 0.0 means that ABACUS will read the applied KPT file. Notice: if gamma_only is set to be true, kspacing is invalid.
- **Default**: 0.0

min_dist_coef

- **Type**: Real
- **Description**: a factor related to the allowed minimum distance between two atoms. At the beginning, ABACUS will check the structure, and if the distance of two atoms is shorter than min_dist_coef*(standard covalent bond length), we think this structure is unreasonable. If you want to calculate some structures in extreme conditions like high pressure, you should set this parameter as a smaller value or even 0.
- **Default**: 0.2

symmetry_prec

- **Type**: Real
- **Description**: The accuracy for symmetry judgment. The unit is Bohr.
- **Default**: 1.0e-5

device

- **Type**: String
- **Description**: Specifies the computing device for ABACUS.

  Available options are:

  – `cpu`: for CPUs via Intel, AMD, or Other supported CPU devices

  – `gpu`: for GPUs via CUDA.

  Known limitations:

  – `pw basis`: required by the `gpu` acceleration options

  – `cg ks_solver`: required by the `gpu` acceleration options
- **Default**: `cpu`

back to top

12.1.2 Variables related to input files

These variables are used to control parameters related to input files.

stru_file

- **Type**: String
- **Description**: This parameter specifies the name of structure file which contains various information about atom species, including pseudopotential files, local orbitals files, cell information, atom positions, and whether atoms should be allowed to move.
- **Default**: STRU

kpoint_file

- **Type**: String
- **Description**: This parameter specifies the name of k-points file. Note that if you use atomic orbitals as basis, and you only use gamma point, you don't need to have k-point file in your directory, ABACUS will automatically generate `KPT` file. Otherwise, if you use more than one k-point, please do remember the algorithm in ABACUS is different for gamma only and various k-point dependent simulations. So first you should turn off the k-point algorithm by set `gamma_only = 0` in `INPUT` and then you should setup your own k-points file.
- **Default**: KPT

pseudo_dir

- **Type**: String
- **Description**: This parameter specifies pseudopotential directory.
- **Default**: ./

orbital_dir

- **Type**: String
- **Description**: This parameter specifies orbital file directory.
- **Default**: ./

read_file_dir

- **Type**: String
- **Description**: when the program needs to read files such as electron density(`SPIN1_CHG.cube`) as a starting point, this variable tells the location of the files. For example, './' means the file is located in the working directory.
- **Default**: OUT.$suffix

wannier_card

- **Type**: String
- **Description**: Relevant when using ABACUS with wannier90. Tells the name of the input file related to wannier90.
- **Default**: "none"

back to top

12.1.3 Plane wave related variables

These variables are used to control the plane wave related parameters.

ecutwfc

- **Type**: Real
- **Description**: Energy cutoff for plane wave functions, the unit is **Rydberg**. Note that even for localized orbitals basis, you still need to setup an energy cutoff for this system. Because our local pseudopotential parts and the related force are calculated from plane wave basis set, etc. Also, because our orbitals are generated by matching localized orbitals to a chosen set of wave functions from a certain energy cutoff, this set of localize orbitals is most accurate under this same plane wave energy cutoff.
- **Default**: 50

nx, ny, nz

- **Type**: Integer
- **Description**: If set to a positive number, then the three variables specify the numbers of FFT grid points in x, y, z directions, respectively. If set to 0, the number will be calculated from ecutwfc.
- **Default**: 0

pw_seed

- **Type**: Integer
- **Description**: Only useful for plane wave basis only now. It is the random seed to initialize wave functions. Only positive integers are available.
- **Default**:0

pw_diag_thr

- **Type**: Real
- **Description**: Only used when you use `diago_type = cg` or `diago_type = david`. It indicates the threshold for the first electronic iteration, from the second iteration the pw_diag_thr will be updated automatically. **For nscf calculations with planewave basis set, pw_diag_thr should be <= 1e-3.**
- **Default**: 0.01

pw_diag_nmax

- **Type**: Integer
- **Description**: Only useful when you use `ks_solver = cg` or `ks_solver = dav`. It indicates the maximal iteration number for cg/david method.
- **Default**: 40

pw_diag_ndim

- **Type**: Integer
- **Description**: Only useful when you use `ks_solver = dav`. It indicates the maximal dimension for the Davidson method.
- **Default**: 4

back to top

12.1.4 Numerical atomic orbitals related variables

These variables are used to control the numerical atomic orbitals related parameters.

nb2d

- **Type**: Integer
- **Description**: In LCAO calculations, we arrange the total number of processors in an 2D array, so that we can partition the wavefunction matrix (number of bands*total size of atomic orbital basis) and distribute them in this 2D array. When the system is large, we group processors into sizes of nb2d, so that multiple processors take care of one row block (a group of atomic orbitals) in the wavefunction matrix. If set to 0, nb2d will be automatically set in the program according to the size of atomic orbital basis:
  - if size <= 500 : nb2d = 1
  - if 500 < size <= 1000 : nb2d = 32
  - if size > 1000 : nb2d = 64;
- **Default**: 0

lmaxmax

- **Type**: Integer
- **Description**: If not equals to 2, then the maximum l channels on LCAO is set to lmaxmax. If 2, then the number of l channels will be read from the LCAO data sets. Normally no input should be supplied for this variable so that it is kept as its default.
- **Default**: 2.

lcao_ecut

- **Type**: Real
- **Description**: Energy cutoff when calculating LCAO two-center integrals. In Ry.
- **Default**: 50

lcao_dk

- **Type**: Real
- **Description**: Delta k for 1D integration in LCAO
- **Default**: 0.01

lcao_dr

- **Type**: Real
- **Description**: Delta r for 1D integration in LCAO
- **Default**: 0.01

lcao_rmax

- **Type**: Real
- **Description**: Max R for 1D two-center integration table
- **Default**: 30

search_radius

- **Type**: Real
- **Description**: Set the search radius for finding neighbouring atoms. If set to -1, then the radius will be set to maximum of projector and orbital cut-off.
- **Default**: -1

search_pbc

- **Type**: Boolean
- **Description**: In searching for neighbouring atoms, if set to 1, then periodic images will also be searched. If set to 0, then periodic images will not be searched.
- **Default**: 1

bx, by, bz

- **Type**: Integer
- **Description**: In the matrix operation of grid integral, bx/by/bz grids (in x, y, z directions) are treated as a whole as a matrix element. A different value will affect the calculation speed.
- **Default**: 2

back to top

12.1.5 Electronic structure

These variables are used to control the electronic structure and geometry relaxation calculations.

basis_type

- **Type**: String
- **Description**: This is an important parameter to choose basis set in ABACUS.
  - *pw*: Using plane-wave basis set only.
  - *lcao_in_pw*: Expand the localized atomic set in plane-wave basis.
  - lcao: Using localized atomic orbital sets.
- **Default**: pw

ks_solver

- **Type**: String
- **Description**: It's about the choice of diagonalization methods for the Hamiltonian matrix expanded in a certain basis set.

  For plane-wave basis,

  - cg: cg method.
  - dav: the Davidson algorithm. (Currently not working with Intel MKL library).

  For atomic orbitals basis,

  - genelpa: This method should be used if you choose localized orbitals.
  - scalapack-gvx: scalapack can also be used for localized orbitals.
  - cusolver: this method needs building with the cusolver component for lcao and at least one gpu is available.

  If you set ks_solver=`genelpa` for basis_type=`pw`, the program will be stopped with an error message:

  ```
  genelpa can not be used with plane wave basis.
  ```

  Then the user has to correct the input file and restart the calculation.
- **Default**: `cg` (pw) or `genelpa` (lcao)

nbands

- **Type**: Integer
- **Description**: Number of Kohn-Sham orbitals to calculate. It is recommended you setup this value, especially when you use smearing techniques, more bands should be included.
- **Default**:
  - nspin=1: max(1.2*occupied_bands, occupied_bands + 10)
  - nspin=2: max(1.2*nelec_spin, nelec_spin + 10), in which nelec_spin = max(nelec_spin_up, nelec_spin_down)
  - nspin=4: max(1.2*nelec, nelec + 20)

nbands_istate

- **Type**: Integer
- **Description**: Only used when `calculation = ienvelope` or `calculation = istate`, this variable indicates how many bands around the Fermi level you would like to calculate. `ienvelope` means to calculate the envelope functions of wave functions $\Psi_i = \Sigma_\mu C_{i\mu}\Phi_\mu$, where Ψ_i is the ith wave function with the band index i and Φ_μ is the localized atomic orbital set. `istate` means to calculate the density of each wave function $|\Psi_i|^2$. Specifically, suppose we have highest occupied bands at 100th wave functions. And if you set this variable to 5, it will print five wave functions from 96th to 105th. But before all this can be carried out, the wave functions coefficients should be first calculated and written into a file by setting the flag `out_wfc_lcao = 1`.
- **Default**: 5

nspin

- **Type**: Integer
- **Description**: Number of spin components of wave functions. There are only two choices now: 1 or 2, meaning non spin or collinear spin. For the case of *noncollinear polarized*, nspin will be automatically set to 4 without being specified in user input.
- **Default**: 1

smearing_method

- **Type**: String
- **Description**: It indicates which occupation and smearing method is used in the calculation.
  - fixed: use fixed occupations.
  - gauss or gaussian: use Gaussian smearing method.
  - mp: use methfessel-paxton smearing method; recommended for metals.
  - fd: Fermi-Dirac smearing method: $f = 1/\{1 + \exp[(E - \mu)/kT]\}$ and smearing_sigma below is the temperature T (in Ry).
- **Default**: fixed

smearing_sigma

- **Type**: Real
- **Description**: energy range for smearing, the unit is Rydberg.
- **Default**: 0.001

smearing_sigma_temp

- **Type**: Real
- **Description**: energy range for smearing, and is the same as smearing_sigma, but the unit is K. smearing_sigma = 1/2 * kB * smearing_sigma_temp.

mixing_type

- **Type**: String
- **Description**: Charge mixing methods. We offer the following 3 options:
  - plain: Just simple mixing.
  - pulay: Standard Pulay method.
  - broyden: Broyden method.
- **Default**: pulay

mixing_beta

- **Type**: Real
- **Description**: mixing parameter. We recommend the following options:
  - default: -10.0 means program will auto set **mixing_beta** and **mixing_gg0** before charge mixing method starts.
    - Default values of transition metal system are **mixing_beta=0.2** and **mixing_gg0=1.5**;
    - Default values of metal system (bandgap <= 1.0 eV) are **mixing_beta=0.2** and **mixing_gg0=0.0**;
    - Default values of other systems (bandgap > 1.0eV) are **mixing_beta=0.7** and **mixing_gg0=0.0**.
  - 0: keep charge density unchanged, usually used for restarting with **init_chg=file** or testing.
  - 0.1 or less: if convergence of SCF calculation is difficult to reach, please try **0 < mixing_beta < 0.1**
  - For low-dimensional large systems, the setup of **mixing_beta=0.1**, **mixing_ndim=20**, and **mixing_gg0=1.5** usually works well.
- **Default**: -10.0

mixing_ndim

- **Type**: Integer
- **Description**: It indicates the mixing dimensions in Pulay, Pulay method uses the density from previous mixing_ndim steps and do a charge mixing based on this density.
- **Default**: 8

mixing_gg0

- **Type**: Real
- **Description**: When set to a positive number, the high frequency wave vectors will be suppressed by multiplying a scaling factor $\frac{k^2}{k^2+gg0^2}$; if set to 0, then no Kerker scaling is performed. Setting mixing_gg0 = 1.5 is normally a good starting point.
- **Default**: 0.0

mixing_tau

- **Type**: Boolean
- **Description**: Only relevant for meta-GGA calculations. If set to true, then the kinetic energy density will also be mixed. It seems for general cases, SCF converges fine even without this mixing. However, if there is difficulty in converging SCF for meta-GGA, it might be helpful to turn this on.
- **Default**: False

mixing_dftu

- **Type**: Boolean
- **Description**: Only relevant for DFT+U calculations. If set to true, then the occupation matrices will also be mixed by plain mixing. From experience this is not very helpful if the +U calculation does not converge.
- **Default**: False

gamma_only

- **Type**: Integer
- **Description**: It is an important parameter **only to be used in localized orbitals set**. If you set gamma_only = 1, ABACUS uses gamma only, the algorithm is faster and you don't need to specify the k-points file. If you set gamma_only = 0, more than one k-point is used and the ABACUS is slower compared to the gamma only algorithm.

  > Note: If gamma_only is set to 1, the KPT file will be overwritten. So make sure to turn off gamma_only for multi-k calculations.

- **Default**: 0

printe

- **Type**: Integer
- **Description**: Print out energy for each band for every printe step
- **Default**: 100

scf_nmax

- **Type**: Integer
- **Description**: This variable indicates the maximal iteration number for electronic iterations.
- **Default**: 100

scf_thr

- **Type**: Real
- **Description**: An important parameter in ABACUS. It's the threshold for electronic iteration. It represents the charge density error between two sequential densities from electronic iterations. Usually for local orbitals, usually 1e-6 may be accurate enough.
- **Default**: 1.0e-9

chg_extrap

- **Type**: String
- **Description**: Methods to do extrapolation of density when ABACUS is doing geometry relaxations.
  - atomic: atomic extrapolation
  - first-order: first-order extrapolation
  - second-order: second-order extrapolation
- **Default**: atomic

lspinorb

- **Type**: Boolean
- **Description**: whether to consider spin-orbital coupling effect in the calculation. When set to 1, `nspin` is also automatically set to 4.
- **Default**: 0

noncolin

- **Type**: Boolean
- **Description**: whether to allow non-collinear polarization, in which case the coupling between spin up and spin down will be taken into account. If set to 1, `nspin` is also automatically set to 4.
- **Default**: 0

soc_lambda

- **Type**: Real
- **Description**: Relevant for soc calculations. Sometimes, for some real materials, both scalar-relativistic and full-relativistic can not describe the exact spin-orbit coupling. Artificial modulation may help in such cases.

  `soc_lambda`, which has value range [0.0, 1.0] , is used for modulate SOC effect.

  In particular, `soc_lambda 0.0` refers to scalar-relativistic case and `soc_lambda 1.0` refers to full-relativistic case.
- **Default**: 1.0

back to top

12.1.6 Electronic structure (SDFT)

These variables are used to control the parameters of stochastic DFT (SDFT), mix stochastic-deterministic DFT (MDFT), or complete-basis Chebyshev method (CT). We suggest using SDFT to calculate high-temperature systems and we only support *smearing_method* "fd".

method_sto

- **Type**: Integer
- **Description**:
  - Different method to do SDFT.
  - 1: SDFT calculates $T_n(\hat{h})\chi$ twice, where $T_n(x)$ is the n-th order Chebyshev polynomial and $\hat{h} = \frac{\hat{H}-\bar{E}}{\Delta E}$ owning eigenvalue $\in (-1, 1)$. This method cost less memory but is slower.
  - 2: SDFT calculates $T_n(\hat{h})\chi$ once but need much more memory. This method is much faster. Besides, it calculates N_e with $\chi\sqrt{\hat{f}}\sqrt{\hat{f}}\chi$, which needs a smaller *nche_sto*. However, when memory is not enough, only method 1 can be used.
  - other: use 2
- **Default**: 2

nbands_sto

- **Type**: Integer
- **Description**:
  - nbands_sto>0: Number of stochastic orbitals to calculate in SDFT and MDFT. More bands obtain more precise results or smaller stochastic errors ($\propto 1/\sqrt{N_\chi}$);
  - nbands_sto=0: Complete basis will be used to replace stochastic orbitals with the Chebyshev method (CT) and it will get the results the same as KSDFT without stochastic errors.
  - If you want to do MDFT. *nbands* which represents the number of KS orbitals should be set.
- **Default**: 256

nche_sto

- **Type**: Integer
- **Description**: Chebyshev expansion orders for SDFT, MDFT, CT methods.
- **Default**:100

emin_sto

- **Type**: Real
- **Description**: Trial energy to guess the lower bound of eigen energies of the Hamiltonian Operator $\hat{H}$. The unit is Ry.
- **Default**:0.0

emax_sto

- **Type**: Real
- **Description**: Trial energy to guess the upper bound of eigen energies of the Hamiltonian Operator $\hat{H}$. The unit is Ry.
- **Default**:0.0

seed_sto

- **Type**: Integer
- **Description**: The random seed to generate stochastic orbitals.
  - seed_sto>=0: Stochastic orbitals have the form of exp$(i2\pi\theta(G))$, where θ is a uniform distribution in $(0, 1)$. If seed_sto = 0, the seed is decided by time(NULL).
  - seed_sto<=-1: Stochastic orbitals have the form of ± 1 with equal probability. If seed_sto = -1, the seed is decided by time(NULL).
- **Default**:0

initsto_freq

- **Type**: Integer
- **Description**: Frequency (once each initsto_freq steps) to generate new stochastic orbitals when running md.
  - positive integer: Update stochastic orbitals
  - 0: Never change stochastic orbitals.
- **Default**:0

npart_sto

- **Type**: Integer
- **Description**: Make memory cost to 1/npart_sto times of the previous one when running post process of SDFT like DOS with method_sto = 2.
- **Default**:1

back to top

12.1.7 Geometry relaxation

These variables are used to control the geometry relaxation.

relax_nmax

- **Type**: Integer
- **Description**: The maximal number of ionic iteration steps, the minimum value is 1.
- **Default**: 1

cal_force

- **Type**: Boolean
- **Description**: If set to 1, calculate the force at the end of the electronic iteration. 0 means the force calculation is turned off. It is automatically set to 1 if `calculation` is `cell-relax`, `relax`, or `md`.
- **Default**: 0

force_thr

- **Type**: Real
- **Description**: The threshold of the force convergence, it indicates the largest force among all the atoms, the unit is Ry=Bohr
- **Default**: 0.001 Ry/Bohr = 0.0257112 eV/Angstrom

force_thr_ev

- **Type**: Real
- **Description**: The threshold of the force convergence, has the same function as force_thr, just the unit is different, it is eV/Angstrom, you can choose either one as you like. The recommended value for using atomic orbitals is 0.04 eV/Angstrom.
- **Default**: 0.0257112 eV/Angstrom

force_thr_ev2

- **Type**: Real
- **Description**: The calculated force will be set to 0 when it is smaller than force_thr_ev2.
- **Default**: 0.0 eV/Angstrom

relax_bfgs_w1

- **Type**: Real
- **Description**: This variable controls the Wolfe condition for BFGS algorithm used in geometry relaxation. You can look into the paper Phys.Chem.Chem.Phys.,2000,2,2177 for more information.
- **Default**: 0.01

relax_bfgs_w2

- **Type**: Real
- **Description**: This variable controls the Wolfe condition for BFGS algorithm used in geometry relaxation. You can look into the paper Phys.Chem.Chem.Phys.,2000,2,2177 for more information.
- **Default**: 0.5

relax_bfgs_rmax

- **Type**: Real
- **Description**: This variable is for geometry optimization. It indicates the maximal movement of all the atoms. The sum of the movements from all atoms can be increased during the optimization steps. However, it will not be larger than relax_bfgs_rmax Bohr.
- **Default**: 0.8

relax_bfgs_rmin

- **Type**: Real
- **Description**: This variable is for geometry optimization. It indicates the minimal movement of all the atoms. When the movement of all the atoms is smaller than relax_bfgs_rmin Bohr, and the force convergence is still not achieved, the calculation will break down.
- **Default**: 1e-5

relax_bfgs_init

- **Type**: Real
- **Description**: This variable is for geometry optimization. It indicates the initial movement of all the atoms. The sum of the movements from all atoms is relax_bfgs_init Bohr.
- **Default**: 0.5

cal_stress

- **Type**: Integer
- **Description**: If set to 1, calculate the stress at the end of the electronic iteration. 0 means the stress calculation is turned off. It is automatically set to 1 if `calculation` is `cell-relax`.
- **Default**: 0

stress_thr

- **Type**: Real
- **Description**: The threshold of the stress convergence, it indicates the largest stress among all the directions, the unit is KBar,
- **Default**: 0.01

press1, press2, press3

- **Type**: Real
- **Description**: the external pressures along three axes, the compressive stress is taken to be positive, and the unit is KBar.
- **Default**: 0

fixed_axes

- **Type**: String
- **Description**: which axes are fixed when do cell relaxation. Possible choices are:
  - None : default; all can relax
  - volume : relaxation with fixed volume
  - shape : fix shape but change volume (i.e. only lattice constant changes)
  - a : fix a axis during relaxation
  - b : fix b axis during relaxation
  - c : fix c axis during relaxation
  - ab : fix both a and b axes during relaxation
  - ac : fix both a and c axes during relaxation
  - bc : fix both b and c axes during relaxation

Note : fixed_axes = “shape” and “volume” are only available for *relax_new* = 1

- **Default**: None

fixed_ibrav

- **Type**: Boolean
- **Description**: when set to true, the lattice type will be preserved during relaxation. Must be used along with *relax_new* set to true, and a specific *latname* must be provided

  Note: it is possible to use fixed_ibrav with fixed_axes, but please make sure you know what you are doing. For example, if we are doing relaxation of a simple cubic lattice (latname = “sc”), and we use fixed_ibrav along with fixed_axes = “volume”, then the cell is never allowed to move and as a result, the relaxation never converges.
- **Default**: False

fixed_atoms

- **Type**: Boolean
- **Description**: when set to true, the direct coordinates of atoms will be preserved during variable-cell relaxation. If set to false, users can still fix certain components of certain atoms by using the `m` keyword in `STRU` file. For the latter option, check the end of this *instruction*.
- **Default**: False

relax_method

- **Type**: String
- **Description**: The method to do geometry optimizations, note that there are two implementations of the CG method, see *relax_new*:
  - bfgs: using BFGS algorithm.
  - sd: using steepest-descent algorithm.
  - cg: using cg algorithm.
- **Default**: cg

relax_cg_thr

- **Type**: Real
- **Description**: When move-method is set to ‘cg-bfgs’, a mixed cg-bfgs algorithm is used. The ions first move according to cg method, then switched to bfgs when the maximum of force on atoms is reduced below cg-threshold. The unit is eV/Angstrom.
- **Default**: 0.5

relax_new

- **Type**: Boolean
- **Description**: At around the end of 2022 we made a new implementation of the CG method for relax and cell-relax calculations. But the old implementation was also kept. To use the new method, set relax_new to true. To use the old one, set it to false.
- **Default**: True

relax_scale_force

- **Type**: Real
- **Description**: This parameter is only relevant when `relax_new` is set to True. It controls the size of the first CG step. A smaller value means the first step along a new CG direction is smaller. This might be helpful for large systems, where it is safer to take a smaller initial step to prevent the collapse of the whole configuration.
- **Default**: 0.5

cell_factor

- **Type**: Real
- **Description**: Used in the construction of the pseudopotential tables. It should exceed the maximum linear contraction of the cell during a simulation.
- **Default**: 1.2

back to top

12.1.8 Variables related to output information

These variables are used to control the output of properties.

out_mul

- **Type**: Boolean
- **Description**: If set to 1, ABACUS will output the Mulliken population analysis result. The name of the output file is mulliken.txt
- **Default**: 0

out_freq_elec

- **Type**: Integer
- **Description**: If set to >1, it represents the frequency of electronic iters to output charge density (if *out_chg* is turned on), wavefunction (if *out_wfc_pw* or *out_wfc_r* is turned on), and density matrix of localized orbitals (if *out_dm* is turned on). If set to 0, ABACUS will output them only when converged in SCF or electronic iters reach its maximum. Used for the restart of SCF.
- **Default**: 0

out_freq_ion

- **Type**: Integer
- **Description**: If set to >1, it represents the frequency of ionic steps to output charge density (if *out_chg* is turned on) and wavefunction (if *out_wfc_pw* or *out_wfc_r* is turned on). If set to 0, ABACUS will output them only when ionic steps reach its maximum step. Used for the restart of MD or Relax.
- **Default**: 0

  Note : out_freq_ion is currently useless.

out_chg

- **Type**: Boolean
- **Description**: If set to 1, ABACUS will output the charge density on real space grid. The name of the density file is SPIN1_CHG.cube and SPIN2_CHG.cube (if nspin = 2). Suppose each density on grid has coordinate (x; y; z). The circle order of the density on real space grid is: z is the outer loop, then y and finally x (x is moving fastest).
- **Default**: 0

out_pot

- **Type**: Integer
- **Description**: If set to 1, ABACUS will output the local potential on real space grid. The name of the file is SPIN1_POT.cube and SPIN2_POT.cube (if nspin = 2). If set to 2, ABACUS will output the electrostatic potential on real space grid. The name of the file is ElecStaticPot and ElecStaticP ot_AV E (along the z-axis).
- **Default**: 0

out_dm

- **Type**: Boolean
- **Description**: If set to 1, ABACUS will output the density matrix of localized orbitals, only useful for localized orbitals set. The name of the output file is SPIN1_DM and SPIN2_DM in the output directory.
- **Default**: 0

out_wfc_pw

- **Type**: Integer
- **Description**: Only used in **planewave basis** and **ienvelope calculation in localized orbitals** set. When set this variable to 1, it outputs the coefficients of wave functions into text files. The file names are WAVEFUNC$K.txt, where$K is the index of k point. When set this variable to 2, results are stored in binary files. The file names are WAVEFUNC$K.dat.
- **Default**: 0

out_wfc_r

- **Type**: Boolean
- **Description**: Only used in **planewave basis** and **ienvelope calculation in localized orbitals** set. When set this variable to 1, it outputs real-space wave functions into `OUT.suffix/wfc_realspace/`. The file names are wfc_realspace$K$B, where $K is the index of kpoint$,B is the index of band.
- **Default**: 0

out_wfc_lcao

- **Type**: Boolean
- **Description**: **Only used in localized orbitals set**. If set to 1, ABACUS will output the wave functions coefficients.
- **Default**: 0

out_dos

- **Type**: Integer
- **Description**: Controls whether to output the density of state (DOS). The unit of DOS is `(number of states)/(eV * unitcell)`. For more information, refer to the *worked example*.
- **Default**: 0

out_band

- **Type**: Boolean
- **Description**: Controls whether to output the band structure. For more information, refer to the *worked example*
- **Default**: 0

out_proj_band

- **Type**: Boolean
- **Description**: Controls whether to output the projected band structure. For more information, refer to the *worked example*
- **Default**: 0

out_stru

- **Type**: Boolean
- **Description**: If set to 1, then the structure files will be written after each ion step
- **Default**: 0

out_bandgap

- **Type**: Boolean
- **Description**: If set to 1, the bandgap will be printed out at each SCF step.
- **Default**: 0

out_level

- **Type**: String
- **Description**: Controls the level of output. `ie` means write output at electron level; `i` means write additional output at ions level.
- **Default**: ie

out_alllog

- **Type**: Boolean
- **Description**: determines whether to write log from all ranks in an MPI run. If set to be 1, then each rank will write detained running information to a file named running_$calculation$_({rank}+1).log. If set to 0, log will only be written from rank 0 into a file named running_${calculation}.log.
- **Default**: 0

out_mat_hs

- **Type**: Boolean
- **Description**: For LCAO calculations, if out_mat_hs is set to 1, ABACUS will print the upper triangular part of the Hamiltonian matrices and overlap matrices for each k point into a series of files in the directory `OUT.${suffix}`. The files are named `data-$k-H` and `data-$k-S`, where `$k` is a composite index consisting of the k point index as well as the spin index.

  For nspin = 1 and nspin = 4 calculations, there will be only one spin component, so `$k` runs from 0 up to $nkpoints-1$. For nspin = 2, `$k` runs from $2 * nkpoints - 1$. In the latter case, the files are arranged into blocks of up and down spins. For example, if there are 3 k points, then we have the following correspondence:

  data-0-H : 1st k point, spin up data-1-H : 2nd k point, spin up data-2-H : 3rd k point, spin up data-3-H : 1st k point, spin down data-4-H : 2nd k point, spin down data-5-H : 3rd k point, spin down

  As for information on the k points, one may look for the `SETUP K-POINTS` section in the running log.

  The first number of the first line in each file gives the size of the matrix, namely, the number of atomic basis functions in the system.

  The rest of the file contains the upper triangular part of the specified matrices. For multi-k calculations, the matrices are Hermitian and the matrix elements are complex; for gamma-only calculations, the matrices are symmetric and the matrix elements are real.
- **Default**: 0

out_mat_r

- **Type**: Boolean
- **Description**: For LCAO calculations, if out_mat_r is set to 1, ABACUS will calculate and print the matrix representation of the position matrix, namely $\langle \chi_\mu|\hat{r}|\chi_\nu \rangle$ in a file named `data-rR-tr` in the directory `OUT.${suffix}`.

  Each file or each section of the appended file starts with "STEP: " followed by the current ion/md step, then the second line starts with "Matrix Dimension of $r(R)$: " followed by the dimension of the matrix, and the third line starts with "Matrix number of $r(R)$: " followed by the matrix number. The rest of the format is arranged into blocks, such as:

  ```
  -5 -5 -5    //R (lattice vector)
  ...
  -5 -5 -4    //R (lattice vector)
  ...
  -5 -5 -3    //R (lattice vector)
  ```

  Each block here contains the matrix for the corresponding cell. There are three columns in each block, giving the matrix elements in x, y, z directions, respectively. There are altogether nbasis * nbasis lines in each block, which emulates the matrix elements.

  In MD calculations, if *out_app_flag* is set to true, then `data-rR-tr` is written in an append manner. Otherwise, output files will be put in a separate directory, `matrix`, and named as `$x`_data-rR-tr, where `$x` is the number of MD step. In addition, The output frequency is controlled by *out_hs2_interval*. For example, if we are running a 10-step MD with out_hs2_interval = 3, then `$x` will be 0, 3, 6, and 9.

  > Note: This functionality is not available for gamma_only calculations. If you want to use it in gamma_only calculations, you should turn off gamma_only, and explicitly specifies that gamma point is the only k point in the KPT file.

- **Default**: 0

out_mat_hs2

- **Type**: Boolean
- **Description**: For LCAO calculations, if out_mat_hs2 is set to 1, ABACUS will generate files containing the Hamiltonian matrix $H(R)$ and overlap matrix $S(R)$.

  For single-point SCF calculations, if nspin = 1 or nspin = 4, two files `data-HR-sparse_SPIN0.csr` and `data-SR-sparse_SPIN0.csr` are generated, which contain the Hamiltonian matrix $H(R)$ and overlap matrix $S(R)$ respectively. For nspin = 2, three files `data-HR-sparse_SPIN0.csr` and `data-HR-sparse_SPIN1.csr` and `data-SR-sparse_SPIN0.csr` are created, where the first two contain $H(R)$ for spin up and spin down, respectively.

  As for molecular dynamics calculations, the format is controlled by *out_hs2_interval* and *out_app_flag* in the same manner as the position matrix as detailed in *out_mat_r*.

  Each file or each section of the appended file starts with three lines, the first gives the current ion/md step, the second gives the dimension of the matrix, and the last indicates how many different `R` are in the file.

  The rest of the files are arranged in blocks. Each block starts with a line giving the lattice vector `R` and the number of nonzero matrix elements, such as:

  ```
  -3 1 1 1020
  ```

which means there are 1020 nonzero elements in the (-3,1,1) cell.

If there is no nonzero matrix element, then the next block starts immediately on the next line. Otherwise, there will be 3 extra lines in the block, which gives the matrix in CSR format. According to Wikipedia:

The CSR format stores a sparse m × n matrix M in row form using three (one-dimensional) arrays (V, COL_INDEX, ROW_INDEX). Let NNZ denote the number of nonzero entries in M. (Note that zero-based indices shall be used here.)

– The arrays V and COL_INDEX are of length NNZ, and contain the non-zero values and the column indices of those values respectively.
– The array ROW_INDEX is of length m + 1 and encodes the index in V and COL_INDEX where the given row starts. This is equivalent to ROW_INDEX[j] encoding the total number of nonzeros above row j. The last element is NNZ , i.e., the fictitious index in V immediately after the last valid index NNZ - 1.

  Note: This functionality is not available for gamma_only calculations. If you want to use it in gamma_only calculations, you should turn off gamma_only, and explicitly specifies that gamma point is the only k point in the KPT file.

- **Default**: 0

out_mat_t

- **Type**: Boolean
- **Description**: For LCAO calculations, if out_mat_t is set to 1, ABACUS will generate files containing the kinetic energy matrix $T(R)$. The format will be the same as the Hamiltonian matrix $H(R)$ and overlap matrix $S(R)$ as mentioned in *out_mat_hs2*. The name of the files will be `data-TR-sparse_SPIN0.csr` and so on. Also controled by *out_hs2_interval* and *out_app_flag*.
- **Default**: 0

out_mat_dh

- **Type**: Boolean
- **Description**: For LCAO calculations, if out_mat_dh is set to 1, ABACUS will generate files containing the derivatives of the Hamiltonian matrix. The format will be the same as the Hamiltonian matrix $H(R)$ and overlap matrix $S(R)$ as mentioned in *out_mat_hs2*. The name of the files will be `data-dHRx-sparse_SPIN0.csr` and so on. Also controled by *out_hs2_interval* and *out_app_flag*.
- **Default**: 0

out_app_flag

- **Type**: Boolean
- **Description**: Whether output $r(R)$, $H(R)$, $S(R)$, $T(R)$, and $dH(R)$ matrices in an append manner during MD. Check input parameters *out_mat_r*, *out_mat_hs2*, *out_mat_t*, and *out_mat_dh* for more information.
- **Default**: true

out_hs2_interval

- **Type**: Integer
- **Description**: Only relevant for printing $H(R)$, $S(R)$, $T(R)$, $dH(R)$ matrices during MD. It controls the interval at which to print. Check input parameter *out_mat_hs2* for more information.
- **Default**: 1

out_element_info

- **Type**: Boolean
- **Description**: When set to 1, ABACUS will generate a new directory under OUT.suffix path named as element name such as 'Si', which contained files "Si-d1-orbital-dru.dat Si-p2-orbital-k.dat Si-s2-orbital-dru.dat Si-d1-orbital-k.dat Si-p2-orbital-r.dat Si-s2-orbital-k.dat Si-d1-orbital-r.dat Si-p2-orbital-ru.dat Si-s2-orbital-r.dat Si-d1-orbital-ru.dat Si-p-proj-k.dat Si-s2-orbital-ru.dat Si.NONLOCAL Si-p-proj-r.dat Si-s-proj-k.dat Si-p1-orbital-dru.dat Si-p-proj-ru.dat Si-s-proj-r.dat Si-p1-orbital-k.dat Si-s1-orbital-dru.dat Si-s-proj-ru.dat Si-p1-orbital-r.dat Si-s1-orbital-k.dat v_loc_g.dat Si-p1-orbital-ru.dat Si-s1-orbital-r.dat Si-p2-orbital-dru.dat Si-s1-orbital-ru.dat" for example.
- **Default**: 0

restart_save

- **Type**: Boolean
- **Description**: Only for LCAO, store charge density file and H matrix file every scf step for restart.
- **Default**: 0

restart_load

- **Type**: Boolean
- **Description**: Only for LCAO, used for restart, only if that:
  - set restart_save as true and do scf calculation before.
  - please ensure the suffix is the same as calculation before and density file and H matrix file exist. Restart from stored density file and H matrix file.
- **Default**: 0

dft_plus_dmft

- **Type**: Boolean
- **Description**: Whether to generate output to be used in dmft. It seems this functionality is not working anymore.
- **Default**: 0

rpa

- **Type**: Boolean
- **Description**: Generate output files used in rpa calculation.
- **Default**: 0

back to top

12.1.9 Density of states

These variables are used to control the calculation of DOS.

dos_edelta_ev

- **Type**: Real
- **Description**: controls the step size in writing DOS (in eV).
- **Default**: 0.01

dos_sigma

- **Type**: Real
- **Description**: controls the width of Gaussian factor when obtaining smeared DOS (in eV).
- **Default**: 0.07

dos_scale

- **Type**: Real
- **Description**: the energy range of dos output is given by (emax-emin)*(1+dos_scale), centered at (emax+emin)/2. This parameter will be used when dos_emin and dos_emax are not set.
- **Default**: 0.01

dos_emin_ev

- **Type**: Real
- **Description**: minimal range for dos (in eV). If we set it, “dos_scale” will be ignored.
- **Default**: minimal eigenenergy of $\hat{H}$

dos_emax_ev

- **Type**: Real
- **Description**: maximal range for dos (in eV). If we set it, “dos_scale” will be ignored.
- **Default**: maximal eigenenergy of $\hat{H}$

dos_nche

- **Type**: Integer
- **Description**: orders of Chebyshev expansions when using SDFT to calculate DOS
- **Default**: 100

back to top

12.1.10 NAOs

These variables are used to control the generation of numerical atomic orbitals (NAOs). NAOs is the linear combination of bessel functions.

bessel_nao_ecut

- **Type**: Real
- **Description**: energy cutoff of bessel functions.
- **Default**: same as ecutwfc

bessel_nao_tolerence

- **Type**: Real
- **Description**: tolerance when searching for the zeros of bessel functions.
- **Default**: 1.0e-12

bessel_nao_rcut

- **Type**: Real
- **Description**: cutoff radius of bessel functions.
- **Default**: 6.0

bessel_nao_smooth

- **Type**: Boolean
- **Description**: whether the bessel functions smooth at radius cutoff.
- **Default**: 1

bessel_nao_sigma

- **Type**: Real
- **Description**: energy range for smooth. See also `bessel_nao_smooth`.
- **Default**: 0.1

back to top

12.1.11 DeePKS

These variables are used to control the usage of DeePKS method (a comprehensive data-driven approach to improve the accuracy of DFT). Warning: this function is not robust enough for the current version. Please try the following variables at your own risk:

deepks_out_labels

- **Type**: Boolean
- **Description**: when set to 1, ABACUS will calculate and output descriptor for DeePKS training. In `LCAO` calculation, a path of *.orb file is needed to be specified under `NUMERICAL_DESCRIPTOR`in `STRU`file. For example:

```
NUMERICAL_ORBITAL
H_gga_8au_60Ry_2s1p.orb
O_gga_7au_60Ry_2s2p1d.orb

NUMERICAL_DESCRIPTOR
jle.orb
```

- **Default**: 0

deepks_scf

- **Type**: Boolean
- **Description**: only when deepks is enabled in `LCAO` calculation can this variable set to 1. Then, a trained, traced model file is needed for self-consistent field iteration in DeePKS method.
- **Default**: 0

deepks_model

- **Type**: String
- **Description**: the path of the trained, traced NN model file (generated by deepks-kit). used when deepks_scf is set to 1.
- **Default**: None

bessel_descriptor_lmax

- **Type**: Integer
- **Description**: the projectors used in DeePKS are bessel functions. To generate such projectors, set calculation type to `gen_bessel` and run ABACUS. The lmax of Bessel functions is specified using bessel_descriptor_lmax. See also *calculation*.
- **Default**: 2

bessel_descriptor_ecut

- **Type**: Real
- **Description**: energy cutoff of bessel functions. See also `bessel_descriptor_lmax`.
- **Default**: same as ecutwfc

bessel_descriptor_tolerence

- **Type**: Real
- **Description**: tolerance when searching for the zeros of bessel functions. See also `bessel_descriptor_lmax`.
- **Default**: 1.0e-12

bessel_descriptor_rcut

- **Type**: Real
- **Description**: cutoff radius of bessel functions. See also `bessel_descriptor_lmax`.
- **Default**: 6.0

bessel_descriptor_smooth

- **Type**: Boolean
- **Description**: whether the bessel functions smooth at radius cutoff. See also `bessel_descriptor_lmax`.
- **Default**: 1

bessel_descriptor_sigma

- **Type**: Real
- **Description**: energy range for smooth. See also `bessel_descriptor_smooth`.
- **Default**: 0.1

deepks_bandgap

- **Type**: Boolean
- **Description**: whether to include deepks bandgap correction.
- **Default**: False

deepks_out_unittest

- **Type**: Boolean
- **Description**: this is used to generate some files for constructing DeePKS unit test. Not relevant when running actual calculations. When set to 1, ABACUS needs to be run with only 1 process.
- **Default**: False

back to top

12.1.12 OFDFT: orbital free density functional theory

of_kinetic

- **Type**: string
- **Description**: the type of kinetic energy density functional, including tf, vw, wt, and tf+.
- **Default**: wt

of_method

- **Type**: string
- **Description**: the optimization method used in OFDFT.
  - cg1: Polak-Ribiere. Standard CG algorithm.
  - cg2: Hager-Zhang (generally faster than cg1).
  - tn: Truncated Newton algorithm.
- **Default**:tn

of_conv

- **Type**: string
- **Description**: criterion used to check the convergence of OFDFT.
  - energy: total energy changes less than ‘of_tole’.
  - potential: the norm of potential is less than ‘of_tolp’.
  - both: both energy and potential must satisfy the convergence criterion.
- **Default**: energy

of_tole

- **Type**: Double
- **Description**: tolerance of the energy change (in Ry) for determining the convergence.
- **Default**: 2e-6

of_tolp

- **Type**: Double
- **Description**: tolerance of potential (in a.u.) for determining the convergence.
- **Default**: 1e-5

of_tf_weight

- **Type**: Double
- **Description**: weight of TF KEDF.
- **Default**: 1

of_vw_weight

- **Type**: Double
- **Description**: weight of vW KEDF.
- **Default**: 1

of_wt_alpha

- **Type**: Double
- **Description**: parameter alpha of WT KEDF.
- **Default**: $5/6$

of_wt_beta

- **Type**: Double
- **Description**: parameter beta of WT KEDF.
- **Default**: $5/6$

of_wt_rho0

- **Type**: Double
- **Description**: the average density of system, in Bohr^-3.
- **Default**: 0

of_hold_rho0

- **Type**: Boolean
- **Description**: If set to 1, the rho0 will be fixed even if the volume of system has changed, it will be set to 1 automatically if of_wt_rho0 is not zero.
- **Default**: 0

of_read_kernel

- **Type**: Boolean
- **Description**: If set to 1, the kernel of WT KEDF will be filled from file of_kernel_file, not from formula. Only usable for WT KEDF.
- **Default**: 0

of_kernel_file

- **Type**: String
- **Description**: The name of WT kernel file.
- **Default**: WTkernel.txt

of_full_pw

- **Type**: Boolean
- **Description**: If set to 1, ecut will be ignored while collecting planewaves, so that all planewaves will be used in FFT.
- **Default**: 1

of_full_pw_dim

- **Type**: Integer
- **Description**: If of_full_pw = 1, the dimension of FFT will be restricted to be (0) either odd or even; (1) odd only; (2) even only. Note that even dimensions may cause slight errors in FFT. It should be ignorable in ofdft calculation, but it may make Cardinal B-**spline** interpolation unstable, so set `of_full_pw_dim = 1` if `nbspline != -1`.
- **Default**: 0

back to top

12.1.13 Electric field and dipole correction

These variables are relevant to electric field and dipole correction

efield_flag

- **Type**: Boolean
- **Description**: If set to true, a saw-like potential simulating an electric field is added to the bare ionic potential.
- **Default**: false

dip_cor_flag

- **Type**: Boolean
- **Description**: If dip_cor_flag == true and efield_flag == true, a dipole correction is also added to the bare ionic potential. If you want no electric field, parameter efield_amp should be zero. Must be used ONLY in a slab geometry for surface calculations, with the discontinuity FALLING IN THE EMPTY SPACE.
- **Default**: false

efield_dir

- **Type**: Integer
- **Description**: The direction of the electric field or dipole correction is parallel to the reciprocal lattice vector, so the potential is constant in planes defined by FFT grid points, efield_dir = 0, 1 or 2. Used only if efield_flag == true.
- **Default**: 2

efield_pos_max

- **Type**: Real
- **Description**: Position of the maximum of the saw-like potential along crystal axis efield_dir, within the unit cell, 0 < efield_pos_max < 1. Used only if efield_flag == true.
- **Default**: 0.5

efield_pos_dec

- **Type**: Real
- **Description**: Zone in the unit cell where the saw-like potential decreases, 0 < efield_pos_dec < 1. Used only if efield_flag == true.
- **Default**: 0.1

efield_amp

- **Type**: Real
- **Description**: Amplitude of the electric field, in ***Hartree*** a.u.; 1 a.u. = 51.4220632*10^10 V/m. Used only if efield_flag == true. The saw-like potential increases with slope efield_amp in the region from (efield_pos_max+efield_pos_dec-1) to (efield_pos_max), then decreases until (efield_pos_max+efield_pos_dec), in units of the crystal vector efield_dir. Important: the change of slope of this potential must be located in the empty region, or else unphysical forces will result.
- **Default**: 0.0

back to top

12.1.14 Gate field (compensating charge)

These variables are relevant to gate field (compensating charge)

gate_flag

- **Type**: Boolean
- **Description**: In the case of charged cells, setting gate_flag == true represents the addition of compensating charge by a charged plate, which is placed at **zgate**. Note that the direction is specified by **efield_dir**.
- **Default**: false

zgate

- **Type**: Real
- **Description**: Specify the position of the charged plate in units of the unit cell (0 <= **zgate** < 1).
- **Default**: 0.5

block

- **Type**: Boolean
- **Description**: Add a potential barrier to the total potential to avoid electrons spilling into the vacuum region for electron doping. Potential barrier is from **block_down** to **block_up** and has a height of **block_height**. If **dip_cor_flag** == true, **efield_pos_dec** is used for a smooth increase and decrease of the potential barrier.
- **Default**: false

block_down

- **Type**: Real
- **Description**: Lower beginning of the potential barrier in units of the unit cell size (0 <= **block_down** < **block_up** < 1).
- **Default**: 0.45

block_up

- **Type**: Real
- **Description**: Upper beginning of the potential barrier in units of the unit cell size (0 <= **block_down** < **block_up** < 1).
- **Default**: 0.55

block_height

- **Type**: Real
- **Description**: Height of the potential barrier in Rydberg.
- **Default**: 0.1

back to top

12.1.15 Exact Exchange

These variables are relevant when using hybrid functionals

exx_hybrid_alpha

- **Type**: Real
- **Description**: fraction of Fock exchange in hybrid functionals, so that $E_X = \alpha E_X + (1-\alpha)E_{X,\text{LDA/GGA}}$
- **Default**: 1 if dft_functional==hf else 0.25

exx_hse_omega

- **Type**: Real
- **Description**: range-separation parameter in HSE functional, such that $1/r = \text{erfc}(\omega r)/r + \text{erf}(\omega r)/r$.
- **Default**: 0.11

exx_separate_loop

- **Type**: Boolean
- **Description**: There are two types of iterative approaches provided by ABACUS to evaluate Fock exchange. If this parameter is set to 0, it will start with a GGA-Loop, and then Hybrid-Loop, in which EXX Hamiltonian H_{exx} is updated with electronic iterations. If this parameter is set to 1, a two-step method is employed, i.e. in the inner iterations, density matrix is updated, while in the outer iterations, H_{exx} is calculated based on density matrix that converges in the inner iteration. (Currently not used)
- **Default**: 1

exx_hybrid_step

- **Type**: Integer
- **Description**: This variable indicates the maximal electronic iteration number in the evaluation of Fock exchange.
- **Default**: 100

exx_lambda

- **Type**: Real
- **Description**: It is used to compensate for divergence points at G=0 in the evaluation of Fock exchange using *lcao_in_pw* method.
- **Default**: 0.3

exx_pca_threshold

- **Type**: Real
- **Description**: To accelerate the evaluation of four-center integrals $(ik|jl)$, the product of atomic orbitals are expanded in the basis of auxiliary basis functions (ABF): $\Phi_i\Phi_j \sim C^k_{ij}P_k$. The size of the ABF (i.e. number of P_k) is reduced using principal component analysis. When a large PCA threshold is used, the number of ABF will be reduced, hence the calculation becomes faster. However, this comes at the cost of computational accuracy. A relatively safe choice of the value is 1e-4.
- **Default**: 1E-4

exx_c_threshold

- **Type**: Real
- **Description**: See also the entry *exx_pca_threshold*. Smaller components (less than exx_c_threshold) of the C^k_{ij} matrix are neglected to accelerate calculation. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-4.
- **Default**: 1E-4

exx_v_threshold

- **Type**: Real
- **Description**: See also the entry *exx_pca_threshold*. With the approximation $\Phi_i\Phi_j \sim C^k_{ij}P_k$, the four-center integral in Fock exchange is expressed as $(ik|jl) = \Sigma_{a,b}C^a_{ij}V_{ab}C^b_{kl}$, where $V_{ab} = (P_a|P_b)$ is a double-center integral. Smaller values of the V matrix can be truncated to accelerate calculation. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 0, i.e. no truncation.
- **Default**: 1E-1

exx_dm_threshold

- **Type**: Real
- **Description**: The Fock exchange can be expressed as $\Sigma_{k,l}(ik|jl)D_{kl}$ where D is the density matrix. Smaller values of the density matrix can be truncated to accelerate calculation. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-4.
- **Default**: 1E-4

exx_c_grad_threshold

- **Type**: Real
- **Description**: See also the entry *exx_pca_threshold*. ∇C^k_{ij} is used in force and stress. Smaller components (less than exx_c_grad_threshold) of the ∇C^k_{ij} matrix are neglected to accelerate calculation. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-4.
- **Default**: 1E-4

exx_v_grad_threshold

- **Type**: Real
- **Description**: See also the entry *exx_pca_threshold*. With the approximation $\Phi_i\Phi_j \sim C^k_{ij}P_k$, the four-center integral in Fock exchange is expressed as $(ik|jl) = \Sigma_{a,b}C^a_{ij}V_{ab}C^b_{kl}$, where $V_{ab} = (P_a|P_b)$ is a double-center integral. ∇V_{ab} is used in force and stress. Smaller values of the V matrix can be truncated to accelerate calculation. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 0, i.e. no truncation.
- **Default**: 1E-1

exx_schwarz_threshold

- **Type**: Real
- **Description**: In practice the four-center integrals are sparse, and using Cauchy-Schwartz inequality, we can find an upper bound of each integral before carrying out explicit evaluations. Those that are smaller than exx_schwarz_threshold will be truncated. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-5. (Currently not used)
- **Default**: 0

exx_cauchy_threshold

- **Type**: Real
- **Description**: In practice the Fock exchange matrix is sparse, and using Cauchy-Schwartz inequality, we can find an upper bound of each matrix element before carrying out explicit evaluations. Those that are smaller than exx_cauchy_threshold will be truncated. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-7.
- **Default**: 1E-7

exx_cauchy_grad_threshold

- **Type**: Real
- **Description**: In practice the Fock exchange matrix in force and stress is sparse, and using Cauchy-Schwartz inequality, we can find an upper bound of each matrix element before carrying out explicit evaluations. Those that are smaller than exx_cauchy_grad_threshold will be truncated. The larger the threshold is, the faster the calculation and the lower the accuracy. A relatively safe choice of the value is 1e-7.
- **Default**: 1E-7

exx_ccp_threshold

- **Type**: Real
- **Description**: It is related to the cutoff of on-site Coulomb potentials. (Currently not used)
- **Default**: 1e-8

exx_ccp_rmesh_times

- **Type**: Real
- **Description**: This parameter determines how many times larger the radial mesh required for calculating Coulomb potential is to that of atomic orbitals. For HSE, setting it to 1 is enough. But for PBE0, a much larger number must be used.
- **Default**: 1.5 if dft_functional==hse else 10

exx_distribute_type

- **Type**: String
- **Description**: When running in parallel, the evaluation of Fock exchange is done by distributing atom pairs on different threads, then gather the results. exx_distribute_type governs the mechanism of distribution. Available options are `htime`, `order`, `kmean1` and `kmeans2`. `order` is where atom pairs are simply distributed by their orders. `hmeans` is a distribution where the balance in time is achieved on each processor, hence if the memory is sufficient, this is the recommended method. `kmeans1` and `kmeans2` are two methods where the k-means clustering method is used to reduce memory requirement. They might be necessary for very large systems. (Currently not used)
- **Default**: `htime`

exx_opt_orb_lmax

- **Type**: Integer
- **Description**: See also the entry *dft_functional*. This parameter is only relevant when dft_functional=`opt_orb`. The radial part of opt-ABFs are generated as linear combinations of spherical Bessel functions. exx_opt_orb_lmax gives the maximum l of the spherical Bessel functions. A reasonable choice is 2.
- **Default**: 0

exx_opt_orb_ecut

- **Type**: Real
- **Description**: See also the entry *dft_functional*. This parameter is only relevant when dft_functional=`opt_orb`. A plane wave basis is used to optimize the radial ABFs. This parameter thus gives the cut-off of plane wave expansion, in Ry. A reasonable choice is 60.
- **Default**: 0

exx_opt_orb_tolerence

- **Type**: Real
- **Description**: See also the entry *dft_functional*. This parameter is only relevant when dft_functional=`opt_orb`. exx_opt_orb_tolerence determines the threshold when solving for the zeros of spherical Bessel functions. A reasonable choice is 1e-12.
- **Default**: 0

exx_real_number

- **Type**: Boolean
- **Description**: If set to 1, it will enforce LIBRI to use `double` data type, otherwise, it will enforce LIBRI to use `complex` data type. The default value depends on the *gamma_only* option.
- **Default**: 1 if gamma_only else 0

back to top

12.1.16 Molecular dynamics

These variables are used to control the molecular dynamics calculations.

md_type

- **Type**: Integer
- **Description**: control the algorithm to integrate the equation of motion for md. When `md_type` is set to 0, `md_thermostat` is used to specify the thermostat based on the velocity Verlet algorithm.
  - -1: FIRE method to relax;
  - 0: velocity Verlet algorithm (default: NVE ensemble);
  - 1: Nose-Hoover style non-Hamiltonian equations of motion;
  - 2: NVT ensemble with Langevin method;
  - 4: MSST method;

    Note: when md_type is set to 1, md_tfreq is required to stabilize temperature. It is an empirical parameter whose value is system-dependent, ranging from 1/(40*md_dt) to 1/(100*md_dt). An improper choice of its value might lead to failure of job.
- **Default**: 1

md_thermostat

- **Type**: String
- **Description**: specify the thermostat based on the velocity Verlet algorithm (useful when `md_type` is set to 0).
  - nve: NVE ensemble.
  - anderson: NVT ensemble with Anderson thermostat, see the parameter `md_nraise`.
  - berendsen: NVT ensemble with Berendsen thermostat, see the parameter `md_nraise`.
  - rescaling: NVT ensemble with velocity Rescaling method 1, see the parameter `md_tolerance`.
  - rescale_v: NVT ensemble with velocity Rescaling method 2, see the parameter `md_nraise`.
- **Default**: NVE

md_nstep

- **Type**: Integer
- **Description**: the total number of md steps.
- **Default**: 10

md_restart

- **Type**: Boolean
- **Description**: to control whether restart md.
  - 0: When set to 0, ABACUS will calculate md normally.
  - 1: When set to 1, ABACUS will calculate md from the last step in your test before.
- **Default**: 0

md_dt

- **Type**: Real
- **Description**: This is the time step(fs) used in md simulation.
- **Default**: 1.0

md_tfirst, md_tlast

- **Type**: Real
- **Description**: This is the temperature (K) used in md simulation. The default value of md_tlast is md_tfirst. If md_tlast is set to be different from md_tfirst, ABACUS will automatically change the temperature from md_tfirst to md_tlast.
- **Default**: No default

md_dumpfreq

- **Type**: Integer
- **Description**: This is the frequency to dump md information.
- **Default**: 1

md_restartfreq

- **Type**: Integer
- **Description**: This is the frequency to output restart information.
- **Default**: 5

md_seed

- **Type**: Integer
- **Description**:
  - md_seed < 0: No srand() in MD initialization.
  - md_seed >= 0: srand(md_seed) in MD initialization.
- **Default**: -1

md_prec_level

- **Type**: Integer
- **Description**: Determine the precision level of vc-md.
  - 0: FFT grids do not change, only G vectors and K vectors are changed due to the change of lattice vector. This level is suitable for cases where the variation of the volume and shape is not large, and the efficiency is relatively higher.
  - 1: A reference cell is constructed at the beginning, controlled by *ref_cell_factor*. Then the reference cell is used to initialize FFT real-space grids and reciprocal space mesh instead of the initial cell. The cost will increase with the size of the reference cell.
  - 2: FFT grids change per MD step. This level is suitable for cases where the variation of the volume and shape is large, such as the MSST method. However, accuracy comes at the cost of efficiency.

    Note: this parameter is only used in variable-cell MD!
- **Default**: 0

ref_cell_factor

- **Type**: Real
- **Description**: Construct a reference cell bigger than the initial cell. Only used in variable-cell MD, if *md_prec_level* is set to 1. The reference cell has to be large enough so that the lattice vectors of the fluctuating cell do not exceed the reference lattice vectors during MD. Typically, 1.02 ~ 1.10 is sufficient. However, the cell fluctuations depend on the specific system and thermodynamic conditions. So users must test for a proper choice. This parameters should be used in conjunction with q2sigma, qcutz, and ecfixed.
- **Default**: 1.0

md_tfreq

- **Type**: Real
- **Description**: control the frequency of the temperature oscillations during the simulation. If it is too large, the temperature will fluctuate violently; if it is too small, the temperature will take a very long time to equilibrate with the atomic system.
- **Default**: 1/40/md_dt

md_tchain

- **Type**: Integer
- **Description**: number of thermostats coupled with the particles in the Nose Hoover Chain method.
- **Default**: 1

md_pmode

- **Type**: String
- **Description**: specify the NVT or NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.
  - none: NVT ensemble.
  - iso: NPT ensemble with isotropic cetl fluctuations.
  - aniso: NPT ensemble with anisotropic cetl fluctuations.
  - tri: NPT ensemble with non-orthogonal (triclinic) simulation box.
- **Default**: none

md_pcouple

- **Type**: String
- **Description**: the coupled lattice vectors will scale proportionally.
  - none: three lattice vectors scale independently.
  - xyz: lattice vectors x, y, and z scale proportionally.
  - xy: lattice vectors x and y scale proportionally.
  - xz: lattice vectors x and z scale proportionally.
  - yz: lattice vectors y and z scale proportionally.
- **Default**: none

md_pfirst, md_plast

- **Type**: Real
- **Description**: This is the target pressure (KBar) used in npt ensemble simulation, the default value of `md_plast` is `md_pfirst`. If `md_plast` is set to be different from `md_pfirst`, ABACUS will automatically change the target pressure from `md_pfirst` to `md_plast`.
- **Default**: No default

md_pfreq

- **Type**: Real
- **Description**: control the frequency of the pressure oscillations during the NPT ensemble simulation. If it is too large, the pressure will fluctuate violently; if it is too small, the pressure will take a very long time to equilibrate with the atomic system.
- **Default**: 1/400/md_dt

md_pchain

- **Type**: Integer
- **Description**: number of thermostats coupled with the barostat in the Nose Hoover Chain method.
- **Default**: 1

dump_force

- **Type**: Boolean
- **Description**: Output atomic forces into the file `MD_dump` or not. If `true`, forces will be written, otherwise forces will not be written.
- **Default**: false

dump_vel

- **Type**: Boolean
- **Description**: Output atomic velocities into the file `MD_dump` or not. If `true`, velocities will be written, otherwise velocities will not be written.
- **Default**: false

dump_virial

- **Type**: Boolean
- **Description**: Output lattice virial into the file `MD_dump` or not. If `true`, lattice virial will be written, otherwise lattice virial will not be written.
- **Default**: false

lj_rcut

- **Type**: Real
- **Description**: Cut-off radius for Leonard Jones potential (angstrom).
- **Default**: 8.5 (for He)

lj_epsilon

- **Type**: Real
- **Description**: The value of epsilon for Leonard Jones potential (eV).
- **Default**: 0.01032 (for He)

lj_sigma

- **Type**: Real
- **Description**: The value of sigma for Leonard Jones potential (angstrom).
- **Default**: 3.405 (for He)

pot_file

- **Type**: String
- **Description**: The filename of potential files for CMD such as DP.
- **Default**: graph.pb

msst_direction

- **Type**: Integer
- **Description**: the direction of shock wave for MSST.
- **Default**: 2 (z direction)

msst_vel

- **Type**: Real
- **Description**: the velocity of shock wave (Angstrom/fs) for MSST.
- **Default**: 0.0

msst_vis

- **Type**: Real
- **Description**: artificial viscosity (mass/length/time) for MSST.
- **Default**: 0.0

msst_tscale

- **Type**: Real
- **Description**: reduction in initial temperature (0~1) used to compress volume in MSST.
- **Default**: 0.01

msst_qmass

- **Type**: Real
- **Description**: Inertia of extended system variable. Used only when md_type is 4, you should set a number that is larger than 0. Note that Qmass of NHC is set by md_tfreq.
- **Default**: No default

md_damp

- **Type**: Real
- **Description**: damping parameter (fs) used to add force in Langevin method.
- **Default**: 1.0

md_tolerance

- **Type**: Real
- **Description**: Tolerance for velocity rescaling. Velocities are rescaled if the current and target temperature differ more than `md_tolerance` (Kelvin).
- **Default**: 100.0

md_nraise

- **Type**: Integer
- **Description**:
  - Anderson: the "collision frequency" parameter is given as 1/`md_nraise`;
  - Berendsen: the "rise time" parameter is given in units of the time step: tau = `md_nraise*md_dt`, so `md_dt`/tau = 1/`md_nraise`;
  - Rescale_v: every `md_nraise` steps the current temperature is rescaled to the target temperature;
- **Default**: 1

back to top

12.1.17 DFT+*U* correction

These variables are used to control DFT+U correlated parameters

dft_plus_u

- **Type**: Boolean
- **Description**: If set to 1, ABCUS will calculate plus U correction, which is especially important for correlated electron.
- **Default**: 0

orbital_corr

- **Type**: Integer
- **Description**: $l_1, l_2, l_3, \ldots$ for atom type 1,2,3 respectively.(usually 2 for d electrons and 3 for f electrons) .Specify which orbits need plus U correction for each atom. If set to -1, the correction would not be calculated for this atom.
- **Default**: None

hubbard_u

- **Type**: Real
- **Description**: Hubbard Coulomb interaction parameter U(ev) in plus U correction, which should be specified for each atom unless Yukawa potential is used.

  Note : since we only implemented the simplified scheme by Duradev, the 'U' here is actually Ueff which is given by hubbard U minus hund J.
- **Default**: 0.0

yukawa_potential

- **Type**: Boolean
- **Description**: whether to use the local screen Coulomb potential method to calculate the values of U and J. If this is set to 1, hubbard_u does not need to be specified.
- **Default**: 0

yukawa_lambda

- **Type**: Real
- **Description**: The screen length of Yukawa potential. Relevant if `yukawa_potential` is set to 1. If left to default, we will calculate the screen length as an average of the entire system. It's better to stick to the default setting unless there is a very good reason.
- **Default**: calculated on the fly.

omc

- **Type**: Integer
- **Description**: The parameter controls what form of occupation matrix control we are using. If set to 0, then no occupation matrix control is performed, and the onsite density matrix will be calculated from wavefunctions in each SCF step. If set to 1, then the first SCF step will use an initial density matrix read from a file named `initial_onsite.dm`, but for later steps, the onsite density matrix will be updated. If set to 2, the same onsite density matrix from `initial_onsite.dm` will be used throughout the entire calculation.

  Note : The easiest way to create `initial_onsite.dm` is to run a DFT+U calculation, look for a file named `onsite.dm` in the OUT.prefix directory, and make replacements there. The format of the file is rather straight-forward.
- **Default**: 0

back to top

12.1.18 vdW correction

These variables are used to control vdW-corrected related parameters.

vdw_method

- **Type**: String
- **Description**: If set to d2 ,d3_0 or d3_bj, ABACUS will calculate corresponding vdW correction, which is DFT-D2, DFT-D3(0) or DFTD3(BJ) method. And this correction includes energy and forces. `none` means that no vdW-corrected method has been used.
- **Default**: none

vdw_s6

- **Type**: Real
- **Description**: This scale factor is to optimize the interaction energy deviations. For DFT-D2, it is found to be 0.75 (PBE), 1.2 (BLYP), 1.05 (B-P86), 1.0 (TPSS), and 1.05 (B3LYP). For DFT-D3, recommended values of this parameter with different DFT functionals can be found on the webpage. The default value of this parameter in ABACUS is set to be the recommended value for PBE.
- **Default**: 0.75 if vdw_method is chosen to be d2; 1.0 if vdw_method is chosen to be d3_0 or d3_bj

vdw_s8

- **Type**: Real
- **Description**: This scale factor is only relevant for D3(0) and D3(BJ) methods. Recommended values of this parameter with different DFT functionals can be found on the webpage. The default value of this parameter in ABACUS is set to be the recommended value for PBE.
- **Default**: 0.722 if vdw_method is chosen to be d3_0; 0.7875 if vdw_method is chosen to be d3_bj

vdw_a1

- **Type**: Real
- **Description**: This damping function parameter is relevant for D3(0) and D3(BJ) methods. Recommended values of this parameter with different DFT functionals can be found on the webpage. The default value of this parameter in ABACUS is set to be the recommended value for PBE.
- **Default**: 1.217 if vdw_method is chosen to be d3_0; 0.4289 if vdw_method is chosen to be d3_bj

vdw_a2

- **Type**: Real
- **Description**: This damping function parameter is only relevant for the DFT-D3(BJ) approach. Recommended values of this parameter with different DFT functionals can be found on the webpage. The default value of this parameter in ABACUS is set to be the recommended value for PBE.
- **Default**: 1.0 if vdw_method is chosen to be d3_0; 4.4407 if vdw_method is chosen to be d3_bj

vdw_d

- **Type**: Real
- **Description**: The variable is to control the damping rate of damping function of DFT-D2.
- **Default**: 20

vdw_abc

- **Type**: Integer
- **Description**: The variable is to control whether three-body terms are calculated for DFT-D3 approaches, including D3(0) and D3(BJ). If set to 1, ABACUS will calculate three-body term, otherwise, the three-body term is not included.
- **Default**: 0

vdw_C6_file

- **Type**: String
- **Description**: This variable is relevant if the user wants to manually set the C_6 parameters in D2 method. It gives the name of the file which contains the list of C_6 parameters for each element.

  If not set, ABACUS will use the default C_6 Parameters stored in the program. The default values of C_6 for elements 1_H up to 86_Rn can be found by searching for `C6_default` in the source code. The unit is Jnm6/mol.

  Otherwise, if user wants to manually set the C_6 Parameters, they should provide a file containing the C_6 parameters to be used. An example is given by:

  ```
  H  0.1
  Si 9.0
  ```

  Namely, each line contains the element name and the corresponding C_6 parameter.
- **Default**: default

vdw_C6_unit

- **Type**: String
- **Description**: This variable is relevant if the user wants to manually set the C_6 parameters in D2 method. It specified the unit of the supplied C_6 parameters. Allowed values are: `Jnm6/mol` (meaning J·nm^{6}/mol) and `eVA`(meaning eV·Angstrom)
- **Default**: Jnm6/mol

vdw_R0_file

- **Type**: String
- **Description**: This variable is relevant if the user wants to manually set the R_0 parameters in D2 method. If not set, ABACUS will use the default C_6 Parameters stored in the program. The default values of C_6 for elements 1_H up to 86_Rn can be found by searching for `R0_default` in the source code. The unit is Angstrom.

  Otherwise, if the user wants to manually set the C_6 Parameters, they should provide a file containing the C_6 parameters to be used. An example is given by:

  ```
  Li 1.0
  Cl 2.0
  ```

  Namely, each line contains the element name and the corresponding R_0 parameter.
- **Default**: default

vdw_R0_unit

- **Type**: String
- **Description**: This variable is relevant if the user wants to manually set the R_0 parameters in D2 method. It specified the unit of the supplied C_6 parameters. Allowed values are: `A`(meaning Angstrom) and `Bohr`.
- **Default**: A

vdw_cutoff_type

- **Type**: String
- **Description**: When applying Van-der-Waals correction in periodic systems, a cutoff radius needs to be supplied to avoid infinite summation. In ABACUS, we restrict the range of correction to a supercell centered around the unit cell at origin.

  In ABACUS, we provide two ways to determine the extent of the supercell.

  When `vdw_cutoff_type` is set to `radius`, the supercell is chosen such that it is contained in a sphere centered at the origin. The radius of the sphere is specified by `vdw_cutoff_radius`.

  When `vdw_cutoff_type` is set to `period`, the extent of the supercell is specified explicitly using keyword `vdw_cutoff_period`.
- **Default**: radius

vdw_cutoff_radius

- **Type**: Real
- **Description**: If `vdw_cutoff_type` is set to `radius`, this variable specifies the radius of the cutoff sphere. For DFT-D2, the default value is 56.6918, while for DFT-D3, the default value is 95.
- **Default**: 56.6918 if vdw_method is chosen to be d2; 95 if vdw_method is chosen to be d3_0 or d3_bj

vdw_radius_unit

- **Type**: String
- **Description**: If `vdw_cutoff_type` is set to `radius`, this variable specifies the unit of `vdw_cutoff_radius`. Two values are allowed: `A`(meaning Angstrom) and `Bohr`.
- **Default**: Bohr

vdw_cutoff_period

- **Type**: Integer Integer Integer
- **Description**: If vdw_cutoff_type is set to `period`, the three integers supplied here will explicitly specify the extent of the supercell in the directions of the three basis lattice vectors.
- **Default**: 3 3 3

vdw_cn_thr

- **Type**: Real
- **Description**: Only relevant for D3 correction. The cutoff radius when calculating coordination numbers.
- **Default**: 40

vdw_cn_thr_unit

- **Type**: String
- **Description**: Unit of the coordination number cutoff. Two values are allowed: `A`(meaning Angstrom) and `Bohr`.
- **Default**: Bohr

back to top

12.1.19 Berry phase and wannier90 interface

These variables are used to control berry phase and wannier90 interface parameters.

berry_phase

- **Type**: Boolean
- **Description**: 1, calculate berry phase; 0, not calculate berry phase.
- **Default**: 0

gdir

- **Type**: Integer
- **Description**:
  - 1: calculate the polarization in the direction of the lattice vector a_1 that is defined in STRU file.
  - 2: calculate the polarization in the direction of the lattice vector a_2 that is defined in STRU file.
  - 3: calculate the polarization in the direction of the lattice vector a_3 that is defined in STRU file.
- **Default**: 3

towannier90

- **Type**: Integer
- **Description**: 1, generate files for wannier90 code; 0, no generate.
- **Default**: 0

nnkpfile

- **Type**: String
- **Description**: the file name when you run "wannier90 -pp …".
- **Default**: seedname.nnkp

wannier_spin

- **Type**: String
- **Description**: If nspin is set to 2,
  - up: calculate spin up for wannier function.
  - down: calculate spin down for wannier function.
- **Default**: up

back to top

12.1.20 TDDFT: time dependent density functional theory

td_edm

- **Type**: int
- **Description**: the method to calculate the energy density matrix.
  - 0: new method (use the original formula).
  - 1: old method (use the formula for ground state).
- **Default**: 0

td_print_eij

- **Type**: double
- **Description**: print the Eij(<\psi_i|H|\psi_j>) elements which are larger than td_print_eij. if td_print_eij <0, don't print Eij
- **Default**: -1

td_force_dt

- **Type**: Real
- **Description**: Time-dependent evolution force changes time step. (fs)
- **Default**: 0.02

td_vext

- **Type**: Boolean
- **Description**:
  - 1: add a laser material interaction (extern laser field).
  - 0: no extern laser field.
- **Default**: 0

td_vext_dire

- **Type**: String
- **Description**: If `td_vext` is true, the td_vext_dire is a string to set the number of electric fields, like "1 2" representing external electric field is added to the x and y axis at the same time. Parameters of electric field can also be written as a string, like `td_gauss_phase 0 1.5707963267948966` representing the Gauss field in the x and y directions has a phase delay of Pi/2. See below for more parameters of electric field.
  - 1: the direction of external light field is along x axis.
  - 2: the direction of external light field is along y axis.
  - 3: the direction of external light field is along z axis.
- **Default**: 1

td_stype

- **Type**: Integer
- **Description**: type of electric field in space domain
  - 0: length gauge.
  - 1: velocity gauge.
- **Default**: 0

td_ttype

- **Type**: String
- **Description**: type of electric field in time domain
  - 0: Gaussian type function.
  - 1: Trapezoid function.
  - 2: Trigonometric function.
  - 3: Heaviside function.
  - 4: HHG function.
- **Default**: 0

td_tstart

- **Type**: Integer
- **Description**: number of step where electric field start
- **Default**: 1

td_tend

- **Type**: Integer
- **Description**: number of step where electric field end
- **Default**: 100

td_lcut1

- **Type**: Double
- **Description**: cut1 of interval in length gauge E = E0 , cut1<x<cut2 E = -E0/(cut1+1-cut2) , x<cut1 or cut2<x<1
- **Default**: 0.05

td_lcut2

- **Type**: Double
- **Description**: cut2 of interval in length gauge
- **Default**: 0.05

td_gauss_freq

- **Type**: String
- **Description**: frequency of Gauss type electric field (fs^-1) amp*cos(2pi*f(t-t0)+phase)exp(-(t-t0)^2/2sigma^2)
- **Default**: 22.13

td_gauss_phase

- **Type**: String
- **Description**: phase of Gauss type electric field amp*cos(2pi*f(t-t0)+phase)exp(-(t-t0)^2/2sigma^2)
- **Default**: 0.0

td_gauss_sigma

- **Type**: String
- **Description**: sigma of Gauss type electric field (fs) amp*cos(2pi*f(t-t0)+phase)exp(-(t-t0)^2/2sigma^2)
- **Default**: 30.0

td_gauss_t0

- **Type**: String
- **Description**: step number of time center of Gauss type electric field amp*cos(2pi*f(t-t0)+phase)exp(-(t-t0)^2/2sigma^2)
- **Default**: 100

td_gauss_amp

- **Type**: String
- **Description**: amplitude of Gauss type electric field (V/A) amp*cos(2pi*f(t-t0)+phase)exp(-(t-t0)^2/2sigma^2)
- **Default**: 0.25

td_trape_freq

- **Type**: String
- **Description**: frequency of Trapezoid type electric field (fs^-1) E = amp*cos(2pif t+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pif**t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 1.60

td_trape_phase

- **Type**: String
- **Description**: phase of Trapezoid type electric field
  E = amp*cos(2pif t+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pif**t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 0.0

td_trape_t1

- **Type**: String
- **Description**: step number of time interval 1 of Trapezoid type electric field
  E = amp*cos(2pif t+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pif**t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 1875

td_trape_t2

- **Type**: String
- **Description**: step number of time interval 2 of Trapezoid type electric field
  E = amp*cos(2pif t+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pif**t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 5625

td_trape_t3

- **Type**: String
- **Description**: step number of time interval 3 of Trapezoid type electric field
  E = amp*cos(2pif t+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pif**t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 7500

td_trape_amp

- **Type**: String
- **Description**: amplitude of Trapezoid type electric field (V/A) E = amp*cos(2pi*ft*+phase) t/t1 , t<t1 E = amp*cos(2pi*f*t+phase) , t1<t<t2 E = amp*cos(2pi*f*t+phase) (1-(t-t2)/(t3-t2)) , t2<t<t3 E = 0 , t>t3
- **Default**: 2.74

td_trigo_freq1

- **Type**: String
- **Description**: frequency 1 of Trigonometric type electric field (fs^-1) amp*cos(*2pi*f1*t+phase1*)sin(*2pi*f2*t+phase2)^2
- **Default**: 1.164656

td_trigo_freq2

- **Type**: String
- **Description**: frequency 2 of Trigonometric type electric field (fs^-1) amp*cos(*2pi*f1*t+phase1*)sin(*2pi*f2*t+phase2)^2
- **Default**: 0.029116

td_trigo_phase1

- **Type**: String
- **Description**: phase 1 of Trigonometric type electric field
  amp*cos(*2pi*f1*t+phase1*)sin(*2pi*f2*t+phase2)^2
- **Default**: 0.0

td_trigo_phase2

- **Type**: String
- **Description**: phase 2 of Trigonometric type electric field
  amp*cos(*2pi*f1*t+phase1*)sin(*2pi*f2*t+phase2)^2
- **Default**: 0.0

td_trigo_amp

- **Type**: String
- **Description**: amplitude of Trigonometric type electric field (V/A) amp*cos(*2pi*f1*t+phase1*)sin(*2pi*f2*t+phase2)^2
- **Default**: 2.74

td_heavi_t0

- **Type**: String
- **Description**: step number of switch time of Heaviside type electric field E = amp , t<t0 E = 0.0 , t>t0
- **Default**: 100

td_heavi_amp

- **Type**: String
- **Description**: amplitude of Heaviside type electric field (V/A) E = amp , t<t0 E = 0.0 , t>t0
- **Default**: 2.74

out_dipole

- **Type**: Integer
- **Description**:
  - 1: Output dipole.
  - 0: do not Output dipole.
- **Default**: 0

out_efield

- **Type**: Integer
- **Description**:
  - 1: Output efield.
  - 0: do not Output efield.
- **Default**: 0

ocp

- **Type**: Boolean
- **Description**: choose whether calculating constrained DFT or not.
  - For PW and LCAO codes. if set to 1, occupations of bands will be setting of "ocp_set".
  - For TDDFT in LCAO codes. if set to 1, occupations will be constrained since second ionic step.
  - For OFDFT, this feature can't be used.
- **Default**:0

ocp_set

- **Type**: string
- **Description**: If ocp is true, the ocp_set is a string to set the number of occupancy, like 1 10 * 1 0 1 representing the 13 band occupancy, 12th band occupancy 0 and the rest 1, the code is parsing this string into an array through a regular expression.
- **Default**: none

td_val_elec_01

- **Type**: Integer
- **Description**: Only useful when calculating the dipole. Specifies the number of valence electron associated with the first element.
- **Default**: 1.0

td_val_elec_02

- **Type**: Integer
- **Description**: Only useful when calculating the dipole. Specifies the number of valence electron associated with the second element.
- **Default**: 1.0

td_val_elec_03

- **Type**: Integer
- **Description**: Only useful when calculating the dipole. Specifies the number of valence electron associated with the third element.
- **Default**: 1.0

back to top

12.1.21 Variables useful for debugging

nurse

- **Type**: Boolean
- **Description**: If set to 1, the Hamiltonian matrix and S matrix in each iteration will be written in output.
- **Default**: 0

t_in_h

- **Type**: Boolean
- **Description**: If set to 0, then kinetic term will not be included in obtaining the Hamiltonian.
- **Default**: 1

vl_in_h

- **Type**: Boolean
- **Description**: If set to 0, then local pseudopotential term will not be included in obtaining the Hamiltonian.
- **Default**: 1

vnl_in_h

- **Type**: Boolean
- **Description**: If set to 0, then non-local pseudopotential term will not be included in obtaining the Hamiltonian.
- **Default**: 1

vh_in_h

- **Type**: Boolean
- **Description**: If set to 0, then Hartree potential term will not be included in obtaining the Hamiltonian.
- **Default**: 1

vion_in_h

- **Type**: Boolean
- **Description**: If set to 0, then local ionic potential term will not be included in obtaining the Hamiltonian.
- **Default**: 1

test_force

- **Type**: Boolean
- **Description**: If set to 1, then detailed components in forces will be written to output.
- **Default**: 0

test_stress

- **Type**: Boolean
- **Description**: If set to 1, then detailed components in stress will be written to output.
- **Default**: 0

colour

- **Type**: Boolean
- **Description**: If set to 1, output to terminal will have some color.
- **Default**: 0

test_skip_ewald

- **Type**: Boolean
- **Description**: If set to 1, then ewald energy will not be calculated.
- **Default**: 0

back to top

12.1.22 Electronic conductivities

Frequency-dependent electronic conductivities can be calculated with Kubo-Greenwood formula[Phys. Rev. B 83, 235120 (2011)].

Onsager coefficients:

$L_{mn}(\omega) = (-1)^{m+n}\frac{2\pi e^2\hbar^2}{3m_e^2\omega\Omega}$

$\times \sum_{ij\alpha\mathbf{k}} W(\mathbf{k}) \left(\frac{\epsilon_{i\mathbf{k}}+\epsilon_{j\mathbf{k}}}{2} - \mu\right)^{m+n-2} \times |\langle\Psi_{i\mathbf{k}}|\nabla_\alpha|\Psi_{j\mathbf{k}}\rangle|^2$

$\times[f(\epsilon_{i\mathbf{k}}) - f(\epsilon_{j\mathbf{k}})]\delta(\epsilon_{j\mathbf{k}} - \epsilon_{i\mathbf{k}} - \hbar\omega).$

They can also be computed by j-j correlation function.

$L_{mn} = \frac{2e^{m+n-2}}{3\Omega\hbar\omega}\Im[\tilde{C}_{mn}(\omega)]$

$\tilde{C}_{mn} = \int_0^\infty C_{mn}(t)e^{-i\omega t}e^{-\frac{1}{2}(\Delta E)^2t^2}dt$

$C_{mn}(t) = -2\theta(t)\Im\left\{Tr\left[\sqrt{\hat{f}}\hat{j}_m(1-\hat{f})e^{i\frac{\hat{H}}{\hbar}t}\hat{j}_n e^{-i\frac{\hat{H}}{\hbar}t}\sqrt{\hat{f}}\right]\right\},$

where j_1 is electric flux and j_2 is thermal flux.

Frequency-dependent electric conductivities: $\sigma(\omega) = L_{11}(\omega)$.

Frequency-dependent thermal conductivities: $\kappa(\omega) = \frac{1}{e^2T}\left(L_{22} - \frac{L_{12}^2}{L_{11}}\right)$.

DC electric conductivities: $\sigma = \lim_{\omega\to 0}\sigma(\omega)$.

Thermal conductivities: $\kappa = \lim_{\omega\to 0}\kappa(\omega)$.

cal_cond

- **Type**: Boolean
- **Description**: If set to 1, electronic conductivities will be calculated. Only supported in calculations of SDFT and KSDFT_PW.
- **Default**: 0

cond_nche

- **Type**: Integer
- **Description**: Chebyshev expansion orders for stochastic Kubo Greenwood. Only used when the calculation is SDFT.
- **Default**: 20

cond_dw

- **Type**: Real
- **Description**: Frequency interval ($d\omega$) for frequency-dependent conductivities. The unit is eV.
- **Default**: 0.1

cond_wcut

- **Type**: Real
- **Description**: Cutoff frequency for frequency-dependent conductivities. The unit is eV.
- **Default**: 10.0

cond_dt

- **Type**: Real
- **Description**: t interval to integrate Onsager coefficients. The unit is a.u.
- **Default**: 0.02

cond_dtbatch

- **Type**: Integer
- **Description**: exp(iHdtcond_dtbatch) is expanded with Chebyshev expansion.
- **Default**: 4

cond_fwhm

- **Type**: Integer
- **Description**: We use gaussian functions to approximate $\delta(E) \approx \frac{1}{\sqrt{2\pi}\Delta E}e^{-\frac{E^2}{2\Delta E^2}}$. FWHM for conductivities, $FWHM = 2 * \sqrt{2\ln 2} \cdot \Delta E$. The unit is eV.
- **Default**: 0.4

cond_nonlocal

- **Type**: Boolean
- **Description**: Conductivities need to calculate velocity matrix $\psi_i \hat{v} \psi_j$ and $m\hat{v} = \hat{p} + \frac{im}{\hbar}[\hat{V}_{NL}, \hat{r}]$. If `cond_nonlocal` is false, $m\hat{v} \approx \hat{p}$.
- **Default**: True

back to top

12.1.23 Implicit solvation model

These variables are used to control the usage of implicit solvation model. This approach treats the solvent as a continuous medium instead of individual "explicit" solvent molecules, which means that the solute embedded in an implicit solvent and the average over the solvent degrees of freedom becomes implicit in the properties of the solvent bath.

imp_sol

- **Type**: Boolean
- **Description**: If set to 1, an implicit solvation correction is considered.
- **Default**: 0

eb_k

- **Type**: Real
- **Description**: The relative permittivity of the bulk solvent, 80 for water. Used only if `imp_sol` == true.
- **Default**: 80

tau

- **Type**: Real
- **Description**: The effective surface tension parameter, which describes the cavitation, the dispersion, and the repulsion interaction between the solute and the solvent that are not captured by the electrostatic terms. The unit is $Ry/Bohr^2$.
- **Default**: 1.0798e-05

sigma_k

- **Type**: Real
- **Description**: We assume a diffuse cavity that is implicitly determined by the electronic structure of the solute. `sigma_k` is the parameter that describes the width of the diffuse cavity.
- **Default**: 0.6

nc_k

- **Type**: Real
- **Description**: It determines at what value of the electron density the dielectric cavity forms. The unit is $Bohr^{-3}$.
- **Default**: 0.00037

back to top

12.2 The STRU file

- *Examples*
  - *no latname*
  - *latname fcc*
- *Structure of the file*

12.2.1 Examples

The `STRU` file contains the information about the lattice geometry, the name(s) and/or location(s) of the pseudopotential and numerical orbital files, as well as the structural information about the system. We supply two ways of specifying the lattice geometry. Below are two examples of the `STRU` file for the same system:

No latname

For this example, no need to supply any input to the variable `latname` in the INPUT file. (See *input parameters*.)

```
ATOMIC_SPECIES
Si 28.00 Si_ONCV_PBE-1.0.upf upf201 // label; mass; pseudo_file

NUMERICAL_ORBITAL
Si_gga_8au_60Ry_2s2p1d.orb //numerical_orbital_file

LATTICE_CONSTANT
10.2 // lattice scaling factor (Bohr)

LATTICE_VECTORS
0.5 0.5 0.0 // latvec1
0.5 0.0 0.5 // latvec2
0.0 0.5 0.5 // latvec3

ATOMIC_POSITIONS
```

(continues on next page)

(continued from previous page)

```
Direct //Cartesian or Direct coordinate.
Si // Element type
0.0 // magnetism(Be careful: value 1.0 refers to 1.0 bohr mag, but not fully spin up !
↪!!)
2 // number of atoms
0.00 0.00 0.00 0 0 0
0.25 0.25 0.25 1 1 1
```

latname fcc

We see that this example is a silicon fcc lattice. Apart from setting the lattice vectors manually, we also provide another solution where only the Bravais lattice type is required, and the lattice vectors will be generated automatically. For this example, we need to set `latname="fcc"` in the INPUT file. (See *input parameters*.) And the `STRU` file becomes:

```
ATOMIC_SPECIES
Si 28.00 Si_ONCV_PBE-1.0.upf // label; mass; pseudo_file

NUMERICAL_ORBITAL
Si_gga_8au_60Ry_2s2p1d.orb //numerical_orbital_file

LATTICE_CONSTANT
10.2 // lattice scaling factor (Bohr)

ATOMIC_POSITIONS
Direct //Cartesian or Direct coordinate.
Si // Element type
0.0 // magnetism
2 // number of atoms
0.00 0.00 0.00 0 0 0//the position of atoms and other parameter specify by key word
0.25 0.25 0.25 1 1 1
```

The LATTICE_VECTORS section is removed.

12.2.2 Structure of the file

The `STRU` file contains several sections, and each section must start with a keyword like `ATOMIC_SPECIES`, `NUMERICAL_ORBITAL`, or `LATTICE_CONSTANT`, etc. to signify what type of information that comes below.

- ATOMIC_SPECIES

  This section provides information about the type of chemical elements contained the unit cell. Each line defines one type of element. The user should specify the name, the mass, and the pseudopotential file used for each element. The mass of the elment is only used in molecular dynamics simulations. For electronic-structure calculations, the actual mass value isn't important. In the above example, we see information is provided for the element `Si`:

  ```
  Si 28.00 Si_ONCV_PBE-1.0.upf upf201 // label; mass; pseudo_file; pseudo_type
  ```

  Here `Si_ONCV_PBE-1.0.upf` is the pseudopotential file. When the path is not specified, the file is assumed to be located in work directory. Otherwise, please explicitly specify the location of the pseudopotential files.

  After the pseudopotential file, `upf201` is the type of pseudopotential. There are five options: `upf`(.UPF format), `upf201`(the new .UPF format), `vwr`(.vwr format), `blps`(bulk-derived local pseudopotential), and `auto`(automatically identified). If no pseudopotential type is assigned, the default value is `auto`, and the pseudopotential type will be automatically identified.

Different types of pseudopotentials can be used for different elements, but note that the XC functionals assigned by all pseudopotentials should be the same one. If not, the choice of XC functional must be set explicitly using the *dft_functional* keyword.

Common sources of the pseudopotential files include:

1. Quantum ESPRESSO.
2. SG15-ONCV.
3. DOJO.
4. BLPS.

- NUMERICAL_ORBITAL

  Numerical atomic orbitals are only needed for `LCAO` calculations. Thus this section will be neglected in calcultions with plane wave basis. In the above example, numerical atomic orbitals is specified for the element `Si`:

  ```
  Si_gga_8au_60Ry_2s2p1d.orb //numerical_orbital_file
  ```

  'Si_gga_8au_60Ry_2s2p1d.orb' is name of the numerical orbital file. Again here the path is not specified, which means that this file is located in the work directory.

  Numerical atomic orbitals may be downloaded from the official website.

- LATTICE_CONSTANT

  The lattice constant of the system in unit of Bohr.

- LATTICE_VECTORS

  The lattice vectors of the unit cell. It is a 3by3 matrix written in 3 lines. Please note that *the lattice vectors given here are scaled by the lattice constant*. This section must be removed if the type Bravais lattice is specified using the input parameter `latname`. (See *input parameters*.)

- LATTICE_PARAMETERS

  This section is only relevant when `latname` (see *input parameters*) is used to specify the Bravais lattice type. The example above is a fcc lattice, where no additional information except the lattice constant is required to determine the geometry of the lattice.

  However, for other types of Bravais lattice, other parameters might be necessary. In that case, the section `LATTICE_PARAMETERS` must be present. It contains **one single line** with some parameters (separated by blank space if multiple parameters are needed), where the number of parameters required depends on specific type of lattice.

  The three lattice vectors v1, v2, v3 (in units of lattice constant) are generated in the following way:

  – latname = "sc": the LATTICE_PARAMETERS section is not required:

  ```
      v1 = (1, 0, 0)
      v2 = (0, 1, 0)
      v3 = (0, 0, 1)
  ```

  – latname = "fcc": the LATTICE_PARAMETERS section is not required:

  ```
      v1 = (-0.5, 0, 0.5)
      v2 = (0, 0.5, 0.5)
      v3 = (-0.5, 0.5, 0)
  ```

  – latname = "bcc" : the LATTICE_PARAMETERS section is not required:

```
    v1 = (0.5, 0.5, 0.5)
    v2 = (-0.5, 0.5, 0.5)
    v3 = (-0.5, -0.5, 0.5)
```

- latname = "hexagonal" : One single parameter is required in the LATTICE_PARAMETERS section, being the ratio between axis length c/a. Denote it by x then:

```
    v1 = (1.0, 0, 0)
    v2 = (-0.5, sqrt(3)/2, 0)
    v3 = (0, 0, x)
```

- latname = "trigonal" : One single parameter is required in the LATTICE_PARAMETERS section, which specifies cosγ with γ being the angle between any pair of crystallographic vectors. Denote it by x then:

```
    v1 = (tx, -ty, tz)
    v2 = (0, 2ty, tz)
    v3 = (-tx, -ty, tz)
```

  where tx=sqrt((1-x)/2), ty=sqrt((1-x)/6), and tz=sqrt((1+2x)/3).

- latname = "st" (simple tetragonal) : One single parameter is required in the LATTICE_PARAMETERS section, which gives ratio between axis length c/a. Denote it by x then:

```
    v1 = (1, 0, 0)
    v2 = (0, 1, 0)
    v3 = (0, 0, x)
```

- latname = "bct" (body-centered tetragonal) : One single parameter is required in the LATTICE_PARAMETERS section, which gives ratio between axis length c/a. Denote it by x then:

```
    v1 = (0.5, -0.5, x)
    v2 = (0.5, 0.5, x)
    v3 = (-0.5, -0.5, x)
```

- latname = "so" (simple orthorhombic) : Two parameters are required in the LATTICE_PARAMETERS section, which gives ratios between axis length b/a and c/a. Denote them by x, y then:

```
    v1 = (1, 0, 0)
    v2 = (0, x, 0)
    v3 = (0, 0, y)
```

- latname = "baco" (base-centered orthorhombic) : Two parameters are required in the LATTICE_PARAMETERS section, which gives ratios between axis length b/a and c/a. Denote them by x, y then:

```
    v1 = (0.5, x/2, 0)
    v2 = (-0.5, x/2, 0)
    v3 = (0, 0, y)
```

- latname = "fco" (face-centered orthorhombic) : Two parameters are required in the LATTICE_PARAMETERS section, which gives ratios between axis length b/a and c/a. Denote them by x, y then:

```
    v1 = (0.5, 0, y/2)
    v2 = (0.5, x/2, 0)
    v3 = (0, x/2, y/2)
```

- latname = "bco" (body-centered orthorhombic) : Two parameters are required in the LATTICE_PARAMETERS section, which gives ratios between lattice vector length b/a and c/a. Denote them by x, y then:

```
v1 = (0.5, x/2, y/2)
v2 = (-0.5, x/2, y/2)
v3 = (-0.5, -x/2, y/2)
```

- latname = "sm" (simple monoclinic) : Three parameters are required in the LATTICE_PARAMETERS section, which are the ratios of lattice vector length b/a, c/a as well as the cosine of angle between axis a and b. Denote them by x, y, z then:

```
v1 = (1, 0, 0)
v2 = (x*z, x*sqrt(1-z^2, 0)
v3 = (0, 0, y)
```

- latname = "bacm" (base-centered monoclinic) : Three parameters are required in the LATTICE_PARAMETERS section, which are the ratios of lattice vector length b/a, c/a as well as the cosine of angle between axis a and b. Denote them by x, y, z then:

```
v1 = (0.5, 0, -y/2)
v2 = (x*z, x*sqrt(1-z^2), 0)
v3 = (0.5, 0, y/2)
```

- latname = "triclinic" : Five parameters are required in the LATTICE_PARAMETERS section, namely the ratios b/a, c/a; the cosines of angle ab, ac, bc. Denote them by x,y,m,n,l, then:

```
v1 = (1, 0, 0)
v2 = (x*m, x*sqrt(1-m^2), 0)
v3 = (y*n, y*(l-n*m/sqrt(1-m^2)), y*fac)
```

  where $fac = \frac{\sqrt{1+2*m*n*l-m^2-n^2-l^2}}{\sqrt{1-m^2}}$

• ATOMIC_POSITIONS

This section specifies the positions and other information of individual atoms.

The first line signifies method that atom positions are given, the following options are supported:

- `Direct` : coordinates of atom positions below would in fraction coordinates.
- `Cartesian` : Cartesian coordinates in unit of 'LATTICE_CONSTANT'.
- `Cartesian_au` : Cartesian coordinates in unit of Bohr, same as setting of `Cartesian` with `LATTICE_CONSTANT` = 1.0 .
- `Cartesian_angstrom` : Cartesian coordinates in unit of Angstrom, same as setting of `Cartesian` with `LATTICE_CONSTANT` = 1.889726125457828 .
- `Cartesian_angstrom_center_xy` : Cartesian coordinates in unit of Angstrom, with Direct coordinate (0.5, 0.5, 0.0) as reference.
- `Cartesian_angstrom_center_xz` : Cartesian coordinates in unit of Angstrom, with Direct coordinate (0.5, 0.0, 0.5) as reference…
- `Cartesian_angstrom_center_yz` : Cartesian coordinates in unit of Angstrom, with Direct coordinate (0.0, 0.5, 0.5) as reference…
- `Cartesian_angstrom_center_xyz` : Cartesian coordinates in unit of Angstrom, with Direct coordinate (0.5, 0.5, 0.5) as reference…

The following three lines tells the elemental type (`Fe`), the initial magnetic moment (`1.0`), and the number of atoms for this particular element (`2`) repsectively. Notice this magnetic moment will be a default value for every atom of this type but will be overrided if one define it for each atom by keyword(see below).

The last two lines in this example are the coordinates of atomic positions. There are three numbers in each line, which specifies the atomic positions, following by other parameters marked by keywords.

Several other parameters could be defined after the atom position using key word :

- `m` or NO key word: three numbers, which take value in 0 or 1, control how the atom move in geometry relaxation calculations. In example below, the numbers `0 0 0` following the coordinates of the first atom means this atom are *not allowed* to move in all three directions, and the numbers `1 1 1` following the coordinates of the second atom means this atom *can* move in all three directions.
- `v` or `vel` or `velocity`: set the three components of initial velocity of atoms in geometry relaxation calculations(e. g. `v 1.0 1.0 1.0`).
- `mag` or `magmom` : set the start magnetization for each atom. In colinear case only one number should be given. In non-colinear case one have two choice:either set one number for the norm of magnetization here and specify two polar angle later(e. g. see below), or set three number for the xyz commponent of magnetization here (e. g. `mag 0.0 0.0 1.0`). Note that if this parameter is set, the initial magnetic moment setting in the second line will be invalid.
- `angle1`: in non-colinear case, specify the angle between c-axis and real spin, in angle measure instead of radian measure
- `angle2`: in non-colinear case, specify angle between a-axis and real spin in projection in ab-plane , in angle measure instead of radian measure

  e.g.:

```
Fe
1.0
2
0.0 0.0 0.0 m 0 0 0 mag 1.0 angle1 90 angle2 0
0.5 0.5 0.5 m 1 1 1 mag 1.0 angle1 90 angle2 180
```

12.3 The KPT file

- *Generate k-mesh automatically*
- *Set k-points explicitly*
- *Band structure calculations*

ABACUS uses periodic boundary conditions for both crystals and finite systems. For isolated systems, such as atoms, molecules, clusters, etc., one uses the so-called supercell model. Lattice vectors of the supercell are set in the `STRU` file. For the input k-point (`KPT`) file, the file should either contain the k-point coordinates and weights or the mesh size for creating the k-point gird. Both options are allowed in `ABACUS`.

12.3.1 Gamma-only Calculations

In ABACUS, we offer th option of running gamma-only calculations for LCAO basis by setting *gamma_only* to be 1. Due to details of implementation, gamma-only calculation will be slightly faster than running a non gamma-only calculation and explicitly setting gamma point to be the only the k-point, but the results should be consistent.

> If gamma_only is set to 1, the KPT file will be overwritten. So make sure to turn off gamma_only for multi-k calculations.

12.3.2 Generate k-mesh automatically

To generate k-mesh automatically, it requires the input subdivisions of the Brillouin zone in each direction and the origin for the k-mesh. ABACUS uses the Monkhorst-Pack method to generate k-mesh, and the following is an example input k-point (`KPT`) file used in `ABACUS`.

```
K_POINTS //keyword for start
0 //total number of k-point, `0' means generate automatically
Gamma //which kind of Monkhorst-Pack method, `Gamma' or `MP'
2 2 2 0 0 0 //first three number: subdivisions along recpri. vectors
            //last three number: shift of the mesh
```

In the above example, the first line is a keyword, and it can be set as `K_POINTS`, or `KPOINTS` or just `K`. The second line is an integer, and its value determines how to get k-points. In this example, `0` means using Monkhorst-Pack (MP) method to generate k-points automatically.

The third line tells the input type of k-points, `Gamma` or `MP`, different Monkhorst Pack (MP) method. Monkhorst-Pack (MP) is a method which uses the uniform k-points sampling in Brillouin-zone, while `Gamma` means the Γ-centered Monkhorst-Pack method. The first three numbers of the last line are integers, which give the MP k grid dimensions, and the rest three are real numbers, which give the offset of the k grid. In this example, the numbers `0 0 0` means that there is no offset, and this is the a standard 2by2by2 k grid.

back to top

12.3.3 Set k-points explicitly

If the user wants to set up the k-points explicitly, the input k-point file should contain the k-point coordinates and weights. An example is given as follows:

```
K_POINTS //keyword for start
8 //total number of k-point
Direct //`Direct' or `Cartesian' coordinate
0.0 0.0 0.0 0.125 //coordinates and weights
0.5 0.0 0.0 0.125
0.0 0.5 0.0 0.125
0.5 0.5 0.0 0.125
0.0 0.0 0.5 0.125
0.5 0.0 0.5 0.125
0.0 0.5 0.5 0.125
0.5 0.5 0.5 0.125
```

back to top

12.3.4 Band structure calculations

ABACUS uses specified high-symmetry directions of the Brillouin zone for band structure calculations. The third line of k-point file should start with 'Line' or 'Line_Cartesian' for line mode. 'Line' means the positions below are in Direct coordinates, while 'Line_Cartesian' means in Cartesian coordinates:

```
K_POINTS // keyword for start
6 // number of high symmetry lines
Line // line-mode
0.5 0.0 0.5 20 // X
0.0 0.0 0.0 20 // G
0.5 0.5 0.5 20 // L
0.5 0.25 0.75 20 // W
0.375 0.375 0.75 20 // K
0.0 0.0 0.0 1 // G
```

The fourth line and the following are special k-point coordinates and number of k-points between this special k-point and the next.

back to top

CHAPTER

THIRTEEN

HOW TO CITE

The following references are required to be cited when using ABACUS. Specifically:

- **For general purpose:**

  Mohan Chen, G. C. Guo, and Lixin He. "Systematically improvable optimized atomic basis sets for ab initio calculations." Journal of Physics: Condensed Matter 22.44 (2010): 445501.

  Pengfei Li, et al. "Large-scale ab initio simulations based on systematically improvable atomic basis." Computational Materials Science 112 (2016): 503-517.

- **If Stochastic DFT is used:**

  Qianrui Liu, and Mohan Chen. "Plane-Wave-Based Stochastic-Deterministic Density Functional Theory for Extended Systems." https://arxiv.org/abs/2204.05662.

- **If DFT+U is used:**

  Xin Qu, et al. "DFT+ U within the framework of linear combination of numerical atomic orbitals." The Journal of Chemical Physics (2022).

- **If second generation numerical orbital basis is used:**

  Peize Lin, Xinguo Ren, and Lixin He. "Strategy for constructing compact numerical atomic orbital basis sets by incorporating the gradients of reference wavefunctions." Physical Review B 103.23 (2021): 235131.

- **If berry curvature calculation is used in LCAO base:**

  Gan Jin, Daye Zheng, and Lixin He. "Calculation of Berry curvature using non-orthogonal atomic orbitals." Journal of Physics: Condensed Matter 33.32 (2021): 325503.

- **If DeePKS is used:**

  Wenfei Li, Qi Ou, et al. "DeePKS+ABACUS as a Bridge between Expensive Quantum Mechanical Models and Machine Learning Potentials." https://arxiv.org/abs/2206.10093.

- **If hybrid functional is used:**

  Peize Lin, Xinguo Ren, and Lixin He. "Efficient Hybrid Density Functional Calculations for Large Periodic Systems Using Numerical Atomic Orbitals." Journal of Chemical Theory and Computation 2021, 17(1), 222–239.

  Peize Lin, Xinguo Ren, and Lixin He. "Accuracy of Localized Resolution of the Identity in Periodic Hybrid Functional Calculations with Numerical Atomic Orbitals." Journal of Physical Chemistry Letters 2020, 11, 3082-3088.

CHAPTER FOURTEEN

DEVELOPMENT TEAM

The current development team consists the following research groups/affiliations:

- University of Science and Technology of China (Dr. Lixin He)
- Peking University (Dr. Mohan Chen)
- Institute of Physics, Chinese Academy of Sciences (Dr. Xinguo Ren)
- Beijing AI for Science Institute
- Institute of Artificial Intelligence, Hefei Comprehensive National Science Center.

CHAPTER

FIFTEEN

ABACUS CONTRIBUTION GUIDE

15.1 Contribution Process

We welcome contributions from the open source community. The technical guide is provided in *Contributing to ABACUS*. Here is the basic contribution process:

- **Find out issues to work on.** We assume you already have a good idea on what to do, otherwise the issue tracker and discussion panel provide good starting points to find out what to work on and to get familiar with the project.
- **Approach the issue.** It is suggested to submit new issues before coding out changes to involve more discussions and suggestions from development team. Refer to the technical guide in *Contributing to ABACUS* when needed.
- **Open a pull request.** The ABACUS developers review the pull request (PR) list regularly. If the work is not ready, convert it to draft until finished, then you can mark it as "Ready for review". It is suggested to open a new PR through forking a repo and creating a new branch on you Github account. A new PR should include as much information as possible in `description` when submmited. Unittests or CI tests are required for new PRs.
- **Iterate the pull request.** All pull requests need to be tested through CI before reviewing. A pull request might need to be iterated several times before accepted, so splitting a long PR into parts reduces reviewing difficulty for us.

CHAPTER

SIXTEEN

CONTRIBUTING TO ABACUS

First of all, thank you for taking time to make contributions to ABACUS! This file provides the more technical guidelines on how to realize it. For more non-technical aspects, please refer to the *ABACUS Contribution Guide*

16.1 Table of Contents

16.2 Got a question?

Please referring to our GitHub issue tracker, and our developers are willing to help. If you find a bug, you can help us by submitting an issue to our GitHub Repository. Even better, you can submit a Pull Request with a patch. You can request a new feature by submitting an issue to our GitHub Repository. If you would like to implement a new feature, please submit an issue with a proposal for your work first, and that ensures your work collaborates with our development road map well. For a major feature, first open an issue and outline your proposal so that it can be discussed. This will also allow us to better coordinate our efforts, prevent duplication of work, and help you to craft the change so that it is successfully accepted into the project.

16.3 Structure of the package

Please refer to *our instructions* on how to installing ABACUS. The source code of ABACUS is based on several modules. Under the ABACUS root directory, there are the following folders:

- `cmake`: relevant files for finding required packages when compiling the code with cmake;
- `docs`: documents and supplementary info about ABACUS;
- `examples`: some examples showing the usage of ABACUS;
- `source`: the source code in separated modules, under which a `test` folder for its unit tests;
- `tests`: End-to-end test cases;
- `tools`: the script for generating the numerical atomic orbitals.

16.4 Submitting an Issue

Before you submit an issue, please search the issue tracker, and maybe your problem has been discussed and fixed. You can submit new issues by filling our issue forms. To help us reproduce and confirm a bug, please provide a test case and building environment in your issue.

16.5 Comment style for documentation

ABACUS uses Doxygen to generate docs directly from `.h` and `.cpp` code files.

For comments that need to be shown in documents, these formats should be used – **Javadoc style** (as follow) is recommended, though Qt style is also ok. See it in official manual.

A helpful VS Code extension – Doxygen Documentation Generator, can help you formating comments.

An practical example is class LCAO_Deepks, the effects can be seen on readthedocs page

- Tips
  - Only comments in .h file will be visible in generated by Doxygen + Sphinx;
  - Private class members will not be documented;
  - Use Markdown features, such as using a empty new line for a new paragraph.
- Detailed Comment Block

```
/**
* ... text ...
*/
```

- Brief + Detailed Comment Block

```
/// Brief description which ends at this dot. Details follow
/// here.

/// Brief description.
/** Detailed description. */
```

- Comments After the Item: Add a “<”

```
int var; /**<Detailed description after the member */
int var; ///<Brief description after the member
```

- Parameters usage: `[in],[out],[in,out] description` *e.g.*

```
void foo(int v/**< [in] docs for input parameter v.*/);
```

or use `@param` command.

- Formula
  - inline: `\f$myformula\f$`
  - separate line: `\f[myformula\f]`
  - environment: `\f{environment}{myformula}`
  - *e.g.*

```
\f{eqnarray*}{
        g &=& \frac{Gm_2}{r^2} \\
        &=& \frac{(6.673 \times 10^{-11}\,\mbox{m}^3\,\mbox{kg}^{-1}\,
            \mbox{s}^{-2})(5.9736 \times 10^{24}\,\mbox{kg})}{(6371.01\,\mbox
↪{km})^2} \\
        &=& 9.82066032\,\mbox{m/s}^2
\f}
```

16.6 Code formatting style

We use `clang-format` as our code formatter. The `.clang-format` file in root directory describes the rules to conform with. For Visual Studio Code developers, the official extension of C/C++ provided by Microsoft can help you format your codes following the rules. With this extension installed, format your code with `shift+command/alt+f`. Configure your VS Code settings as `"C_Cpp.clang_format_style": "file"` (you can look up this option by pasting it into the search box of VS Code settings page), and all this stuff will take into effect. You may also set `"editor.formatOnSave": true` to avoid formatting files everytime manually.

16.7 Adding a unit test

We use GoogleTest as our test framework. Write your test under the corresponding module folder at `abacus-develop/tests`, then append the test to `tests/CMakeLists.txt`. If there are currently no unit tests provided for the module, do as follows. `module_base` provides a simple demonstration.

- Add a folder named `test` under the module.
- Append the content below to `CMakeLists.txt` of the module:

```
IF (BUILD_TESTING)
  add_subdirectory(test)
endif()
```

- Add a blank `CMakeLists.txt` under `module*/test`.

To add a unit test:

- Write your test under `GoogleTest` framework.

- Add your testing source code with suffix `*_test.cpp` in `test` directory.
- Append the content below to `CMakeLists.txt` of the module:

```
AddTest(
  TARGET <module_name>_<test_name> # this is the executable file name of the test
  SOURCES <test_name>.cpp

  # OPTIONAL: if this test requires external libraries, add them with "LIBS"␣
↪statement.
  LIBS math_libs # `math_libs` includes all math libraries in ABACUS.
)
```

- Build with `-D BUILD_TESTING=1` flag. You can find built testing programs under `build/source/<module_name>/test`.
- Follow the installing procedure of CMake. The tests will move to `build/test`.

16.8 Debugging the codes

For the unexpected results when developing ABACUS, GDB will come in handy.

1. Compile ABACUS with debug mode.

```
cmake -B build -DCMAKE_BUILD_TYPE=Debug
```

2. After building and installing the executable, enter the input directory, and launch the debug session with `gdb abacus`. For debugging in Visual Studio Code, please set cwd to the input directory, and program to the path of ABACUS executable.
3. Set breakpoints, and run ABACUS by typing "run" in GDB command line interface. If the program hits the breakpoints or exception is throwed, GDB will stop at the erroneous code line. Type "where" to show the stack backtrace, and "print i" to get the value of variable i.
4. For debugging ABACUS in multiprocessing situation, `mpirun -n 1 gdb abacus : -n 3 abacus` will attach GDB to the master process, and launch 3 other MPI processes.

For segmentation faults, ABACUS can be built with Address Sanitizer to locate the bugs. This feature requires a GCC or Clang compiler, and does not support Intel compiler.

```
cmake -B build -DENABLE_ASAN=1
```

Run ABACUS as usual, and it will automatically detect the buffer overflow problems and memory leaks. It is also possible to use GDB with binaries built by Address Sanitizer.

Valgrind is another option for performing dynamic analysis.

16.9 Generating code coverage report

This feature requires using GCC compiler. We use `gcov` and `lcov` to generate code coverage report.

1. Add `-DENABLE_COVERAGE=ON` for CMake configure command.

```
cmake -B build -DBUILD_TESTING=ON -DENABLE_COVERAGE=ON
```

2. Build, install ABACUS, and run test cases. Please note that since all optimizations are disabled to gather running status line by line, the performance is drastically decreased. Set a longer time out to ensure all tests are executed.

```
cmake --build build --target test ARGS="-V --timeout 21600"
```

3. Generate HTML report.

```
cd build/
make lcov
```

Now you can copy `build/lcov` to your local device, and view `build/lcov/html/all_targets/index.html`.

We use Codecov to host and visualize our **code coverage report**. Analysis is scheduled after a new version releases; this action can also be manually triggered.

16.10 Submitting a Pull Request

1. Fork the ABACUS repository. If you already had an existing fork, sync the fork to keep your modification up-to-date.
2. Pull your forked repository, create a new git branch, and make your changes in it:

```
git checkout -b my-fix-branch
```

3. Coding your patch, including appropriate test cases and docs. To run a subset of unit test, use `ctest -R <test-match-pattern>` to perform tests with name matched by given pattern.
4. After tests passed, commit your changes *with a proper message*.
5. Push your branch to GitHub:

```
git push origin my-fix-branch
```

6. In GitHub, send a pull request (PR) with `deepmodeling/abacus-develop:develop` as the base repository. It is required to document your PR following *our guidelines*.
7. After your pull request is merged, you can safely delete your branch and sync the changes from the main (upstream) repository:

- Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```
git push origin --delete my-fix-branch
```

- Check out the master branch:

```
git checkout develop -f
```

- Delete the local branch:

```
git branch -D my-fix-branch
```

- Update your master with the latest upstream version:

```
git pull --ff upstream develop
```

16.11 Commit message guidelines

A well-formatted commit message leads a more readable history when we look through some changes, and helps us generate change log. We follow up The Conventional Commits specification for commit message format. This format is also required for PR title and message. The commit message should be structured as follows:

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

- Header
  - type: The general intention of this commit
    - `Feature`: A new feature
    - `Fix`: A bug fix
    - `Docs`: Only documentation changes
    - `Style`: Changes that do not affect the meaning of the code
    - `Refactor`: A code change that neither fixes a bug nor adds a feature
    - `Perf`: A code change that improves performance
    - `Test`: Adding missing tests or correcting existing tests
    - `Build`: Changes that affect the build system or external dependencies
    - `CI`: Changes to our CI configuration files and scripts
    - `Revert`: Reverting commits
  - scope: optional, could be the module which this commit changes; for example, `orbital`
  - description: A short summary of the code changes: tell others what you did in one sentence.
- Body: optional, providing detailed, additional, or contextual information about the code changes, e.g. the motivation of this commit, referenced materials, the coding implementation, and so on.
- Footer: optional, reference GitHub issues or PRs that this commit closes or is related to. Use a keyword to close an issue, e.g. “Fix #753”.

Here is an example:

```
Fix(lcao): use correct scalapack interface.

`pzgemv_` and `pzgemm_` used `double*` for alpha and beta parameters but not␣
↪`complex*` , this would cause error in GNU compiler.

Fix #753.
```

CHAPTER

SEVENTEEN

FREQUENTLY ASKED QUESTIONS

17.1 General Questions

1. What are the merits of ABACUS in respect of functionality, performance, and/or accuracy?

Users are referred to the introduction of features of ABACUS in the feature list.

17.2 Installation

17.3 Setting up jobs

1. Why pseudopotential files must be provided in LCAO calculation?

The pseudopotentials are used to approximate the potential of nuclear and core electrons, while the numerical orbitals are basis sets used to expand the Hamiltonian. So both pseudopotential and numerical orbital files are needed in LCAO calculation.

2. What is the correlation between pseudopotential and numerical orbital files?

The numerical orbital files are generated for a specific pseudopotential. So the right numerical orbital files must be chosen for a specific pseudopotential. We suggest users choose numerical orbital files and the corresponding pseudopotential files from the ABACUS website because their accuracy has been tested. However, interesting users may also generate their own numerical orbitals for a specific type of pseudopential by using the tools provided in the abacus-develop/tools directory.

3. How to set `ecutwfc` in LCAO calculation? Must it be 100 Ry for a numerical orbital file like `Cu_lda_7.0au_100Ry_2s2p2d`?

It is recommended to set `ecutwfc` to the value that the numerical orbital file suggests, but it is not a must. The `ecutwfc` value only affects the number of FFT grids.

4. Does ABACUS support LCAO calculations accounting for external electric field effects?

Yes, users are referred to documentation on *external electric field*.

5. Can ABACUS calculate non-periodic systems, such as ionic liquids?

Non-periodic systems such as liquid systems can be calculated by using supercell and gamma-only calculation.

6. How to perform spin-orbital coupling (SOC) calculations in ABACUS?

Apart from setting relavant keys (`lspinorb` to 1) in the `INPUT` file, SOC calculations can only be performed with fully-relativistic pseudopotentials. Users are suggested to download fully-relativistic versions of SG15_ONCV pseudopotential files from a website. The numerical orbital files generated from the corresponding scalar-relativistic pseudoptential files by ABACUS (here) can be used in collaboration with the fully-relativistic pseudopotentials.

7. How to restart jobs in abacus?

For restarting SCF calculations, users are referred to the documentation about *continuation of job*. For restarting MD calculations, please see *md_restart*.

8. Can DeePKS model be used for structural optimization calculation? What parameters need to be modified or called?

If you train the DeePKS model with force labels, then the DeePKS model can provide force calculation with the same accuracy as your target method, and can thus be used for structural optimization. To do that, you just need to train the model with force label enabled.

back to top

17.4 Failed jobs

1. Why my calculation is pend by using mpirun?

This is usually caused by overloading of CPUs' memory without specifying thread numbers. ABACUS detects available hardware threads, and provides these information at the beginning of the program if used threads mismatches with hardware availability. User should keep total used threads(i.e. the number of OpenMP threads times the number of MPI processes) no more than the count of available CPUs(can be determined by `lscpu`). Setting `export OMP_NUM_THREADS=1` will solve this problem, or one can use the command like `OMP_NUM_THREADS=1 mpirun -n 8 abacus` to rerun failed jobs.

2. My relaxation failed. How to deal with it?

This is usually caused by the difficulty in converging charge density. Reducing charge mixing coefficient (`mixing_beta`) might help. For large systems up to hundreds of atoms, it is suggested to choose the Kerker mixing method by setting parameter "mixing_gg0" as "1.5".

Sometimes, loose convergence threshold of charge density (parameter "scf_thr") will cause atomic forces not correctly enough, please set it at most "1e-7" for relaxation calculation.

3. Why the program is halted?

If the program prompt something like "KILLED BY SIGNAL: 9 (Killed)", it may be caused by insufficient memory. You can use `dmesg` to print out system info regarding memory management, and check if there is "Out of memory: Killed" at the end of output info. Please try using less processes and threads for calculation, or modify the input parameters requiring less memory.

If the error message is "Segmentation fault", or there is no enough information on the error, please feel free to submit an issue.

17.5 Miscellaneous

1. How to visualize charge density file?

The output file SPIN1_CHG.cube can be visualized by using VESTA.

2. How to change cif file directly to STRU file?

One way to change from cif to STRU is to use the ASE-ABACUS interface. An example of the converting script is provided below:

```python
from ase.io import read, write
from pathlib import Path

cs_dir = './'
cs_cif = Path(cs_dir, 'SiO.cif')
cs_atoms = read(cs_cif, format='cif')
cs_stru = Path(cs_dir, 'STRU')
pp = {'Si':'Si.upf','O':'O.upf'}
basis = {'Si':'Si.orb','O':'O.orb'}
write(cs_stru, cs_atoms, format='abacus', pp=pp, basis=basis)
```

3. What is the convergence criterion for the SCF process in ABACUS?

ABACUS applies the density difference between two SCF steps (labeled as `DRHO` in the screen output) as the convergence criterion, which is considered as a more robust choice compared with the energy difference. `DRHO` is calculated via `DRHO = |rho(G)-rho_previous(G)|^2`. Note that the energy difference between two SCF steps (labed as `EDIFF`) is also printed out in the screen output.

back to top