ABACUS

ABACUS

Jun 14, 2026

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 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 use *toolchain* to install ABACUS and dependencies in a source-code compilation way with convience. 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. For a scripted one-click setup that provisions WSL2 and installs ABACUS inside it automatically, see *Windows One-Click Installer*.

1.1 Get ABACUS source code

ABACUS source code 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`. This Gitee repo is updated synchronously with GitHub.

1.2 Update to latest release by git

Please check the release page for the release note of a new version.

It is OK to download the new source code from beginning following the previous step.

You can update your cloned git repo (from Github or Gitee) in-place with the following commands:

```
git remote -v
# Check if the output contains the line below
# origin https://github.com/deepmodeling/abacus-develop.git (fetch)
# The remote name is marked as "upstream" if you clone the repo from your own fork.

# Replace "origin" with "upstream" or the remote name corresponding to deepmodeling/
↪abacus-develop if necessary
git fetch origin
```

(continues on next page)

(continued from previous page)

```
git checkout v3.x.x # Replace the tag with the latest version, like v3.10.0
git describe --tags # Verify if the tag has been successfully checked out
```

Then proceed to the *Build and Install* part. If you encountered errors, try remove the `build` directory first and reconfigure.

To use the codes under active development:

```
git checkout develop
git pull
```

1.3 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).

1.4 Install by toolchain

We offer a set of toolchain scripts to compile and install all the requirements and ABACUS itself automatically and suitable for machine characteristic in an online or offline way. The toolchain can be downloaded with ABACUS repo, and users can easily compile the requirements by running *toolchain_[gnu,intel,gcc-aocl,aocc-aocl].sh* and ABACUS itself by running *build_abacus_[gnu,intel,gcc-aocl,aocc-aocl].sh* script in the toolchain directory in `GNU`, `Intel-oneAPI`, `GCC-AMD AOCL` and `AMD AOCC-AOCL` toolchain. Sometimes, ABACUS by toolchain installation may have better efficient performance due to the suitable compiled dependencies. One should read the README in toolchain for most of the information before use, and related tutorials can be accessed via ABACUS WeChat platform.

1.5 Install by conda

Conda is a package management system with a separated environment, not requiring system privileges. You can refer to DeepModeling conda FAQ for how to setup a conda environment. A pre-built ABACUS binary with all requirements is available at conda-forge. It supports advanced features including Libxc, LibRI, and DeePKS. Conda will install the GPU-supported version of ABACUS if a valid GPU driver is present. Please refer to *the advanced installation guide* for more details.

```
# Install
# We recommend installing ABACUS in a new environment to avoid potential conflicts:
conda create -n abacus_env abacus "libblas=*=*mkl" mpich -c conda-forge

# Run
conda activate abacus_env
OMP_NUM_THREADS=1 mpirun -n 4 abacus

# Update
conda update -n abacus_env abacus -c conda-forge
```

If OpenBLAS gives warning about OpenMP threads, please install conda package `"openblas=*=openmp*"` or `"libblas=*=*mkl"`. See switching BLAS implementation in conda.

ABACUS supports `OpenMPI` and `MPICH` variant. Install `mpich` or `openmpi` package to switch MPI library if required.

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

Note: The deepmodeling conda channel offers historical versions of ABACUS.

1.5.1 Developing with conda

It is possible to build ABACUS from source based on the conda environment.

```
conda create -n abacus_env abacus -c conda-forge
conda activate abacus_env
export CMAKE_PREFIX_PATH=$CONDA_PREFIX:$CMAKE_PREFIX_PATH

# By default OpenBLAS is used; run `conda install "blas=*=mkl" mkl_fft mkl-devel -c␣
↪conda-forge` to switch implementation.
export MKLROOT=$CONDA_PREFIX # If Intel MKL is required.

export CMAKE_PREFIX_PATH=`python -c 'import torch;print(torch.utils.cmake_prefix_path)
↪'`:$CMAKE_PREFIX_PATH # If DEEPKS support is required;
# usually expands to `$CONDA_PREFIX/lib/python3.1/site-packages/torch/share/cmake`
```

And, follow the instructions in *Build and Install* part above withou manually setting paths to dependencies. See *the advanced installation guide* for more features. Make sure the environment variables are set before running `cmake`. Possible command: `cmake -B build -DENABLE_MLALGO=ON -DENABLE_LIBXC=ON -DENABLE_LIBRI=ON`.

1.6 Install ABACUS manually

1.6.1 Install requirements

Some of these packages can be installed with popular package management system via root permission if you have, 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 pkgconf
```

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.

1.6.2 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 `icpx`(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 `mpiicpx`(for Intel toolkits) or `mpiicpc`(for classic Intel Compiler Classic MPI before 2024.0).
- 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: In ABACUS v3.5.1 or earlier, 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` to help the build system find ELPA headers.
  - `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: In ABACUS v3.5.1 or earlier, Libxc built from source with Makefile is NOT supported; please compile Libxc with CMake instead.
  - `LIBRI_DIR`: (Optional) Path to LibRI.
  - `LIBCOMM_DIR`: (Optional) Path to LibComm.
- 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'.
- `ENABLE_LIBRI=OFF`: *Enable LibRI* to suppport variety of functionals. If `LIBRI_DIR` and `LIBCOMM_DIR` is defined, `ENABLE_LIBRI` 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_GOOGLEBENCH=OFF`: *Build performance tests*
- `ENABLE_MPI=ON`: Enable MPI parallel compilation. If set to `OFF`, a serial version of ABACUS will be compiled. It now supports both PW and LCAO.
- `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.
- `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=mpiicpx cmake -B build -DCMAKE_INSTALL_PREFIX=~/abacus -DELPA_DIR=~/elpa-2025.01.
↪001/build -DCEREAL_INCLUDE_DIR=~/cereal/include
```

1.6.3 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.

1.7 Run ABACUS

1.7.1 Load ABACUS

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. *(Note: `my-install-dir` should be changed to the location of your installed abacus:`/home/your-path/abacus/bin/`.)*

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

If ABACUS is installed by toolchain, there will be an environment script in the toolchain directory named as *abacus_env.sh*. You can source it to set the environment variables.

```
source /path/to/abacus/toolchain/abacus_env.sh
```

If ABACUS is installed by conda, please make sure the conda environment is activated before running ABACUS.

```
conda activate abacus_env
```

1.7.2 Run with Parallelism Setting

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. To use 4 threads and 4 MPI processes, set the environment variable `OMP_NUM_THREADS` before running `mpirun`:

```
OMP_NUM_THREADS=4 mpirun -n 4 abacus
```

In this case, the total thread count is 16.

> Notice: If the MPI library you are using is OpenMPI, which is commonly the case, when you set the number of processes to 1 or 2, OpenMPI will default to `--bind-to core`. This means that no matter how many threads you set, these threads will be restricted to run on 1 or 2 CPU cores. Therefore, setting a higher number of OpenMP threads might result in slower program execution. Hence, when using `mpirun -n` set to 1 or 2, it is recommended to set `--bind-to none` to avoid performance degradation. For example:`OMP_NUM_THREADS=6 mpirun --bind-to none -n 1 abacus`. The detailed binding strategy of OpenMPI can be referred to at https://docs.open-mpi.org/en/v5.0.x/man-openmpi/man1/mpirun.1.html#quick-summary.

ABACUS will try to determine the number of threads used by each process if `OMP_NUM_THREADS` is not set. However, it is **required** to set `OMP_NUM_THREADS` before running `mpirun` to avoid potential performance issues.

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.

1.8 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

1.9 Command line options

Users can check the version of ABACUS by running the command `abacus --version`, the result will be like:

```
ABACUS version v3.9.0.2
```

Users may check the correctness of the setting of parameters in the `INPUT` file by running the command `abacus --check-input`, the result will be like:

```
                            ABACUS v3.9.0.2

               Atomic-orbital Based Ab-initio Computation at UStc

                     Website: http://abacus.ustc.edu.cn/
               Documentation: https://abacus.deepmodeling.com/
                  Repository: https://github.com/abacusmodeling/abacus-develop
                              https://github.com/deepmodeling/abacus-develop
                      Commit: unknown

 Tue Jun 18 14:20:31 2024
 MAKE THE DIR         : OUT.ABACUS/
---------------------------------------------------------
  INPUT parameters have been successfully checked!
---------------------------------------------------------
```

Warnings will be given if there are any errors in the `INPUT` file.

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. 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
```

(continues on next page)

(continued from previous page)

```
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
```

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
pseudo_dir              ./
orbital_dir                ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-6                    # SCF criterion
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` defined in `INPUT` (the default directory is "./"), and the orbital files `Mg_gga_8au_100Ry_4s2p1d.orb` and `O_gga_8au_100Ry_2s2p1d.orb` under the directory of `orbital_dir` also defined in `INPUT` (the default directory is "./"). 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.:

```
OMP_NUM_THREADS=1 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 =
```

(continues on next page)

(continued from previous page)

```
                                    DRANK = 1
                                    DSIZE = 2
                                   DCOLOR = 1
                                    GRANK = 1
                                    GSIZE = 1
 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
pseudo_dir              ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-6                    # SCF criterion
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.
```

(continues on next page)

(continued from previous page)

```
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

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
nelec                   0.0
pseudo_dir              ./
orbital_dir             ./
```

(continues on next page)

(continued from previous page)

```
ecutwfc                 100             # Rydberg
scf_thr                 1e-6              # SCF criterion
basis_type              lcao
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`, 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
nelec                   0.0
pseudo_dir              ./
ecutwfc                 100             # Rydberg
scf_thr                 1e-6              # SCF criterion
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 `STRU_ION_D` with different format.

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. 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  # the output files will be in OUT.{suffix} directory
pseudo_dir              ./   # where the pseudopotential for each element is
orbital_dir             ./   # where the orbital file for each element is
ecutwfc                 100  # in Rydberg
scf_thr                 1e-6 # dimensionless for LCAO, Rydberg for PW. See documents␣
↪for details.
basis_type              lcao # lcao or pw
calculation             scf  # this is the key parameter telling abacus to do a scf␣
↪calculation
out_chg                 0    # only output binary charge file for restart
```

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 (> 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`, and output files will be in OUT.{suffix} directory.
- `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 for PW, dimensionless for LCAO), we recommend `1e-7` for LCAO and `1e-9` for PW basis.
- `basis_type` : the type of basis set for expanding the electronic wave functions, one can set lcao or pw.
- `calculation` : the type of calculation to be performed by ABACUS
- `out_chg` : setting for output the charge density in real space grid, -1 for no output, 0 for binary output, 1 for binary and cube output.

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. Specifically for the `out_chg`, `-1` option is also available, which means turn off the checkpoint of charge density in binary (always dumped in `OUT.{suffix}`, whose name ends with `CHARGE-DENSITY.restart`). Some parameters controlling the output also support a second option to control the output precision, e.g., `out_chg 1 8` will output the charge density on realspace grid with 8 digits after the decimal point.

3.2 Getting Help with Parameters

ABACUS includes a built-in help system to look up INPUT parameters directly from the command line.

3.2.1 Basic Commands

```
# Show general help and common parameters
abacus -h

# Search for parameters by keyword
abacus -s ecut
# Finds: ecutwfc, ecutrho, lcao_ecut, etc.

# Get detailed help for a specific parameter
abacus -h ecutwfc
```

3.2.2 Example Output

```
$ abacus -h ecutwfc

Parameter: ecutwfc
Type:      Real
Default:   50 for PW basis, 100 for LCAO basis
Category:  Plane wave related variables
Unit:      Ry

Description:
  Energy cutoff for plane wave functions...
```

3.2.3 Fuzzy Matching

If you mistype a parameter name, ABACUS suggests similar parameters:

```
$ abacus -h exx_hybrid_steps
Error: Unknown parameter 'exx_hybrid_steps'

Did you mean one of these?
  - exx_hybrid_step

Use 'abacus -s <keyword>' to search for parameters.
```

Parameter lookups are case-insensitive, so `ECUTWFC`, `Ecutwfc`, and `ecutwfc` all work.

3.3 *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
1.889726126 # 1.0 Ang = 1/a_0 = 1/0.529177210544
# Bohr radius ref: https://physics.nist.gov/cgi-bin/cuu/Value?bohrrada0

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
```

(continues on next page)

(continued from previous page)

```
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
```

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`.

Important: When specifying atomic positions, you can use various coordinate systems (Direct, Cartesian, Cartesian_angstrom, etc.) and add optional properties like magnetization, velocity, and movement constraints. See the *detailed STRU documentation* for all available options and best practices.

3.4 *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

BRIEF INTRODUCTION OF THE OUTPUT FILES

The following files are the central output files for ABACUS. After executing the program, you can obtain a log file containing the screen output, more detailed outputs are stored in the working directory `OUT.suffix` (Default one is `OUT.ABACUS`). Here we give some simple descriptions.

4.1 *INPUT*

Different from `INPUT` given by the users, `OUT.suffix/INPUT` contains all parameters in ABACUS.

> **Note:** `OUT.suffix/INPUT` contains the **actual parameters used in the calculation**, including:
>
> 1. **User-specified parameters** (explicitly defined in your input file or command-line arguments, overriding default parameters).
> 2. **System default parameters** (automatically applied when not explicitly provided by the user).

This file ensures calculations can be fully reproduced, even if default values change in future ABACUS versions. Also notice that in rare cases, a small number of parameters may be dynamically reset to appropriate values during runtime.

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

4.2 *running_scf.log*

`running_scf.log` contains information on nearly all function calls made during the execution of ABACUS.

4.3 *KPT.info*

This file contains the information of all generated k-points, as well as the list of k-points actually used for calculations after considering symmetry.

4.4 *eig.txt*

This file includes the energy levels and occupations computed for all k-points. Note: In 3.10-LTS version, the file is named 'istate.info'

4.5 *STRU.cif*

ABACUS generates a `.cif` format structure file based on the input file `STRU`, facilitating users to visualize with commonly used software.

4.6 *warning.log*

The file contains all the warning messages generated during the ABACUS run.

CHAPTER

FIVE

ADVANCED INSTALLATION OPTIONS

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

5.1 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
```

5.2 Build with ML-ALGO

If machine learning algorithms, including DeePKS and machine learning based kinetic energy density functional (ML-KEDF) for OFDFT, is required for DFT calculation, 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 or GPU
- Libnpy

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

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

5.3 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 (optional)
- LibTorch (optional)

In the simplest case, the `tensorflow_cc` and `torch` libraries are in the same directory as the `deepmd_c`/`deepmd_cc` libraries, then

```
cmake -B build -DDeePMD_DIR=/dir_to_deepmd-kit
```

DeePMD-kit supports TensorFlow backend but its libraries are placed at another directory, then

```
cmake -B build -DDeePMD_DIR=/dir_to_deepmd-kit -DTensorFlow_DIR=/dir_to_tensorflow
```

Similarly, DeePMD-kit supports PyTorch backend but its libraries are placed at another directory, then

```
cmake -B build -DDeePMD_DIR=/dir_to_deepmd-kit -DTorch_DIR=/dir_to_pytorch
```

5.4 Build with NEP

This interface enables running MD simulations with the NEP model. It requires the NEP_CPU library, which can be easily installed using toolchain as shown below:

```
./install_abacus_toolchain_new.sh --with-nep=install
```

To build ABACUS:

```
cmake -B build -DNEP_DIR=/path/to/nep_cpu
```

5.5 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, provide `-DLIBRI_DIR=${path to your LibRI folder} -DLIBCOMM_DIR=${path to your LibComm folder}`.

5.6 Build with DFT-D4 support

ABACUS can use the external DFT-D4 library for Grimme's DFT-D4 dispersion correction. DFT-D4 support is optional and disabled by default.

DFT-D4 must be built and installed with CMake, so that ABACUS can locate it through the exported `dftd4-config.cmake` package. Meson-built installations, including the Conda package of `dftd4`, are not supported unless they provide a compatible CMake package file.

To build ABACUS with DFT-D4 support, pass `-DENABLE_DFTD4=ON` to CMake and provide `CMAKE_PREFIX_PATH` environment variable.

In the input file, enable DFT-D4 with:

```
vdw_method d4
vdw_d4_xc pbe
vdw_d4_model d4    # or d4s for the smooth D4S model
```

If `vdw_d4_xc` is set to `default`, ABACUS will infer the functional name from `dft_functional` or pseudopotential metadata and pass it to the DFT-D4 library. The `vdw_d4_model` keyword selects the dispersion model inside the DFT-D4 library; the default is `d4`, while `d4s` enables the smooth D4S model.

5.7 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.

5.8 Build Performance Tests

To build performance tests for ABACUS, define `ENABLE_GOOGLEBENCH` flag. You can also specify the path to a local installation of Google Benchmark by setting `BENCHMARK_DIR` flags. If not found locally, the configuration process will try to download it automatically.

```
cmake -B build -DENABLE_GOOGLEBENCH=1
```

Google Benchmark requires Google Test to build and run the tests. When setting `ENABLE_GOOGLEBENCH` to ON, `BUILD_TESTING` is automatically enabled. After building and installing, performance tests can be executed with `ctest`.

5.9 Build with CUDA support

5.9.1 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 `CMAKE_CUDA_COMPILER` flags.

```
cmake -B build -DUSE_CUDA=1 -DCMAKE_CUDA_COMPILER=${path to cuda toolkit}/bin/nvcc
```

If you are confident that your MPI supports CUDA Aware, you can add `-DUSE_CUDA_MPI=ON`. In this case, the program will directly communicate data with the CUDA hardware, rather than transferring it to the CPU first before communication. But note that if CUDA Aware is not supported, adding `-DUSE_CUDA_MPI=ON` will cause the program to throw an error.

5.10 Build math library from source

> Note: We recommend using the latest available compiler sets, since they offer faster implementations of math functions.

This flag is disabled by default. To build math functions from source code, define `USE_ABACUS_LIBM` flag. It is expected to get a better performance on legacy versions of `gcc` and `clang`.

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

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

5.11 Build with PEXSI support

ABACUS supports the PEXSI library for gamma only LCAO calculations. PEXSI version 2.0.0 is tested to work with ABACUS, please always use the latest version of PEXSI.

To build ABACUS with PEXSI support, you need to compile PEXSI (and its dependencies) first. Please refer to the PEXSI Installation Guide for more details. Note that PEXSI requires ParMETIS and SuperLU_DIST.

After compiling PEXSI, you can set `ENABLE_PEXSI` to `ON`. If the libraries are not installed in standard paths, you can set `PEXSI_DIR`, `ParMETIS_DIR` and `SuperLU_DIST_DIR` to the corresponding directories.

```
cmake -B build -DENABLE_PEXSI=ON -DPEXSI_DIR=${path to PEXSI installation directory} -
↪DParMETIS_DIR=${path to ParMETIS installation directory} -DSuperLU_DIST_DIR=${path␣
↪to SuperLU_DIST installation directory}
```

5.12 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
OPENMP = OFF
# the default is not to use OPENMP to accelerate.
# Change OPENMP to ON to use OPENMP.

#=======================================================================
```

(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 MLALGO: set ENABLE_MLALGO = ON, and 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 LIBTORCH_DIR and TensorFlow_DIR
## To use LibRI:  set LIBRI_DIR and LIBCOMM_DIR
## To use PEXSI: set PEXSI_DIR DSUPERLU_DIR and PARMETIS_DIR
##-----------------------------------------------------------------------

# LIBTORCH_DIR  = /usr/local
# LIBNPY_DIR    = /usr/local
ENABLE_MLALGO   = OFF

# LIBXC_DIR                    = /public/soft/libxc

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

# LIBRI_DIR     = /public/software/LibRI
```

(continues on next page)

(continued from previous page)

```
# LIBCOMM_DIR   = /public/software/LibComm

# PEXSI_DIR = /public/software/pexsi
# DSUPERLU_DIR = /public/software/superlu_dist
# PARMETIS_DIR    = /public/software/parmetis

##----------------------------------------------------------------------
# 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`.

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/`.

5.12.1 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.

5.12.2 Add ML-ALGO Support

To compile ABACUS with machine learning algorithms, you need to set `ENABLE_MLALGO = ON`, and define `LIBTORCH_DIR` and `LIBNPY_DIR` in the file `Makefile.vars` or use

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

directly.

5.12.3 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` (TensorFlow Backend, optional) and/or `LIBTORCH_DIR` (PyTorch Backend, optional) in the file `Makefile.vars`.

Or the `tensorflow_cc` and `torch` libraries are in the same directory as the `deepmd_c`/`deepmd_cc` libraries, then

```
make DeePMD_DIR=/dir_to_deepmd-kit
```

DeePMD-kit supports TensorFlow backend but its libraries are placed at another directory, then

```
make DeePMD_DIR=/dir_to_deepmd-kit TensorFlow_DIR=/dir_to_tensorflow
```

Similarly, DeePMD-kit supports PyTorch backend but its libraries are placed at another directory, then

```
make DeePMD_DIR=/dir_to_deepmd-kit Torch_DIR=/dir_to_pytorch
```

5.12.4 Add LibRI 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

SIX

RUNNING SCF

6.1 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.

6.1.1 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`*.
- `auto`: Abacus first attempts to read the density from a file; if not found, it defaults to using atomic density.
- `dm` (LCAO only): initial charge density from density matrix files in CSR format. For `nspin=1`, reads `dmrs1_nao.csr`. For `nspin=2` (spin-polarized), reads both `dmrs1_nao.csr` (spin-up) and `dmrs2_nao.csr` (spin-down). These files are generated by previous calculations with *`out_dmr 1`*. This method is particularly useful for restarting spin-polarized calculations.
- `hr` (LCAO only): initial charge density from Hamiltonian matrix files in CSR format. The Hamiltonian is read from file, then diagonalized to obtain wavefunctions and charge density. For `nspin=1`, reads `hrs1_nao.csr`. For `nspin=2` (spin-polarized), reads both `hrs1_nao.csr` (spin-up) and `hrs2_nao.csr` (spin-down). These files are generated by previous calculations with *`out_mat_hs2 1`*.

6.1.2 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`*.

6.2 Constructing the Hamiltonian

6.2.1 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 N^4, with N being the number of atoms.

   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.

6.2.2 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.

6.3 Solving the Hamiltonian

6.3.1 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`, `bpcg` 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)

6.3.2 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:

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

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.

6.4 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.

6.4.1 Charge Mixing

In ABACUS, KS-DFT is solved by self-consistent field (SCF) iteration method. By mixing the electron density with that obtained from previous steps, numerical instabilities can be ameliorated while also accelerating convergence. 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_ndim`, `mixing_type`, `mixing_beta`, `mixing_gg0`, `mixing_beta_mag`, `mixing_gg0_mag`, `mixing_gg0_min`, `mixing_angle`.

`mixing_ndim` is the mixing dimensions in DIIS (broyden or pulay) mixing. Gerenally, a larger `mixing_ndim` leads to a better convergence. the default choice `mixing_ndim=8` should work fine in most cases. For `mixing_type`, the default choice is `broyden`, which is slightly better than `Pulay` typically. Besides that, 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.

For most isolated systems, Kerker preconditioning is unnecessary. You can turn off it by setting `mixing_gg0 0.0` to get a faster convergence.

For non-spin-polarized calculations, the default choices usually achieve convergence. If convergence issue arises in metallic systems, you can try different value of Kerker preconditioning *mixing_gg0* and *mixing_gg0_min*, and try to reduce `mixing_beta`, which is 0.8 defaultly for `nspin=1`.

For magnetic calculations, `mixing_beta_mag` and `mixing_gg0_mag` are activated. Considering collinear calculations, you can rely on the default value for most cases. If convergence issue arises, you can try to reduce `mixing_beta` and `mixing_beta_mag` together. For non-collinear calculations, tradtional broyden usually works, especially for a given magnetic configuration. If one is not interested in the energies of a given magnetic configuration but wants to determine the ground state by relaxing the magnetic moments' directions, the standard Broyden mixing algorithm sometimes fails to find the correct magnetic configuration. If so, we can set *mixing_angle=1.0*, which is a promising mixing method proposed by J. Phys. Soc. Jpn. 82 (2013) 114706.

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

As for DFT+U calculations, where the hamiltonian is not only dependent on charge density, but also dependent on density matrix. You can try `mixing_restart>0` and `mixing_dmr=1` to improve convergence. For case extremely hard to converge, you can use so-called U-Ramping method by setting a finite positive `uramping` with `mixing_restart>0` and `mixing_dmr=1`.

6.4.2 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.

6.5 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.

6.5.1 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.

6.5.2 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.

6.5.3 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.

6.5.4 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.

6.6 SCF in Complex Environments

6.6.1 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.

6.6.2 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.

6.6.3 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.

6.6.4 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.

6.6.5 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.

6.7 Spin-polarization and SOC

6.7.1 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.

6.7.2 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"**.

6.7.3 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:

non-colin	lspin	nspir	Effect	When to Use
0	0	<4	No non-collinear magnetism, no SOC	Standard collinear spin-polarized or non-spin-polarized calculations
0	0	4	Same as above, but larger calculation	**Not recommended** - wastes computational resources
1	0	4	Non-collinear magnetism WITHOUT SOC	Systems with complex magnetic structures (e.g., spin spirals, frustrated magnets) where SOC is negligible
0	1	4	SOC WITH z-axis magnetism only	Non-magnetic materials with SOC (e.g., semiconductors with band splitting), or magnetic materials where magnetism is along z-axis
1	1	4	Both SOC AND non-collinear magnetism	Heavy-element magnetic materials where both SOC and non-collinear magnetism are important (e.g., magnetic anisotropy, Dzyaloshinskii-Moriya interaction)

Special case: `noncolin=0, lspinorb=1` is commonly used for non-magnetic materials with SOC effects (e.g., topological insulators, semiconductors with spin-orbit splitting). In this case, the magnetization is NOT automatically set, implying no magnetic moments in the system.

6.7.4 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.

6.8 SOC Effects

6.8.1 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 be calculated with SOC effects (supported since ABACUS v3.9.0).

6.8.2 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.

6.8.3 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.

6.8.4 Pseudopotential Requirements for SOC

When performing SOC calculations (`lspinorb=1`), specific pseudopotential requirements must be met:

Checking Pseudopotential Files

In the UPF (Unified Pseudopotential Format) file header (`PP_HEADER` section), look for:

- `has_so="T"` or `has_so="1"`: Indicates SOC information is included
- `relativistic="full"`: Indicates full-relativistic pseudopotential

Example from a full-relativistic UPF file:

```
<PP_HEADER
   ...
   relativistic="full"
   has_so="T"
   ...
/>
```

Pseudopotential Usage Rules

1. **For SOC calculations** (`lspinorb=1`):
   - **MUST** use full-relativistic pseudopotentials with `has_so=true`
   - Code will terminate with error: "no soc upf used for lspinorb calculation" if scalar-relativistic PP is used
2. **For non-SOC calculations** (`lspinorb=0`):
   - Can use either scalar-relativistic or full-relativistic pseudopotentials
   - If full-relativistic PP is used, ABACUS automatically transforms it to scalar-relativistic version
3. **For ultrasoft pseudopotentials (USPP)**:
   - Full-relativistic USPP **requires** `lspinorb=true`
   - Code will show warning: "FR-USPP please use lspinorb=.true." if this requirement is not met

Where to Find SOC Pseudopotentials

- **SG15_ONCV**: Full-relativistic versions available at quantum-simulation.org
- **PseudoDOJO**: Provides both scalar and full-relativistic versions
- **ABACUS official**: abacus.ustc.edu.cn

6.8.5 Automatic Parameter Settings

When using SOC or non-collinear calculations, ABACUS automatically adjusts several parameters:

When `lspinorb=true`:

1. **nspin**: Automatically set to 4 (noncollinear spin representation)
2. **Symmetry**: Automatically disabled (`symm_flag=-1`) because SOC breaks inversion symmetry
3. **Magnetization**: NOT automatically set when `noncolin=0` (implies non-magnetic material with SOC)

When `noncolin=true`:

1. **nspin**: Automatically set to 4
2. **npol**: Set to 2 (wave function has two spinor components)
3. **Magnetization**: Automatically set if user provides zero values (unless `lspinorb=1` and `noncolin=0`)

Important Notes:

- You do NOT need to manually set `nspin=4` when using `lspinorb=1` or `noncolin=1`
- Symmetry operations are incompatible with SOC, so they are automatically turned off
- For `lspinorb=1, noncolin=0`: This is a special case for non-magnetic materials with SOC, where magnetization is not initialized

6.8.6 Common Errors and Solutions

Error: "no soc upf used for lspinorb calculation"

Cause: Using scalar-relativistic pseudopotentials with `lspinorb=1`

Solution: Download and use full-relativistic pseudopotentials with `has_so=true`. Check the UPF file header to verify `relativistic="full"` and `has_so="T"`.

Error: "nspin=4(soc or noncollinear-spin) does not support gamma only calculation"

Cause: Trying to use `gamma_only=true` with `lspinorb=1` or `noncolin=1`

Solution: Set `gamma_only=false` or `gamma_only=0` in your INPUT file. SOC and non-collinear calculations require k-point sampling beyond the gamma point.

Warning: "FR-USPP please use lspinorb=.true."

Cause: Using full-relativistic ultrasoft pseudopotentials without enabling SOC

Solution: Set `lspinorb=true` in your INPUT file, or switch to scalar-relativistic USPP if SOC is not needed.

Issue: Forces or stresses not calculated

Note: This issue has been resolved. Atomic forces and cell stresses can now be calculated with SOC effects (supported since ABACUS v3.9.0).

If you are using an older version of ABACUS (before v3.9.0), force and stress calculations with SOC were not supported. Please upgrade to the latest version to use this feature.

6.8.7 INPUT File Examples

Example 1: SOC without Non-collinear Magnetism

For non-magnetic materials with SOC (e.g., GaAs, topological insulators):

```
INPUT_PARAMETERS
calculation         scf
basis_type          pw
ecutwfc             50
lspinorb            1         # Enable SOC
noncolin            0         # No non-collinear magnetism
# nspin will be automatically set to 4
# symmetry will be automatically disabled
```

Example 2: Non-collinear Magnetism without SOC

For systems with complex magnetic structures but negligible SOC:

```
INPUT_PARAMETERS
calculation         scf
basis_type          lcao
lspinorb            0         # No SOC
noncolin            1         # Enable non-collinear magnetism
# nspin will be automatically set to 4
# Magnetization directions should be specified in STRU file
```

Example 3: Both SOC and Non-collinear Magnetism

For heavy-element magnetic materials (e.g., Fe with SOC, materials with DMI):

```
INPUT_PARAMETERS
calculation         scf
basis_type          pw
ecutwfc             60
lspinorb            1         # Enable SOC
noncolin            1         # Enable non-collinear magnetism
# nspin will be automatically set to 4
# symmetry will be automatically disabled
# Magnetization directions should be specified in STRU file
```

Example 4: Partial-relativistic SOC

For fine-tuning SOC strength:

```
INPUT_PARAMETERS
calculation         scf
basis_type          pw
ecutwfc             50
lspinorb            1         # Enable SOC
soc_lambda          0.5       # 50% SOC strength
# Useful when full SOC overestimates or underestimates experimental results
```

CHAPTER

SEVEN

BASIS SET AND PSEUDOPOTENTIALS

7.1 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*.

The angular part of orbitals are real spherical harmonics defined (in terms of conventional spherical harmonics in quantum mechanical literature) as

$$Y_{lm} = \begin{cases} \sqrt{2}\,\mathrm{Im}Y_l^{|m|} & m0 \\ Y_l^0 & m=0 \\ \sqrt{2}\,\mathrm{Re}Y_l^{|m|} & m0 \end{cases}$$

Note that real spherical harmonics adopted by ABACUS differ from some other definition, e.g. Table of spherical harmonics - Wikipedia, by a factor of $(-1)^m$.

Inside ABACUS, orbitals in LCAO basis are arranged lexicographically by species-position-l-zeta-m **except for the intra-m ordering**. Specifically, orbitals are first ordered by their atomic species in accordance with the `ATOMIC_SPECIES` section of the STRU file. For orbitals of the same species, orbitals belonging to each atom are put together, with their overall order following the `ATOMIC_POSITIONS` section of the STRU file. Orbitals on each atom are further ascendingly ordered by their angular momentum (s,p,d,f,…), followed by an order based on their their zeta number. Finally, m is ordered as 0, 1, -1, 2, 2, . . ., l, -l, which is the only exception of the lexicographic order.

7.2 Generating atomic orbital bases

Users may also generate ABACUS numerical atomic obitals based on their own flavor. The theoretical background of orbital generation can be found in following works:

- Spillage: Chen M, Guo G C, He L. Systematically improvable optimized atomic basis sets for ab initio calculations[J]. Journal of Physics: Condensed Matter, 2010, 22(44): 445501.
- PTG DPSI: Lin P, Ren X, He L. Strategy for constructing compact numerical atomic orbital basis sets by incorporating the gradients of reference wavefunctions[J]. Physical Review B, 2021, 103(23): 235131.

Guidelines for generating atomic orbital bases are as follows:

- Numerical Atomic Orbitals 1: the nomenclature and usage of numerical atomic orbitals in ABACUS (Chinese)

- Numerical Atomic Orbitals 2: generate numerical atomic orbitals based on given norm-conserving pseudopotential (Chinese)
- Numerical Atomic Orbitals 3: generate high-precision numerical atomic orbitals (Chinese)

Stable orbital generation programs can be found in guidelines above, there is also another developing version of orbital generation program, in which algorithms are consecutively improved: Github repository of ABACUS ORBGEN project, the usage of which can be found in README (in English) file.

NOTE: users are encouraged to cite the above works when numerical atomic orbitals and its generation codes are used in their research.

7.3 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:

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

7.4 Pseudopotentials

7.4.1 Supported formats

ABACUS supports both norm-conserving and ultrasoft pseudopotentials. For norm-conserving pseudopotentials, UPF, UPF2, VWR, and BLPS formats are supported. For ultrasoft pseudopotentials, UPF and UPF2 formats are supported.

7.4.2 Pseudopotentials for SOC Calculations

When performing spin-orbit coupling (SOC) calculations with `lspinorb=1`, specific pseudopotential requirements must be met:

Identifying SOC Pseudopotentials

Full-relativistic pseudopotentials suitable for SOC calculations can be identified by checking the UPF file header (`PP_HEADER` section):

```
<PP_HEADER
   ...
   relativistic="full"
   has_so="T"
   ...
/>
```

- **`relativistic="full"`**: Indicates a full-relativistic pseudopotential
- **`has_so="T"` or `has_so="1"`**: Indicates SOC information is included in the pseudopotential

Usage Rules

1. **SOC calculations** (`lspinorb=1`):
   - **Required**: Full-relativistic pseudopotentials with `has_so=true`
   - **Error if not met**: "no soc upf used for lspinorb calculation"
2. **Non-SOC calculations** (`lspinorb=0`):
   - **Flexible**: Can use either scalar-relativistic (`relativistic="scalar"`) or full-relativistic pseudopotentials
   - **Automatic conversion**: If full-relativistic PP is used, ABACUS automatically transforms it to scalar-relativistic version
3. **Ultrasoft pseudopotentials (USPP)**:
   - **Constraint**: Full-relativistic USPP must be used with `lspinorb=true`
   - **Warning if violated**: "FR-USPP please use lspinorb=.true."

Validation by ABACUS

ABACUS performs automatic validation when reading pseudopotentials:

- Checks if `lspinorb=1` but pseudopotential has `has_so=false` → terminates with error
- Checks if full-relativistic USPP is used without `lspinorb=1` → shows warning
- Automatically averages SOC-related beta functions when `lspinorb=0`

Where to Find SOC Pseudopotentials

For SOC calculations, download full-relativistic pseudopotentials from:

- **SG15_ONCV**: quantum-simulation.org - widely used in ABACUS
- **PseudoDOJO**: pseudo-dojo.org - provides both scalar and full-relativistic versions
- **ABACUS official**: abacus.ustc.edu.cn - includes both pseudopotentials and numerical atomic orbitals

For more details on SOC calculations, see *Spin-polarization and SOC*.

7.4.3 Usage

For more information about pseudopotential usage, check the `ATOMIC_SPECIES` section in the specification of the *STRU file*.

7.4.4 Download

Users can find pseudopotentials in the following links:

Website

- Quantum ESPRESSO: the official website of Quantum ESPRESSO, where you can find a large number of pseudopotential files.
- Stantard Solid State Pseudopotential library: a library of **high-quality** pseudopotentials for solid-state calculations, with **a large number of tests on efficiency and precison**.
- PWmat: a website that provides a large number of pseudopotential files, various kinds of semi-core constructed pseudopotentials are included. **Several sets (with or without f-electrons/noncolinear core correction) of Lanthanide pseudopotentials are also available**.
- THEOS: PSlibrary 0.3.1, a library of pseudopotentials for DFT calculations, including ultrasoft, norm-conserving both full-relativistic and scalar-relativistic pseudopotentials.
- ABACUS@USTC: **ABACUS official website** where you can find a large number of pseudopotential files and numerical atomic orbital files.
- BLPS: BLPS format pseudopotential library

Norm-conserving pseudopotentials

- SG15: **vastly used in ABACUS** DFT calculation and numerical atomic orbital generation.
- PseudoDOJO: another widely used pseudopotential database, developed by Abinit group, **including Lanthanide pseudopotentials (f-electrons frozen)**.
- The Rappe group: a collection of GGA pseudopotentials which are generated with Opium code, several tests proves that are out-performing in alloy systems.
- Matteo Giantomassi's Github repo: a Github repository that contains norm-conserving pseudopotentials for **Actinides and superheavy elements to 120-th element**.

Ultrasoft pseudopotentials

- Vanderbilt: a collection of ultrasoft pseudopotentials generated by Vanderbilt group.
- GBRV by Kevin F. Garrity, Joseph W. Bennett, Karin M. Rabe, and David Vanderbilt: presently the most popular ultrasoft pseudpotentials in Quantum ESPRESSO user community.

7.4.5 Pseudopotential Generation

For pseudopotential generation, please refer to the following links for more information:

- Quantum ESPRESSO
- ONCVPSP
- Opium

A Chinese guideline is also available here: A brief introduction of norm-conserving pseudopotential generation

CHAPTER

EIGHT

ABACUS PSEUDOPOTENTIAL-NUMERICAL ATOMIC ORBITAL SQUARE (APNS) PROJECT

For the purpose of providing high-quality pseudopotentials and numerical atomic orbitals, we have initiated the APNS project. The project is aimed at providing a large number of high-quality pseudopotentials and numerical atomic orbitals, along with diverse test data for the ABACUS user community, reduce the cost of generating and testing pseudopotentials and numerical atomic orbitals by users, and promote the development of ABACUS software. The project is currently in the development stage, and we welcome contributions from the community. For more information, please refer to the following links:

- APNS website: test data and results
- APNS workflow (Github repository): high-throughput test of pseudopotentials and numerical atomic orbitals

There are also other excellent projects that provide high-quality pseudopotentials along with test data:

- Solid State Pseudopotential library
- Verification of the precision of DFT implementation via AiiDA common workflows

CHAPTER

NINE

GEOMETRY OPTIMIZATION

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

ABACUS provides two implementations for variable-cell relaxation, controlled by the *relax_new* parameter:

- **New implementation** (`relax_new = True`, default since v3.8): Uses a simultaneous conjugate gradient (CG) optimization for both ionic positions and cell parameters. Both degrees of freedom are optimized together in each step.
- **Old implementation** (`relax_new = False`): Follows a nested procedure where fixed-cell structural relaxation is performed first, 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. When using the old implementation (`relax_new = False`), 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.

9.1 Optimization Algorithms

ABACUS offers multiple optimization algorithms for structural relaxation, which can be selected using the *relax_method* keyword. The available algorithms and their behavior depend on the *relax_new* setting:

9.1.1 Algorithm Availability

New implementation (`relax_new = True`, default):

- **CG (Conjugate Gradient)**: Simultaneous optimization of both ionic positions and cell parameters using CG with line search. This is the only algorithm available for the new implementation.

Old implementation (`relax_new = False`):

- **CG (Conjugate Gradient)**: For ionic relaxation; CG is also used for cell parameter optimization in the nested procedure
- **BFGS**: Quasi-Newton method for ionic relaxation
- **LBFGS**: Limited-memory BFGS for ionic relaxation
- **SD (Steepest Descent)**: Simple gradient descent for ionic relaxation

- **CG-BFGS**: Mixed method that starts with CG and switches to BFGS when force convergence reaches the threshold set by *relax_cg_thr*

We also provide a *list of keywords* for controlling the relaxation process.

9.1.2 BFGS method

The BFGS method is a quasi-Newton method for solving nonlinear optimization problems. It belongs to the class of quasi-Newton methods 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.

Note: BFGS is only available with the old implementation (`relax_new = False`).

ABACUS provides two BFGS implementations, controlled by the second element of *relax_method*:

- **Default BFGS** (`relax_method = bfgs 2` or `relax_method = bfgs`): Updates the inverse of the approximate Hessian matrix B directly. This is the recommended implementation.
- **Traditional BFGS** (`relax_method = bfgs 1`): Updates the approximate Hessian matrix B itself, then obtains the inverse by solving matrix eigenvalues and taking their reciprocals. Both methods are mathematically equivalent, but in some cases the traditional variant may perform better.

9.1.3 LBFGS method

The L-BFGS (Limited-memory BFGS) method is a memory-efficient variant of BFGS that stores only a few vectors representing the Hessian approximation instead of the full matrix. This makes it particularly suitable for large systems with many atoms.

Note: LBFGS is only available with the old implementation (`relax_new = False`). Set `relax_method = lbfgs` to use this method.

9.1.4 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 descends the fastest.

Note: SD is only available with the old implementation (`relax_new = False`).

In practice, the SD method may take many iterations to converge, and is generally not recommended for production calculations.

9.1.5 CG method

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

ABACUS provides two implementations of the CG method:

- **New CG implementation** (`relax_new = True`, default): Performs simultaneous optimization of both ionic positions and cell parameters using a line search algorithm. This implementation is more efficient for `cell-relax` calculations as it optimizes all degrees of freedom together. The step size can be controlled by *relax_scale_force*.
- **Old CG implementation** (`relax_new = False`): Uses a nested procedure where ionic positions are optimized first using CG, followed by cell parameter optimization (also using CG) in `cell-relax` calculations. This is the traditional approach where the two optimization steps are separated.

9.2 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.

9.2.1 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.

9.2.2 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*.

Available constraints by implementation:

- **New implementation** (`relax_new = True`):
  - `fixed_axes = "shape"`: Only allows volume changes (hydrostatic pressure), cell shape is fixed
  - `fixed_axes = "volume"`: Allows shape changes but keeps volume constant
  - `fixed_axes = "a"`, `"b"`, `"c"`, etc.: Fix specific lattice vectors or combinations
  - `fixed_ibrav = True`: Maintain the Bravais lattice type during relaxation
- **Old implementation** (`relax_new = False`):
  - **All `fixed_axes` options now supported**: “shape”, “volume”, “a”, “b”, “c”, “ab”, “ac”, “bc”, “abc”
  - **`fixed_ibrav` now supported**: Maintains Bravais lattice type during relaxation
  - Can combine `fixed_axes` with `fixed_ibrav` for constrained relaxation
  - **Implementation approach**: Uses post-update constraint enforcement (volume rescaling and lattice reconstruction after each CG step)

VASP ISIF correspondence:

If you are familiar with the `ISIF` option from VASP, here is the 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-relax", fixed_axes = "shape", fixed_atoms = True

9.2.3 Stop Geometry Optimization Manually

It is usually difficult to converge when calculating large systems, but people do not want to give up this calculation result. Providing a file named `EXIT`:

```
stop_ion    true
```

ABACUS will end normally and produce a complete file.

CHAPTER

TEN

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:

- fire: a MD-based relaxation algorithm, see *details* here
- nve: NVE ensemble with velocity Verlet algorithm
- nvt: NVT ensemble
- npt: Nose-Hoover style NPT ensemble
- langevin: NVT ensemble with Langevin thermostat
- msst: MSST method

When *md_type* is set to nvt, *md_thermostat* is used to specify the temperature control method used in NVT ensemble.

- nhc: Nose-Hoover chain
- anderson: Anderson thermostat
- berendsen: Berendsen thermostat
- rescaling: velocity Rescaling method 1
- rescale_v: velocity Rescaling method 2

When *md_type* is set to npt, *md_pmode* is used to specify the cell fluctuation mode in NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.

- iso: isotropic cell fluctuations
- aniso: anisotropic cell fluctuations
- tri: non-orthogonal (triclinic) simulation box

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

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.

10.1 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.

10.2 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.

10.3 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.

10.4 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.

10.5 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.

10.6 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.

10.7 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).

10.8 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.

10.9 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.

10.10 DPMD

Compiling ABACUS with DeePMD-kit, MD calculations based on machine learning DP model is enabled.

To employ DPMD calculations, *esolver_type* should be set to `dp`. And the filename of DP model is specified by keyword *pot_file*.

10.11 NEP

If ABACUS is compiled with the Neuroevolution Potential (NEP), MD simulations using NEP models are enabled. To use this feature, set *esolver_type* to `nep` and specify the potential file path with the *pot_file* keyword in your INPUT file.

CHAPTER

ELEVEN

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.

11.1 CUDA GPU Implementations

In ABACUS, we provide the option to use GPU devices to accelerate performance. The implementation of GPU acceleration differs between PW basis and LCAO basis. Specifically, under PW basis, it has the following 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.
- **Accelerated 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.

Unlike PW basis, only the grid integration module (module_gint) and the diagonalization of the Hamiltonian matrix (source_hsolver) have been implemented with GPU acceleration under LCAO basis.

11.1.1 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)

11.1.2 Building ABACUS with the GPU support:

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

Setting both USE_ELPA and USE_CUDA to ON does not automatically enable ELPA to run on GPUs. ELPA support for GPUs needs to be enabled when ELPA is compiled. enable GPU support.

The ABACUS program will automatically determine whether the current ELPA supports GPU based on the elpa/elpa_configured_options.h header file. Users can also check this header file to determine the GPU support of ELPA in their environment. ELPA introduced a new API elpa_setup_gpu in version 2023.11.001. So if you want to enable ELPA GPU in ABACUS, the ELPA version must be greater than or equal to 2023.11.001.

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

In `INPUT` file we need to set the input parameter *device* to `gpu`. If this parameter is not set, ABACUS will try to determine if there are available GPUs.

- Set `ks_solver`: For the PW basis, CG, BPCG and Davidson methods are supported on GPU; set the input parameter *ks_solver* to `cg`, `bpcg` or `dav`. For the LCAO basis, `cusolver`, `cusolvermp` and `elpa` is supported on GPU.
- **single-card**: ABACUS allows for single-GPU acceleration. You can run ABACUS without any MPI process by command `abacus`, and `ks_solver cusolver` is recommended for the LCAO basis. *note: avoid using `mpirun -n 1 abacus`.*
- **multi-cards**: ABACUS allows for multi-GPU acceleration. If you have multiple GPU cards, you can run ABACUS with several MPI processes, and each process will utilize one GPU card. For example, the command `mpirun -n 2 abacus` will by default launch two GPUs for computation. If you only have one card, this command will only start one GPU. *note: the number of MPI processes SHOULD be equal to the number of GPU cards, unless you are using MPS in your computer.*

11.1.4 Examples

We provides examples of gpu calculations.

11.1.5 Known limitations

PW basis:

- Only k point parallelization is supported, so the input keyword `kpar` will be set to match the number of MPI tasks automatically.
- By default, CUDA architectures are automatically selected based on the CUDA Toolkit version. For CUDA versions before 13.0, architectures 60, 70, 75, 80, 86, 89, and 90 are compiled (if supported by the CUDA version). For CUDA 13.0 and later, only architectures 75 and above are compiled, as CUDA 13 dropped support for older architectures. This can be overridden using the CMake variable `CMAKE_CUDA_ARCHITECTURES` or the environmental variable `CUDAARCHS`.

LCAO basis:

- Unless there is a specific reason, avoid using multiple GPUs, as it can be slower than using a single GPU. This is because the generalized eigenvalue solution of the LCAO basis set will incur additional communication overhead when calculated on multiple cards. When the memory limit of a GPU card makes it insufficient to complete the task, it is recommended to use multiple cards for calculation.
- When using elpa on GPUs, some ELPA internal logs will be output.

CHAPTER

TWELVE

ELECTRONIC PROPERTIES AND OUTPUTS

12.1 Extracting Band Structure

In ABACUS, in order to obtain the eigenvalues of Hamiltonian, or generally called band structure, examples can be found in examples/band. Similar to the DOS case, one first needs to perform a ground-state energy calculation ***with one additional keyword "out_chg" in the INPUT file***:

```
out_chg 1
```

With this input parameter, the converged charge density will be output in the files such as `chgs1.cube`, `chgs2.cube`, etc. Then, one can use the same `STRU` file, pseudopotential files and atomic orbital files (and the local density matrix file onsite.dm if DFT+U is used) to do a non-self-consistent (NSCF) 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)
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 is a relevant k-point file KPT (in LINE mode):

```
K_POINTS # keyword for start
6 # number of high symmetry lines
Line # line-mode
```

(continues on next page)

(continued from previous page)

```
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
```

This means we are using the following k-points:

- 6 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.

Next, run ABACUS and you will see a file named `eigs1.txt` in the output directory. Plot it and you will obtain the 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)

12.2 Calculating DOS and PDOS

12.2.1 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
```

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
            ...
```

12.2.2 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)`.

12.3 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 MD calculations, the output interval is controlled by the keyword *out_freq_ion*. In the file, there are contents like (`nspin 1`):

```
STEP: 0
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
pz                        0                      0.85945806
px                        0                      0.85945806
py                        0                      0.85945806
  sum over m                                      2.5783742
pz                        1                     0.0065801228
px                        1                     0.0065801228
py                        1                     0.0065801228
  sum over m                                     0.019740368
  sum over m+zeta                                 2.5981145
dz^2                        0                      0.0189287
dxz                        0                      0.046491729
dyz                        0                      0.046491729
dx^2-y^2                        0                      0.0189287
dxy                        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`

12.4 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 broyden
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
```

(continues on next page)

(continued from previous page)

```
-3.8400149345 6.6511009684 0.0000000000
0.0000000000 0.0000000000 65.6767997742

ATOMIC_POSITIONS
Cartesian
Si
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
```

(continues on next page)

(continued from previous page)

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

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

- ElecStaticPot.cube: contains electrostatic potential (unit: Rydberg) in realspace. This file can be visually viewed by the software of VESTA.

12.5 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/11_wfc.

12.5.1 Wave Function in G-Space

To output wave functions in G-space, add one of the following keywords to the `INPUT` file while performing SCF calculation:

- **PW basis**: Set `out_wfc_pw` to `1`. Output file format: `wfs[spin]k[kpoint]_pw.txt`, where `[spin]` is the spin channel index, and `[kpoint]` the k-point index.
- **LCAO basis**: Set `out_wfc_lcao` to `1`.
  - **Multi-k calculations**: Generates multiple files `wfs[spin]k[kpoint]_nao.txt`.
  - **Gamma-only calculations**: `wfs[spin]_nao.txt` instead.

12.5.2 Wave Function in Real Space

One can also choose to output real-space wave functions with the keyword `out_wfc_norm` or `out_wfc_re_im`.

Notice: When the `basis_type` is `lcao`, only `get_wf calculation` is effective. An example is examples/11_wfc/lcao_ienvelope_Si2.

12.6 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:

```
STEP: 0  Cubefile created from ABACUS. Inner loop is z, followed by y and x
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
```

(continues on next page)

(continued from previous page)

```
 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
 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 contains the current ion step.
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

12.7 Extracting Hamiltonian and Overlap Matrices

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

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

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

and

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

12.7.1 out_mat_hs

Users can set the keyword *out_mat_hs* to true to print the upper triangular part of the Hamiltonian matrices and overlap matrices for each k point into files in the directory `OUT.${suffix}`. It is available for both gamma_only and multi-k calculations.

The $H(k)$ and $S(k)$ matrices are stored with numerical atomic orbitals as basis, and the corresponding sequence of the numerical atomic orbitals can be seen in *Basis Set*.

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.

12.7.2 out_mat_hs2

The output of $H(R)$ and $S(R)$ 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.

Output Format

The H® and S® matrices are output in standard Compressed Sparse Row (CSR) format, matching the format used by `out_dmr`.

For single-point SCF calculations:

- **nspin = 1 or nspin = 4**: Two files `hrs1_nao.csr` and `srs1_nao.csr` are generated, containing the Hamiltonian matrix $H(R)$ and overlap matrix $S(R)$ respectively.
- **nspin = 2**: Three files `hrs1_nao.csr`, `hrs2_nao.csr`, and `srs1_nao.csr` are created, where the first two files correspond to $H(R)$ for spin up and spin down, respectively.

File Structure

Each file starts with a header:

```
 --- Ionic Step 1 ---
 # print H matrix in real space H(R)
 1 # number of spin directions
 1 # spin index
 100 # number of localized basis
 50 # number of Bravais lattice vector R

[UnitCell information]

#------------------------------------------------------------------------#
#                                CSR Format                              #
...
 0 0 0 5
 # CSR values
 1.234e-01 2.345e-02 ...
 # CSR column indices
 0 5 10 ...
 # CSR row pointers
 0 3 7 ...
```

The CSR format stores a sparse m × n matrix M in row form using three arrays (values, column indices, row pointers). According to Wikipedia:

- The arrays **values** and **column indices** are of length NNZ (number of nonzero entries), and contain the non-zero values and the column indices of those values respectively.
- The array **row pointers** is of length m + 1 and encodes the index where each row starts. The last element is NNZ.

Precision Control

Use `out_mat_hs2 1 12` to output with 12-digit precision (default is 8).

For calculations involving ionic movements, the output frequency of the matrix is controlled by *out_freq_ion* and *out_app_flag*.

12.7.3 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_nao.csr` will be generated in the working directory, which contains the overlap matrix.

> When `nspin` is set to 1 or 2, the dimension of the overlap matrix is nlocal $\times$ nlocal, where nlocal is the total number of numerical atomic orbitals. These numerical atomic orbitals are ordered from outer to inner loop as atom, angular quantum number l, zeta (multiple radial orbitals corresponding to each l), and magnetic quantum number m. When `nspin` is set to 4, the dimension of the overlap matrix is (2 $\times$ nlocal) $\times$ (2 $\times$ nlocal). In this case, the numerical atomic orbitals are ordered from outer to inner loop as atom, angular quantum number l, zeta (multiple radial orbitals corresponding to each l), magnetic quantum number m, and npol (index of spin, ranges from 0 to 1).

12.7.4 examples

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

- 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_hs2_multik: writing H® and S® for multi-k calculation
- out_s_multik: running calculation=get_s to obtain overlap matrix for multi-k calculation

Reference output files are provided in each directory.

12.8 Extracting Density Matrices

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

```
out_dmk    1
```

After finishing the calculation, the density matrix is written into `OUT.${suffix}/`.

For current develop versions:

- gamma-only (`gamma_only = 1`): `dmg1_nao.txt` (`nspin=1/4`) or `dms1g1_nao.txt` and `dms2g1_nao.txt` (`nspin=2`)
- multi-k (`gamma_only = 0`): `dmk1g1_nao.txt`, `dmk2g1_nao.txt`, ... (`nspin=1/4`) or `dmk1s1g1_nao.txt`, `dmk1s2g1_nao.txt`, ... (`nspin=2`)

Here `g{istep}` denotes the geometry/step index in the output filename.

For 3.10-LTS, the corresponding keyword is `out_dm`, and file names follow the legacy style such as `SPIN1_DM` and `SPIN2_DM`.

The file content 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: Version difference summary:
  - develop: `out_dmk` supports both gamma-only and multi-k-point output.
  - 3.10-LTS: use `out_dm`.

12.8.1 Real-space Density Matrix (CSR format)

ABACUS can also output the real-space density matrix DM® in CSR (Compressed Sparse Row) format by setting:

```
out_dmr    1
```

This feature is only valid for multi-k calculations (`gamma_only = 0`).

After the calculation, the density matrix files are written to `OUT.${suffix}/`:

- develop naming pattern: `dmr{s}{spin index}{g}{geometry index}{_nao}.csr`
- `nspin=1`: `dmrs1_nao.csr`
- `nspin=2` (spin-polarized): `dmrs1_nao.csr` (spin-up) and `dmrs2_nao.csr` (spin-down)

For 3.10-LTS, the corresponding keyword is `out_dm1`, and the file names are `data-DMR-sparse_SPIN0.csr` and `data-DMR-sparse_SPIN1.csr`, etc.

These files can be used to restart calculations by setting `init_chg dm` in the INPUT file together with `read_file_dir` pointing to the directory containing the CSR files. This is supported for both `nspin=1` and `nspin=2`.

12.9 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        -1 // 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
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 vectors R1, R2, and R3 refer to the three lattice vectors of the unit cell. When gdir=3, the calculated polarization is along the R3 direction. The three values in parentheses represent the re-projection of the polarization along the R3 direction onto the Cartesian coordinate system (i.e., the xyz coordinate system). To obtain the full polarization components in the Cartesian system, you need to calculate the polarization for R1, R2, and R3 separately, and then sum their respective x, y, and z components.

CHAPTER

THIRTEEN

DETAILED INTRODUCTION OF THE OUTPUT FILES

13.1 ABACUS Output File Specification

13.1.1 1. Background and Motivation

ABACUS is an integrated software package designed to provide a cohesive user experience. To achieve this goal, **we are establishing unified output file standards** that all developers should follow.

Recent versions of ABACUS have been working on standardizing all output file naming conventions. This work is ongoing, and if there are discrepancies between actual file names and this document, please refer to the latest documentation.

All output file naming conventions can be found in the online documentation (with corresponding 3.10-LTS file names): ABACUS Input Documentation

13.1.2 2. File Naming Conventions

2.1 Basic Rules

Rule 1: All ABACUS output files are stored in the `OUT.{suffix}/` directory.

Rule 2: File extensions by category:

Extension	Description
`.txt`	Text file
`.dat`	Binary file
`.csr`	Sparse matrix format
`.cube`	3D spatial data format

Rule 3: For special output quantities (e.g., wavefunctions), add `_pw` or `_nao` to distinguish between plane wave basis and numerical atomic orbital basis.

Rule 4: File names are lowercase. Physical quantities appear at the beginning:

Prefix	Physical Quantity
`chg`	Charge density
`pot`	Potential
`eig`	Eigenvalue / Energy level
`wf`	Wavefunction
`dm`	Density matrix
`h`	Hamiltonian matrix H
`s`	Overlap matrix S
`t`	Kinetic energy operator
`r`	Position operator or Bravais lattice vector R
`k`	k-point in Brillouin zone
`xyz`	Three spatial directions
`ini`	Initial state (before electronic iteration)

Rule 5: Suffixes following the physical quantity:

Suffix	Meaning
`s1`, `s2`, `s3`, `s4`	Spin channel (1, 2 for collinear; 1, 2, 3, 4 for non-collinear with SOC)
`s12`	Non-collinear spin calculation
`k#`	k-point index (e.g., `k1`, `k2`)
`g#`	Ionic step index for relax/md (e.g., `g1`, `g2`)
`ini`	Initial state (before electronic iteration), used before or after `g#`

Important:

- All index numbers start from 1 (not 0)
- For Gamma-only algorithm in LCAO, no `k` index is included
- Overlap matrix `s` does not distinguish spin, so only one matrix is output
- For initial charge density output (`out_chg = 2`):
  - `out_freq_ion = 0`: `chg_ini.cube` (nspin=1) or `chgs{#}_ini.cube` (nspin=2/4) - output at every step (overwrite same file)
  - `out_freq_ion > 0`: `chgg{#}_ini.cube` (nspin=1) or `chgs{#}g{#}_ini.cube` (nspin=2/4) - output every out_freq_ion steps
- For initial potential output (`out_pot = 3`):
  - `out_freq_ion = 0`: `pot_ini.cube` (nspin=1) or `pots{#}_ini.cube` (nspin=2/4) - output at every step (overwrite same file)
  - `out_freq_ion > 0`: `potg{#}_ini.cube` (nspin=1) or `pots{#}g{#}_ini.cube` (nspin=2/4) - output every out_freq_ion steps

2.2 Examples

File Name	Interpretation
`chgs1.cube`	Charge density, spin 1
`chgs2.cube`	Charge density, spin 2
`chgs3.cube`	Charge density, spin 3 (non-collinear with SOC)
`chg_ini.cube`	Initial charge density (out_freq_ion=0, nspin=1)
`chgs1_ini.cube`	Initial charge density (out_freq_ion=0, spin 1, nspin=2/4)
`chgg1_ini.cube`	Initial charge density, geometry step 1 (nspin=1)
`chgs1g1_ini.cube`	Initial charge density, spin 1, geometry step 1 (nspin=2/4)
`pot_ini.cube`	Initial potential (out_freq_ion=0, nspin=1)
`pots1_ini.cube`	Initial potential (out_freq_ion=0, spin 1, nspin=2/4)
`potg1_ini.cube`	Initial potential, geometry step 1 (nspin=1)
`pots1g1_ini.cube`	Initial potential, spin 1, geometry step 1 (nspin=2/4)
`pots1.cube`	Local potential, spin 1
`elftot.cube`	ELF, total (nspin=1/4)
`elfs1.cube`	ELF, spin 1 (nspin=2)
`elfs2.cube`	ELF, spin 2 (nspin=2)
`elftot.cube`	ELF, total (nspin=2)
`elftotg1.cube`	ELF, total, geometry step 1 (nspin=1/4)
`elfs1g1.cube`	ELF, spin 1, geometry step 1 (nspin=2)
`elftotg1.cube`	ELF, total, geometry step 1 (nspin=2)
`eig_occ.txt`	Eigenvalues and occupations
`doss1g1_nao.txt`	DOS, spin 1, geometry step 1, NAO basis
`wf_pw.dat`	Wavefunction, plane wave basis
`sr.csr`	Overlap matrix in real space (no spin index)

2.3 Common Output Files

File Name	Description
`running_scf.log`	SCF iteration log
`dos.txt`	Density of states
`eig_occ.txt`	Eigenvalues and occupations
`mulliken.txt`	Mulliken population analysis
`band.txt`	Band structure
`chgs1.cube`, `chgs2.cube`	Charge density (spin 1, spin 2)
`chg.cube`	Total charge density
Initial charge density (out_chg=2)	
`chg_ini.cube`	Initial charge density (out_freq_ion=0, nspin=1)
`chgs{#}_ini.cube`	Initial charge density (out_freq_ion=0, spin {#}, nspin=2/4)
`chgg{#}_ini.cube`	Initial charge density (out_freq_ion>0, geometry step {#}, nspin=1)
`chgs{#}g{#}_ini.cube`	Initial charge density (out_freq_ion>0, spin {#}, geometry step {#}, nspin=2/4)
Initial potential (out_pot=3)	
`pot_ini.cube`	Initial potential (out_freq_ion=0, nspin=1)
`pots{#}_ini.cube`	Initial potential (out_freq_ion=0, spin {#}, nspin=2/4)
`potg{#}_ini.cube`	Initial potential (out_freq_ion>0, geometry step {#}, nspin=1)
`pots{#}g{#}_ini.cube`	Initial potential (out_freq_ion>0, spin {#}, geometry step {#}, nspin=2/4)
`taus1.cube`, `taus2.cube`	Kinetic energy density (tau)
`pots1.cube`, `pots2.cube`	Local potential
ELF (out_elf)	
`elftot.cube`	ELF, total (nspin=1/4)
`elfs1.cube`, `elfs2.cube`	ELF, spin 1/2 (nspin=2)
`elftot.cube`	ELF, total (nspin=2)
`elftotg{#}.cube`	ELF, total, geometry step {#} (nspin=1/4)
`elfs{#}g{#}.cube`	ELF, spin {#}, geometry step {#} (nspin=2)
`elftotg{#}.cube`	ELF, total, geometry step {#} (nspin=2)

13.1.3 3. File Format Standards

3.1 Header Section with Comments

Every output file should include # comment lines to explain:

- Data meaning
- Units
- Source module
- Key parameters

```
# <description of file content>
# Module: <source module name>
# Units: <unit information>
<value>    # <description>
# <column headers with units>
```

Example:

```
1    # ionic step
8207 # number of points
```

(continues on next page)

(continued from previous page)

```
# Module: DOS calculation
# Units: energy in eV, DOS in states/eV
#        energy    elec_states     sum_states   states_smear     sum_states
```

3.2 Data Section

Requirement	Description
Separator	Use spaces for column alignment
Precision	Controlled by input parameter (see Section 3.4)
Units	Always specify units in header or column names
Comments	Use # for comment lines

3.3 Compact Data Layout

Avoid sparse format with single value per line. Instead, output 6-8 values per line with proper alignment:

Bad (sparse):

```
1.234567
2.345678
3.456789
4.567890
```

Good (compact):

```
1.234567    2.345678    3.456789    4.567890    5.678901    6.789012
7.890123    8.901234
```

3.4 Precision Control via Input Parameters

Output precision should be controllable via input parameters, similar to `out_chg` and `out_pot`:

Example (from `out_pot`):

```
out_pot 1 8
```

- First integer: output type (1 = output total local potential)
- Second integer: precision (number of significant digits, default 8)

Implementation pattern:

```
// In input parameters
int out_type = 0;     // output type
int out_precision = 8;  // precision

// In output function
ofs << std::setprecision(out_precision) << value;
```

13.1.4 4. Output Volume Control

4.1 Reduce File Size

Current integration tests have output files with tens of thousands of lines. Recommendations:

Issue	Solution
Too many output files	Consolidate related data into fewer files
Files too large	Reduce output frequency, use compact format
Redundant information	Avoid repeating header information in every block

4.2 Balance Test Coverage and Efficiency

- Output only essential data for integration tests
- Use `out_level` parameter to control verbosity
- Consider binary format for large datasets

13.1.5 5. Naming Conventions for Physical Quantities

5.1 Standard Names (Keep Short: 3-8 Characters)

Physical Quantity	Recommended Name	Unit
Total energy	`etot`	eV
Kinetic energy	`ekin`	eV
Potential energy	`epot`	eV
Force	`force`	eV/Angstrom
Stress	`stress`	kBar
Charge	`chg`	e
Magnetization	`mag`	μB
Band index	`n`	-
k-point	`kpt`	-
Spin	`spin`	-
Occupation	`occ`	-
Energy level	`eig`	eV

5.2 Naming Style

- Use `snake_case` (lowercase with underscores)
- **Keep names short (3-8 characters)**
- Avoid abbreviations unless widely understood

13.1.6 6. Developer Checklist

Before adding a new output file or modifying an existing one, verify:

- **File name is short (3-8 characters)**
- File name follows naming conventions (Section 2)
- File is stored in `OUT.{suffix}/` directory
- File numbering starts from 1 (not 0)

- No redundant keywords (e.g., `data`) in filename
- Use correct extension (`.txt`, `.dat`, `.csr`, `.cube`)
- Add `_pw` or `_nao` for basis-specific outputs
- Header section includes `#` comment lines with:
  - Data meaning
  - Units
  - Source module
- Column headers include units
- Data uses compact format (6-8 values per line, not single value per line)
- Output precision is controllable via input parameter
- File size is reasonable (avoid tens of thousands of lines)
- Physical quantities use standard names (Section 5)
- Documentation is updated in `docs/` directory

13.1.7 7. Code Implementation Guidelines

7.1 Output Function Template

```cpp
void OutputMyData::write(const std::string& filename, int precision)
{
    std::ofstream ofs(filename);

    // Header section with comments
    ofs << "# Density of states" << std::endl;
    ofs << "# Module: DOS" << std::endl;
    ofs << "# Units: energy in eV, DOS in states/eV" << std::endl;
    ofs << ionic_step << "    # ionic step" << std::endl;
    ofs << num_data << "    # number of data points" << std::endl;
    ofs << "#";
    ofs << std::setw(15) << "energy(eV)";
    ofs << std::setw(15) << "dos";
    ofs << std::endl;

    // Data section - compact format (6 values per line)
    ofs << std::setprecision(precision);
    for (int i = 0; i < num_data; ++i)
    {
        ofs << std::setw(15) << energy[i];
        ofs << std::setw(15) << dos[i];
        if ((i + 1) % 3 == 0) ofs << std::endl;  // 3 pairs = 6 values per line
    }
    if (num_data % 3 != 0) ofs << std::endl;

    ofs.close();
}
```

7.2 Key Points

1. **Keep file names short (3-8 characters)**
2. **Add `#` comment lines for data meaning, units, and source module**
3. **Use compact format: 6-8 values per line**
4. **Make precision controllable via input parameter**
5. **Use `std::setw()` for column alignment**

13.1.8 8. Existing Good Examples

File	Location	Strengths
`chgs1.cube`	`tests/03_NAO_multik/scf_out_chg_tau/OUT.autotest/`	Short name, spin index convention
`dos.txt`	`tests/03_NAO_multik/scf_out_dos_spin2/OUT.autotest/`	Clear header with metadata
`eig_occ.txt`	Same as above	Clear block structure for spin/k-point
`mulliken.txt`	`tests/03_NAO_multik/scf_out_mul/OUT.autotest/`	Clear atom separators

13.1.9 9. Review Process

For new output formats:

1. Check if similar output already exists - reuse format if possible
2. Follow this specification
3. Add documentation in `docs/advanced/output_files/`
4. Submit PR with output sample for review

13.1.10 10. Summary

Aspect	Requirement
Directory	All outputs in `OUT.{suffix}/`
File name	**Short (3-8 chars)**, lowercase, physical quantity prefix
Extensions	`.txt`, `.dat`, `.csr`, `.cube`
Basis suffix	`_pw` or `_nao` for basis-specific outputs
Spin index	`s1`, `s2`, `s3`, `s4` (SOC), `s12` (non-collinear)
Numbering	Start from 1, not 0
Header	`#` comments with meaning, units, source module
Data	**Compact: 6-8 values per line**, space-separated
Precision	Controllable via input parameter
Volume	Reasonable file size, avoid excessive output

Remember: ABACUS outputs should present a unified, professional interface to users.

13.2 The running_scf.log file

13.2.1 Reading information

Reading version information

```
ABACUS v3.7.0
Atomic-orbital Based Ab-initio Computation at UStc
// ABACUS version and description.

Website: http://abacus.ustc.edu.cn/
// Official website for more information about ABACUS.

Documentation: https://abacus.deepmodeling.com/
// Documentation provides detailed user guides and references.

Repository: https://github.com/abacusmodeling/abacus-develop
// GitHub repository where the source code is hosted.
             https://github.com/deepmodeling/abacus-develop
```

(continues on next page)

(continued from previous page)

```
 // Another link to the same repository, ensuring access.

Commit: a339356 (Thu Jun 27 16:40:42 2024 +0800)
// The specific commit of the ABACUS source code used for this run.

Start Time is Thu Jul 18 11:34:56 2024
// The start time of the ABACUS calculation.
```

Reading general information

```
READING GENERAL INFORMATION
// The following section is reading general settings and preparing the computation.

global_out_dir = OUT.CeAl/
// The directory where output files will be stored.

global_in_card = INPUT
// The main input file for the ABACUS calculation.

pseudo_dir =
orbital_dir =
// Directories for pseudopotential and orbital files, if used.

DRANK = 1
DSIZE = 8
DCOLOR = 1
GRANK = 1
GSIZE = 1
// Parameters related to the domain decomposition and parallel execution.

The esolver type has been set to : ksdft_lcao
// The type of electronic structure solver being used in this calculation.

RUNNING WITH DEVICE  : CPU / Intel(R) Xeon(R) Platinum
// The device and CPU model used for the calculation.
```

Reading unitcell

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>=
|                                                                              |
| Reading atom information in unitcell:                                        |
| From the input file and the structure file we know the number of             |
| different elements 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.                                  |
```

(continues on next page)

(continued from previous page)

```
|                                                                              |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

READING UNITCELL INFORMATION
// The following section reads the unit cell information necessary for the␣
↪calculation.

ntype = 2
// There are two different types of atoms in the unit cell.

lattice constant (Bohr) = 1.88973
// The lattice constant of the unit cell in Bohr units.

lattice constant (Angstrom) = 1
// The lattice constant of the unit cell in Angstrom units.

READING ATOM TYPE 1
// The section starts reading details about the first type of atom in the unit cell.

atom label = Ce
// The label for the first type of atom, which is Cerium (Ce).

L=0, number of zeta = 6
// For angular momentum quantum number L=0, there are 6 zeta functions defined for␣
↪the atomic orbitals.

L=1, number of zeta = 3
// For L=1, there are 3 zeta functions.

L=2, number of zeta = 3
// For L=2, there are also 3 zeta functions.

L=3, number of zeta = 3
// Similarly, for L=3, there are 3 zeta functions.

L=4, number of zeta = 2
// And for L=4, there are 2 zeta functions.

number of atom for this type = 8
// There are 8 atoms of this type in the unit cell.

READING ATOM TYPE 2
// The section now reads details about the second type of atom in the unit cell.

atom label = Al
// The label for the second type of atom, which is Aluminum (Al).

L=0, number of zeta = 4
// For L=0, there are 4 zeta functions for Al.

L=1, number of zeta = 4
// For L=1, there are 4 zeta functions.
```

(continues on next page)

(continued from previous page)

```
L=2, number of zeta = 1
// And for L=2, there is 1 zeta function.

number of atom for this type = 16
// There are 16 atoms of this type in the unit cell.

TOTAL ATOM NUMBER = 24
// The total number of atoms in the unit cell is 24.

DIRECT COORDINATES
    atom                   x                   y                   z     mag         ␣
↪        vx                  vy                  vz
// The following table lists the direct coordinates (in units of the lattice␣
↪constant), magnetic quantum number, and velocities (vx, vy, vz) for each atom.

[...]
// The actual coordinates and velocities for each atom are listed here.

                          Volume (Bohr^3) = 3395.25
// The volume of the unit cell in Bohr^3.

                             Volume (A^3) = 503.123
// The volume of the unit cell in cubic Angstroms.

Lattice vectors: (Cartesian coordinate: in unit of a_0)
// The lattice vectors of the unit cell in Cartesian coordinates, in units of the␣
↪Bohr radius (a_0).

Reciprocal vectors: (Cartesian coordinate: in unit of 2 pi/a_0)
// The reciprocal lattice vectors, also in Cartesian coordinates, in units of 2π/a_0.
```

Reading pseudopotentials files

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                              |
| Reading pseudopotentials files:                                              |
| The pseudopotential file is in UPF format. The 'NC' indicates that           |
| the type of pseudopotential is 'norm conserving'. Functional of              |
| exchange and correlation is decided by 4 given parameters in UPF             |
| file.  We also read in the 'core correction' if there exists.                |
| Also we can read the valence electrons number and the maximal                |
| angular momentum used in this pseudopotential. We also read in the           |
| trail wave function, trail atomic density and local-pseudopotential          |
| on logrithmic grid. The non-local pseudopotential projector is also          |
| read in if there is any.                                                     |
|                                                                              |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

PAO radial cut off (Bohr) = 15
// The radial cutoff for pseudo atomic orbitals is given in Bohr units.
```

(continues on next page)

(continued from previous page)

```
Read in pseudopotential file is Ce-sp.PD04.PBE.UPF
// The filename of the pseudopotential file used for Cerium (Ce) in UPF format.

                     pseudopotential type = NC
// Indicates that the pseudopotential is of 'norm conserving' (NC) type.

          exchange-correlation functional = PBE
// Specifies that the Perdew-Burke-Ernzerhof (PBE) functional is used for exchange-
↪correlation.

                 nonlocal core correction = 1
// Indicates that there is a nonlocal core correction included in the pseudopotential.

                        valence electrons = 12
// The number of valence electrons considered in the pseudopotential, which affects␣
↪the accuracy and efficiency of the calculation.

                                     lmax = 3
// The maximum angular momentum quantum number l used in the pseudopotential.

                           number of zeta = 5
// The number of zeta functions used in the pseudopotential.

                     number of projectors = 8
// The number of projectors in the pseudopotential, which is related to the nonlocal␣
↪part.

                           L of projector = 0, 0, 1, 1, 2, 2, 3, 3
// The angular momentum quantum numbers for each projector in the pseudopotential.

                PAO radial cut off (Bohr) = 15
// The radial cutoff for pseudo atomic orbitals (PAOs) in Bohr units, which defines␣
↪the range of the orbitals.

Read in pseudopotential file is Al_ONCV_PBE-1.0.upf
// The filename of the pseudopotential file used for Aluminum (Al) in UPF format.

                     pseudopotential type = NC
// Again, indicates that the pseudopotential is of 'norm conserving' (NC) type.

          exchange-correlation functional = PBE
// The PBE functional is used for exchange-correlation for Aluminum as well.

                 nonlocal core correction = 0
// Indicates that there is no nonlocal core correction for Aluminum.

                        valence electrons = 11
// The number of valence electrons considered in the pseudopotential for Aluminum.

                                     lmax = 1
// The maximum angular momentum quantum number l for Aluminum is 1.
```

(continues on next page)

(continued from previous page)

```
                           number of zeta = 0
// There are no additional zeta functions for Aluminum in this pseudopotential.

                     number of projectors = 4
// The number of projectors for the Aluminum pseudopotential.

                           L of projector = 0, 0, 1, 1
// The angular momentum quantum numbers for each projector in the Aluminum␣
↪pseudopotential.

      initial pseudo atomic orbital number = 136
// The initial number of pseudo atomic orbitals used in the calculation.

                                   NLOCAL = 888
// The number of nonlocal operations, which may be related to the nonlocal␣
↪pseudopotential projectors.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
↪%%%%%%%%%%%%%%%%%%%%
 Warning: the number of valence electrons in pseudopotential > 4 for Ce: [Xe] 4f1 5d1␣
↪6s2
 Warning: the number of valence electrons in pseudopotential > 3 for Al: [Ne] 3s2 3p1
 Pseudopotentials with additional electrons can yield (more) accurate outcomes, but␣
↪may be less efficient.
 If you‘re confident that your chosen pseudopotential is appropriate, you can safely␣
↪ignore this warning.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
↪%%%%%%%%%%%%%%%%%%%%
// A warning message indicating that the pseudopotentials include more valence␣
↪electrons than typical, which can improve accuracy but at the cost of computational␣
↪efficiency.

Warning_Memory_Consuming allocated:  FFT::grid 6.5918 MB
// A warning about the memory consumption for the Fast Fourier Transform (FFT) grid.
```

13.2.2 Setup Tasks

Setup plane waves of charge/potential

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                    |
| Setup plane waves of charge/potential:                             |
| Use the energy cutoff and the lattice vectors to generate the      |
| dimensions of FFT grid. The number of FFT grid on each processor   |
| is 'nrxx'. The number of plane wave basis in reciprocal space is   |
| different for charge/potential and wave functions. We also set    |
| the 'sticks' for the parallel of FFT. The number of plane waves    |
| is 'npw' in each processor.                                        |
|                                                                    |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

(continues on next page)

(continued from previous page)

```
SETUP THE PLANE WAVE BASIS
energy cutoff for charge/potential (unit:Ry) = 600
// The energy cutoff used for charge and potential calculations in Rydberg units.

            fft grid for charge/potential = [ 120, 120, 120 ]
// The dimensions of the FFT grid used for charge and potential calculations.

                        fft grid division = [ 3, 3, 3 ]
// The division of the FFT grid among processors.

        big fft grid for charge/potential = [ 40, 40, 40 ]
// The dimensions of the 'big' FFT grid, which might be used for parallelization.

                                     nbxx = 8000
// The total number of plane waves in the charge/potential basis.

                                     nrxx = 216000
// The number of FFT grid points on each processor.

SETUP PLANE WAVES FOR CHARGE/POTENTIAL
                    number of plane waves = 842641
// The total number of plane waves used for charge/potential.

                         number of sticks = 10781
// The number of 'sticks' used in the FFT parallelization.

PARALLEL PW FOR CHARGE/POTENTIAL
// The distribution of plane waves among processors for charge/potential calculations.

[...]
// The actual distribution of plane waves and the number of plane waves per processor.

--------------- sum -------------------
        8          10781         842641
// The sum of plane waves and the total number of processors used.

                           number of |g| = 2844
// The total number of G-vectors in the reciprocal space.

                                  max |g| = 54.2697
// The maximum magnitude of the G-vectors.

                                  min |g| = 0.0790412
// The minimum magnitude of the G-vectors.

----------- Double Check Mixing Parameters Begin ------------
mixing_type: pulay
// The type of charge mixing used in the SCF procedure, which is Pulay mixing.

mixing_beta: 0.4
// The Pulay mixing beta parameter, which controls the degree of mixing.
```

(continues on next page)

(continued from previous page)

```
mixing_gg0: 0
mixing_gg0_min: 0.1
mixing_ndim: 8
// Parameters related to the Pulay mixing method.

----------- Double Check Mixing Parameters End ------------

SETUP THE ELECTRONS NUMBER
// The section where the total number of electrons is set up based on the atomic␣
↪composition.

[...]
// The setup for the electron numbers for each atom type and the total number of␣
↪electrons.

DONE : SETUP UNITCELL Time : 0.611609 (SEC)
// The time taken to complete the unit cell setup.
```

Doing symmetry analysis

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                              |
| Doing symmetry analysis:                                                     |
| We calculate the norm of 3 vectors and the angles between them,              |
| the type of Bravais lattice is given. We can judge if the unticell           |
| is a primitive cell. Finally we give the point group operation for           |
| this unitcell. We use the point group operations to do symmetry              |
| analysis on given k-point mesh and the charge density.                       |
|                                                                              |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

LATTICE VECTORS: (CARTESIAN COORDINATE: IN UNIT OF A0)
// The lattice vectors of the unit cell in Cartesian coordinates, normalized to the␣
↪Bohr radius (a0).

              +7.9535                  +0                  +0
                   +0             +7.9535                  +0
                   +0                  +0             +7.9535

                        right hand lattice = 1
// Indicates the handedness of the lattice.

                                    NORM_A = 7.9535
                                    NORM_B = 7.9535
                                    NORM_C = 7.9535
// The norms (lengths) of the lattice vectors.

                            ALPHA (DEGREE) = 90
                            BETA  (DEGREE) = 90
                            GAMMA (DEGREE) = 90
```

(continues on next page)

(continued from previous page)

```
// The angles between the lattice vectors in degrees.

The lattice vectors have been changed (STRU_SIMPLE.cif)
// Indicates that the lattice vectors have been updated or defined in the STRU_SIMPLE.
↪cif file.

(for optimal symmetric configuration:)
// The following parameters describe the lattice for the optimal symmetric␣
↪configuration.

                            BRAVAIS TYPE = 1
                    BRAVAIS LATTICE NAME = 01. Cubic P (simple)
                                   ibrav = 1
                                   IBRAV = 1
                                BRAVAIS = SIMPLE CUBIC
                      LATTICE CONSTANT A = 7.9535
// The lattice constant for the simple cubic lattice.

optimized lattice volume: 503.124
// The optimized volume of the unit cell.

optimized primitive cell volume: 125.781
// The optimized volume of the primitive cell.

Original cell was built up by 4 primitive cells.
// The original unit cell is composed of 4 primitive cells.

                         ROTATION MATRICES = 48
// The number of rotation matrices for the symmetry operations.

              PURE POINT GROUP OPERATIONS = 24
                   SPACE GROUP OPERATIONS = 48
// The number of operations in the point group and the space group.

                                        C2 = 3
                                        C3 = 8
                                        C4 = 0
                                        C6 = 0
                                        S1 = 6
                                        S3 = 0
                                        S4 = 6
                                        S6 = 0
// The counts of different symmetry operations (rotations and reflections).

                                POINT GROUP = T_d
                 POINT GROUP IN SPACE GROUP = O_h
// The point group and the space group for the unit cell.

DONE : SYMMETRY Time : 0.781568 (SEC)
// The time taken to complete the symmetry analysis.
```

Setup K-points

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                     |
| Setup K-points                                                      |
| We setup the k-points according to input parameters.               |
| The reduced k-points are set according to symmetry operations.     |
| We treat the spin as another set of k-points.                      |
|                                                                     |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

SETUP K-POINTS
// This section describes the setup of k-points, which are used for integrating over␣
↪the Brillouin zone.

                                    nspin = 1
// The number of spin dimensions; 1 for non-spin-polarized calculations.

                   Input type of k points = Monkhorst-Pack(Gamma)
// The method used for generating k-points, which is the Monkhorst-Pack scheme␣
↪centered at the Gamma point.

                                   nkstot = 1
// The total number of k-points generated.

                       right hand lattice = 1
// Indicates the handedness of the lattice used for k-point generation.

(for reciprocal lattice: )
// The following parameters describe the reciprocal lattice used for k-point␣
↪generation.

                             BRAVAIS TYPE = 1
                     BRAVAIS LATTICE NAME = 01. Cubic P (simple)
                                    ibrav = 1
                       right hand lattice = 1
// The Bravais lattice type and its properties for the reciprocal lattice.

(for k-lattice: )
// Parameters for the k-point lattice.

                             BRAVAIS TYPE = 1
                     BRAVAIS LATTICE NAME = 01. Cubic P (simple)
                                    ibrav = 1
                       right hand lattice = 1

                        ROTATION MATRICES = 48
// The number of rotation matrices for the symmetry operations in the reciprocal␣
↪space.

                               nkstot_ibz = 1
// The number of k-points in the irreducible Brillouin zone.
```

(continues on next page)

(continued from previous page)

```
K-POINTS REDUCTION ACCORDING TO SYMMETRY
// The reduction of k-points according to the symmetry operations to find the␣
↪irreducible Brillouin zone.

     IBZ    DIRECT_X    DIRECT_Y    DIRECT_Z  WEIGHT  ibz2bz
       1  0.00000000  0.00000000  0.00000000  1.0000       0

                               nkstot now = 1
// The current total number of k-points after symmetry reduction.

K-POINTS DIRECT COORDINATES
// The direct coordinates of the k-points in the Brillouin zone.

 KPOINTS    DIRECT_X    DIRECT_Y    DIRECT_Z  WEIGHT
       1  0.00000000  0.00000000  0.00000000  1.0000

           k-point number in this process = 1
       minimum distributed K point number = 1

K-POINTS CARTESIAN COORDINATES
// The Cartesian coordinates of the k-points.

 KPOINTS CARTESIAN_X CARTESIAN_Y CARTESIAN_Z  WEIGHT
       1  0.00000000  0.00000000  0.00000000  2.0000

K-POINTS DIRECT COORDINATES
// The direct coordinates of the k-points, repeated for clarity.

 KPOINTS    DIRECT_X    DIRECT_Y    DIRECT_Z  WEIGHT
       1  0.00000000  0.00000000  0.00000000  2.0000

 DONE : INIT K-POINTS Time : 1.01765 (SEC)
// The time taken to initialize the k-points.
```

Setup plane waves of wave functions

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                              |
| Setup plane waves of wave functions:                                         |
| Use the energy cutoff and the lattice vectors to generate the                |
| dimensions of FFT grid. The number of FFT grid on each processor             |
| is 'nrxx'. The number of plane wave basis in reciprocal space is             |
| different for charge/potential and wave functions. We also set               |
| the 'sticks' for the parallel of FFT. The number of plane wave of            |
| each k-point is 'npwk[ik]' in each processor                                 |
|                                                                              |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

SETUP PLANE WAVES FOR WAVE FUNCTIONS
// This section describes the setup of plane waves for the wave functions in the␣
↪calculation.
```

(continues on next page)

(continued from previous page)

```
     energy cutoff for wavefunc (unit:Ry) = 150
// The energy cutoff used for the wave functions in Rydberg units.

              fft grid for wave functions = [ 120, 120, 120 ]
// The dimensions of the FFT grid used specifically for wave functions.

                    number of plane waves = 105591
// The total number of plane waves used for the wave functions.

                         number of sticks = 2709
// The number of 'sticks' used in the FFT parallelization for wave functions.

PARALLEL PW FOR WAVE FUNCTIONS
// The distribution of plane waves among processors for wave function calculations.

     PROC   COLUMNS(POT)             PW
        1            339          13201
        2            338          13198
        3            338          13198
        4            338          13198
        5            339          13199
        6            339          13199
        7            339          13199
        8            339          13199
-------------- sum -------------------
        8           2709         105591

// The sum of plane waves and the total number of processors used for wave function␣
↪calculations.

                           occupied bands = 136
                                  NLOCAL = 888
                                  NBANDS = 200
                                  NBANDS = 200
// Parameters indicating the number of occupied bands, local bands, and total bands␣
↪in the calculation.

SET NONLOCAL PSEUDOPOTENTIAL PROJECTORS
// The setup for nonlocal pseudopotential projectors, which are important for systems␣
↪with nonlocal pseudopotentials.

SET NONLOCAL PSEUDOPOTENTIAL PROJECTORS
// Repetition of the setup for nonlocal pseudopotential projectors.

max number of nonlocal projetors among all species is 8
// The maximum number of nonlocal projectors for all species in the system.

Warning_Memory_Consuming allocated:  TwoCenterTable: Kinetic 11.3983 MB
// A warning about the memory consumption for the kinetic part of the two-center␣
↪table.
```

(continues on next page)

(continued from previous page)

```
Warning_Memory_Consuming allocated:  TwoCenterTable: Overlap 11.3983 MB
// A warning about the memory consumption for the overlap part of the two-center␣
↪table.

Warning_Memory_Consuming allocated:  TwoCenterTable: Nonlocal 5.26246 MB
// A warning about the memory consumption for the nonlocal part of the two-center␣
↪table.

SETUP THE DIVISION OF H/S MATRIX
// The division of the Hamiltonian and overlap matrices using 2D block algorithms.

 divide the H&S matrix using 2D block algorithms.
                                    nb2d = 1
                                    nloc = 100352

// Parameters indicating the number of 2D blocks and the number of local orbitals.

-----------------------------------------
SELF-CONSISTENT
-----------------------------------------
// The beginning of the self-consistent field (SCF) calculation section.
```

13.2.3 Running scf processes

Search adjacent atoms (init)

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
|                                                                              |
| Search adjacent atoms:                                                       |
| Set the adjacent atoms for each atom and set the periodic boundary           |
| condition for the atoms on real space FFT grid. For k-dependent              |
| algorithm, we also need to set the sparse H and S matrix element             |
| for each atom.                                                               |
|                                                                              |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

SETUP SEARCHING RADIUS FOR PROGRAM TO SEARCH ADJACENT ATOMS
// The program sets up a searching radius to find adjacent atoms for each atom in the␣
↪unit cell.

                  longest orb rcut (Bohr) = 7
// The longest orbital radius cutoff used in the search.

   longest nonlocal projector rcut (Bohr) = 2.46
// The longest nonlocal projector radius cutoff.

              searching radius is (Bohr)) = 18.9
// The overall searching radius for finding adjacent atoms.

         searching radius unit is (Bohr)) = 1.89

SETUP EXTENDED REAL SPACE GRID FOR GRID INTEGRATION
```

(continues on next page)

(continued from previous page)

```
// The setup of an extended real space grid for numerical integration.

                          real space grid = [ 120, 120, 120 ]
// The dimensions of the real space grid.

                 big cell numbers in grid = [ 40, 40, 40 ]
// The number of 'big cells' within the grid.

             meshcell numbers in big cell = [ 3, 3, 3 ]
// The number of mesh cells within each big cell.

                        extended fft grid = [ 19, 19, 19 ]
// The dimensions of the extended FFT grid.

                dimension of extened grid = [ 79, 79, 79 ]
// The actual dimensions of the extended grid.

                           UnitCellTotal = 27
// The total number of unit cells.

              Atom number in sub-FFT-grid = 24
// The number of atoms within the sub-FFT grid.

    Local orbitals number in sub-FFT-grid = 888
// The number of local orbitals within the sub-FFT grid.

                                ParaV.nnr = 441130
// A parameter related to the parallelization of the calculation.

                                     nnrg = 1878264
// The number of G-vectors used in the calculation.

Warning_Memory_Consuming allocated:  Gint::hRGint 14.5 MB
// A warning about the memory consumption for the Hamiltonian real space grid␣
↪interaction.

Warning_Memory_Consuming allocated:  Gint::DMRGint 14.5 MB
// A warning about the memory consumption for the overlap real space grid interaction.

Warning_Memory_Consuming allocated:  pvpR_reduced 14.3 MB
// A warning about the memory consumption for the reduced pseudopotential real space␣
↪grid.

Warning_Memory_Consuming allocated:  LOC::DM_R 14.3 MB
// A warning about the memory consumption for the local orbitals real space grid.

                                  init_chg = atomic
// The initial charge density is set to be atomic.

DONE : INIT SCF Time : 4.10934 (SEC)
// The time taken to initialize the SCF calculation.
```

(continues on next page)

(continued from previous page)

Scf iteration

```
LC AO ALGORITHM --------------- ION= 1 ELEC= 1--------------------------------

Density error is 0.0441106498563
// The error in the electron density from the current iteration of the SCF␣
↪calculation.

---------------------------------------------------------
     Energy           Rydberg                 eV
---------------------------------------------------------
 E_KohnSham     -2970.5084378350     -40415.8407116355
// The Kohn-Sham energy, which is a key quantity in DFT calculations.
 E_Harris       -2971.5066745628     -40429.4224190859
// The Harris energy, an alternative energy expression sometimes used in SCF␣
↪iterations.
 E_Fermi        0.9352224901         12.7243547631
// The Fermi energy, which represents the highest occupied energy level in the system.
---------------------------------------------------------

LCAO ALGORITHM --------------- ION=   1  ELEC=   2--------------------------------

 Density error is 0.07324240119
---------------------------------------------------------
     Energy           Rydberg                 eV
---------------------------------------------------------
 E_KohnSham     -2970.8761527408     -40420.8437295927
 E_Harris       -2985.6598714921     -40621.9865422408
 E_Fermi        0.9942982614         13.5281218666
---------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   3--------------------------------

 Density error is 0.0256800696305
---------------------------------------------------------
     Energy           Rydberg                 eV
---------------------------------------------------------
 E_KohnSham     -2971.1839545654     -40425.0315882625
 E_Harris       -2973.1043954412     -40451.1605268455
 E_Fermi        0.9712074216         13.2139548737
---------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   4--------------------------------

 Density error is 0.0102215946188
---------------------------------------------------------
     Energy           Rydberg                 eV
```

(continues on next page)

(continued from previous page)

```
----------------------------------------------------------
 E_KohnSham     -2971.2225078250     -40425.5561322694
 E_Harris       -2974.8325144469     -40474.6727921456
 E_Fermi        0.9535174273         12.9732701541
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   5--------------------------------

 Density error is 0.00465926410973
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2191319329     -40425.5102009013
 E_Harris       -2972.1662486317     -40438.3963846764
 E_Fermi        0.9547114124         12.9895151541
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   6--------------------------------

 Density error is 0.00290323197151
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2213953394     -40425.5409961268
 E_Harris       -2971.3541089783     -40427.3466578181
 E_Fermi        0.9597988625         13.0587334638
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   7--------------------------------

 Density error is 0.000225469262322
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221507153     -40425.5512735432
 E_Harris       -2971.1959132112     -40425.1942939861
 E_Fermi        0.9575455903         13.0280761230
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   8--------------------------------

 Density error is 0.000105298036339
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221543072     -40425.5513224128
 E_Harris       -2971.2345871224     -40425.7204795427
 E_Fermi        0.9573864389         13.0259107570
```

(continues on next page)

(continued from previous page)

```
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=   9--------------------------------

 Density error is 1.73342865404e-05
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545084     -40425.5513251506
 E_Harris       -2971.2284300744     -40425.6367086066
 E_Fermi        0.9573421242         13.0253078248
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  10--------------------------------

 Density error is 2.44145213497e-05
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545026     -40425.5513250720
 E_Harris       -2971.2237593087     -40425.5731595783
 E_Fermi        0.9573530336         13.0254562547
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  11--------------------------------

 Density error is 1.77043483202e-05
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545218     -40425.5513253324
 E_Harris       -2971.2233998438     -40425.5682688075
 E_Fermi        0.9573601195         13.0255526627
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  12--------------------------------

 Density error is 1.08567789394e-06
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545365     -40425.5513255324
 E_Harris       -2971.2226767646     -40425.5584308108
 E_Fermi        0.9573493003         13.0254054601
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  13--------------------------------
```

(continues on next page)

(continued from previous page)

```
 Density error is 1.39646278617e-05
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545147     -40425.5513252362
 E_Harris       -2971.2222832103     -40425.5530762294
 E_Fermi        0.9573484591         13.0253940153
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  14--------------------------------

 Density error is 3.71116003478e-07
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545386     -40425.5513255615
 E_Harris       -2971.2222838082     -40425.5530843652
 E_Fermi        0.9573492888         13.0254053046
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  15--------------------------------

 Density error is 2.46641928838e-07
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545386     -40425.5513255614
 E_Harris       -2971.2221840706     -40425.5517273647
 E_Fermi        0.9573494485         13.0254074765
----------------------------------------------------------


 LCAO ALGORITHM --------------- ION=   1  ELEC=  16--------------------------------

 Density error is 2.05528957543e-07
----------------------------------------------------------
     Energy           Rydberg                 eV
----------------------------------------------------------
 E_KohnSham     -2971.2221545386     -40425.5513255622
 E_Harris       -2971.2221699838     -40425.5515357039
 E_Fermi        0.9573494392         13.0254073506
----------------------------------------------------------

LC AO ALGORITHM --------------- ION= 1 ELEC= 17--------------------------------

Density error is 6.53205775131e-08
// The error in the electron density has become very small, indicating near-
↪convergence of the SCF cycle.
// Additional iterations of the SCF calculation, showing the gradual convergence of␣
```

(continues on next page)

(continued from previous page)

```
↪the density error and energies.
```

Scf results

```
------------------------------------------------------------
     Energy           Rydberg                 eV
------------------------------------------------------------
 E_KohnSham     -2971.2221545387     -40425.5513255626
 E_KS(sigma->0) -2971.1513960918     -40424.5886075033
// The Kohn-Sham energy with the sigma component of the density matrix going to zero,␣
↪an indicator of the kinetic energy contribution.
 E_Harris       -2971.2221622072     -40425.5514298979
 E_band         -562.7029473746      -7655.9663656893
// The band energy, associated with the dispersion of the energy bands.
 E_one_elec     -1460.3005783359     -19868.4086580639
// The one-electron contribution to the energy.
 E_Hartree      672.7815109284       9153.6620576760
// The Hartree energy, representing the classical electrostatic interaction between␣
↪electron densities.
 E_xc           -486.3000633355      -6616.4517991236
// The exchange-correlation energy, accounting for quantum mechanical effects.
 E_Ewald        -1697.2615069019     -23092.4274899325
// The Ewald energy, the contribution to the total energy from the Ewald summation␣
↪for long-range interactions.
 E_entropy(-TS) -0.1415168938        -1.9254361186
// The entropy contribution to the free energy (negative of the product of␣
↪temperature and entropy).
 E_descf        0.0000000000         0.0000000000
// The contribution to the energy from the density-cutoff term in the pseudopotential.
 E_exx          0.0000000000         0.0000000000
// The exact exchange energy, if applicable (usually zero in standard DFT␣
↪calculations).
 E_Fermi        0.9573493498         13.0254061340
// The Fermi energy at this stage of the calculation.

charge density convergence is achieved
// A statement indicating that the charge density has converged to the desired␣
↪accuracy.

final etot is -40425.551326 eV
// The final total energy of the system after the SCF calculation has fully converged.

EFERMI = 13.025406134 eV
// The final Fermi energy of the system.

STATE ENERGY(eV) AND OCCUPATIONS    NSPIN == 1
// The energies of the electronic states and their occupations for each k-point.

1/1 kpoint (Cartesian) = 0.0000 0.0000 0.0000 (13201 pws)
// The k-point at the center of the Brillouin zone and the number of plane waves␣
```

(continues on next page)

(continued from previous page)

```
↪associated with it.
       1       -90.2506        2.00000
       2       -90.2494        2.00000
       3       -90.2494        2.00000
       4       -90.2494        2.00000
       5       -90.2494        2.00000
       6       -90.2494        2.00000
       7       -90.2494        2.00000
       8       -90.2482        2.00000
       9       -90.2482        2.00000
      10       -90.2482        2.00000
      11       -90.2482        2.00000
      12       -90.2482        2.00000
      13       -90.2482        2.00000
      14       -90.2482        2.00000
      15       -90.2482        2.00000
      16       -90.2482        2.00000
      17       -51.8782        2.00000
      18       -51.8681        2.00000
      19       -51.8681        2.00000
      20       -51.8681        2.00000
      21       -51.8681        2.00000
      22       -51.8681        2.00000
      23       -51.8681        2.00000
      24       -51.8583        2.00000
      25       -51.8583        2.00000
      26       -51.8583        2.00000
      27       -51.8580        2.00000
      28       -51.8580        2.00000
      29       -51.8580        2.00000
      30       -51.8580        2.00000
      31       -51.8580        2.00000
      32       -51.8580        2.00000
      33       -51.7617        2.00000
      34       -51.7617        2.00000
      35       -51.7612        2.00000
      36       -51.7612        2.00000
      37       -51.7612        2.00000
      38       -51.7612        2.00000
      39       -51.7612        2.00000
      40       -51.7612        2.00000
      41       -51.7604        2.00000
      42       -51.7604        2.00000
      43       -51.7604        2.00000
      44       -51.7591        2.00000
      45       -51.7591        2.00000
      46       -51.7591        2.00000
      47       -51.7591        2.00000
      48       -51.7591        2.00000
      49       -51.7591        2.00000
      50       -51.7578        2.00000
      51       -51.7578        2.00000
```

(continues on next page)

(continued from previous page)

```
      52       -51.7578        2.00000
      53       -51.7578        2.00000
      54       -51.7578        2.00000
      55       -51.7578        2.00000
      56       -51.7556        2.00000
      57       -51.7556        2.00000
      58       -51.7556        2.00000
      59       -51.7556        2.00000
      60       -51.7556        2.00000
      61       -51.7556        2.00000
      62       -51.7539        2.00000
      63       -51.7539        2.00000
      64       -51.7539        2.00000
      65       -21.8364        2.00000
      66       -21.7006        2.00000
      67       -21.7006        2.00000
      68       -21.7006        2.00000
      69       -21.7006        2.00000
      70       -21.7006        2.00000
      71       -21.7006        2.00000
      72       -21.5621        2.00000
      73       -5.23097        2.00000
      74       -5.23097        2.00000
      75       -5.23097        2.00000
      76       -5.23097        2.00000
      77       -5.23097        2.00000
      78       -5.23097        2.00000
      79       -5.02839        2.00000
      80       -5.02839        2.00000
      81       -5.02839        2.00000
      82       -4.76611        2.00000
      83       -4.76611        2.00000
      84       -4.76611        2.00000
      85       -4.76611        2.00000
      86       -4.76611        2.00000
      87       -4.76611        2.00000
      88       -4.42730        2.00000
      89       -4.42730        2.00000
      90       -4.42730        2.00000
      91       -4.21659        2.00000
      92       -4.21659        2.00000
      93       -4.21659        2.00000
      94       -4.21659        2.00000
      95       -4.21659        2.00000
      96       -4.21659        2.00000
      97        3.53302        2.00000
      98        5.39913        2.00000
      99        5.39913        2.00000
     100        5.39913        2.00000
     101        5.39913        2.00000
     102        5.39913        2.00000
     103        5.39913        2.00000
```

(continues on next page)

(continued from previous page)

```
    104        8.74954        2.00000
    105        9.08287        2.00000
    106        9.08287        2.00000
    107        9.08287        2.00000
    108        9.08287        2.00000
    109        9.08287        2.00000
    110        9.08287        2.00000
    111        9.57721        2.00000
    112        9.57721        2.00000
    113        9.57721        2.00000
    114        9.77291        2.00000
    115        9.77291        2.00000
    116        9.77291        2.00000
    117        9.77291        2.00000
    118        9.77291        2.00000
    119        9.77291        2.00000
    120        10.1656        2.00000
    121        10.1656        2.00000
    122        10.4293        2.00000
    123        10.4293        2.00000
    124        10.4293        2.00000
    125        12.1630        1.99999
    126        12.1630        1.99999
    127        12.1630        1.99999
    128        12.1630        1.99999
    129        12.1630        1.99999
    130        12.1630        1.99999
    131        12.9733        1.21344
    132        12.9733        1.21344
    133        12.9733        1.21344
    134        12.9733        1.21344
    135        12.9733        1.21344
    136        12.9733        1.21344
    137        13.1019       0.690941
    138        13.1019       0.690941
    139        13.1019       0.690941
    140        13.1019       0.690941
    141        13.1019       0.690940
    142        13.1019       0.690940
    143        13.1733       0.442128
    144        13.4597      0.0240081
    145        13.4597      0.0240081
    146        13.4597      0.0240081
    147        13.5463     0.00678529
    148        13.5463     0.00678529
    149        13.5463     0.00678528
    150        13.5463     0.00678527
    151        13.5463     0.00678527
    152        13.5463     0.00678526
    153        13.6052     0.00258518
    154        13.6052     0.00258518
    155        13.6052     0.00258518
```

(continues on next page)

(continued from previous page)

```
     156        13.6052     0.00258517
     157        13.6052     0.00258517
     158        13.6052     0.00258517
     159        13.7097    0.000376289
     160        13.7097    0.000376289
     161        13.7097    0.000376289
     162        13.7131    0.000351588
     163        13.7131    0.000351587
     164        13.7131    0.000351586
     165        13.7131    0.000351586
     166        13.7131    0.000351586
     167        13.7131    0.000351585
     168        13.8238    3.33179e-05
     169        13.8238    3.33179e-05
     170        13.8238    3.33179e-05
     171        13.8978    5.78882e-06
     172        13.8978    5.78881e-06
     173        13.8978    5.78881e-06
     174        13.8978    5.78879e-06
     175        13.8978    5.78879e-06
     176        13.8978    5.78879e-06
     177        13.9176    3.54072e-06
     178        13.9176    3.54072e-06
     179        13.9176    3.54072e-06
     180        13.9299    2.58893e-06
     181        14.1919    1.33980e-09
     182        14.1919    1.33980e-09
     183        14.1919    1.33980e-09
     184        14.1919    1.33980e-09
     185        14.1919    1.33980e-09
     186        14.1919    1.33980e-09
     187        14.3213    1.64246e-11
     188        14.3213    1.64246e-11
     189        14.3213    1.64246e-11
     190        14.3213    1.64246e-11
     191        14.3213    1.64246e-11
     192        14.3213    1.64246e-11
     193        14.5304    5.21805e-15
     194        14.5304    5.21805e-15
     195        14.5304    5.21805e-15
     196        14.5304    5.21805e-15
     197        14.5304    5.21805e-15
     198        14.5304    5.21805e-15
     199        14.5998    3.33067e-16
     200        14.7810         0.00000
// The list of state energies and their occupations for the converged calculation.

Warning_Memory_Consuming allocated:  Force::dS_K 10.0967 MB
// A warning indicating the memory allocated for the calculation of forces,␣
↪specifically the kinetic contribution.

Warning_Memory_Consuming allocated:  Stress::dHr 10.0967 MB
```

(continues on next page)

(continued from previous page)

```
// A warning about the memory consumption for the calculation of stress, specifically␣
↪the Hellmann-Feynman contribution.

Warning_Memory_Consuming allocated:  Stress::dSR 20.1933 MB
// A warning about the memory consumption for the calculation of stress, specifically␣
↪the nonlocal pseudopotential contribution.

Warning_Memory_Consuming allocated:  Force::dTVNL 10.0967 MB
// A warning about the memory consumption for the calculation of forces, specifically␣
↪the nonlocal pseudopotential contribution.

correction force for each atom along direction 1 is 2.90404e-14
correction force for each atom along direction 2 is 2.49665e-14
correction force for each atom along direction 3 is -3.50742e-14
// The correction forces applied to each atom in the unit cell along the three␣
↪principal directions.
```

13.2.4 Results summary

TOTAL-FORCE

```
------------------------------------------------------------------------------------------
↪----
 TOTAL-FORCE (eV/Angstrom)
------------------------------------------------------------------------------------------
↪----
// The total force experienced by each atom in the unit cell, expressed in␣
↪electronvolts per angstrom.
                      Ce1        -0.0000024495        -0.0000024495         0.
↪0000024495
                      Ce2        -0.0000024495        -0.0000024495        -0.
↪0000024495
                      Ce3        -0.0000024495         0.0000024495        -0.
↪0000024495
                      Ce4        -0.0000024495         0.0000024495         0.
↪0000024495
                      Ce5         0.0000024495        -0.0000024495        -0.
↪0000024495
                      Ce6         0.0000024495        -0.0000024495         0.
↪0000024495
                      Ce7         0.0000024495         0.0000024495         0.
↪0000024495
                      Ce8         0.0000024495         0.0000024495        -0.
↪0000024495
                      Al1         0.0000000000         0.0000042432        -0.
↪0000042432
                      Al2         0.0000000000         0.0000000000         0.
↪0000000000
                      Al3        -0.0000026880         0.0000026880         0.
↪0000015552
                      Al4        -0.0000042432         0.0000000000        -0.
↪0000042432
```

(continues on next page)

(continued from previous page)

```
                        Al5          0.0000000000        -0.0000042432          0.
↪0000042432
                        Al6          0.0000000000         0.0000000000          0.
↪0000000000
                        Al7         -0.0000042432        -0.0000042432          0.
↪0000000000
                        Al8         -0.0000026880         0.0000015552          0.
↪0000026880
                        Al9         -0.0000015552         0.0000026880          0.
↪0000026880
                       Al10          0.0000000000         0.0000000000          0.
↪0000000000
                       Al11          0.0000042432         0.0000042432          0.
↪0000000000
                       Al12          0.0000042432         0.0000000000          0.
↪0000042432
                       Al13          0.0000015552        -0.0000026880         -0.
↪0000026880
                       Al14          0.0000000000         0.0000000000          0.
↪0000000000
                       Al15          0.0000026880        -0.0000026880         -0.
↪0000015552
                       Al16          0.0000026880        -0.0000015552         -0.
↪0000026880
-------------------------------------------------------------------------------------
↪----
// The table listing the total forces on each atom, indicating whether the system is␣
↪close to a force minimum (stable configuration).
```

TOTAL-STRESS

```
---------------------------------------------------------------
 TOTAL-STRESS (KBAR)
---------------------------------------------------------------
// The total stress experienced by the unit cell, a measure of the pressure in␣
↪different directions.

        20.9280307419        -0.0000000000         0.0000000000
        -0.0000000000        20.9280307419         0.0000000000
         0.0000000000         0.0000000000        20.9280307419

TOTAL-PRESSURE: 20.928031 KBAR
// The total pressure exerted on the system, calculated from the stress tensor.
```

FINAL_ETOT_IS

```
-----------------------------------------
 !FINAL_ETOT_IS -40425.5513255625628517 eV
 -----------------------------------------
// The final total energy of the system after the SCF calculation, expressed in␣
↪electron volts.
```

TIME STATISTICS

```
TIME STATISTICS
----------------------------------------------------------------------------------
      CLASS_NAME                   NAME            TIME/s  CALLS   AVG/s  PER/%
----------------------------------------------------------------------------------
// A summary of the time spent in different parts of the calculation, providing␣
↪insights into the efficiency and potential bottlenecks.
                       total                       500.80 11       45.53  100.00
 Driver                reading                     0.09   1        0.09   0.02
 Input                 Init                        0.08   1        0.08   0.02
 Input_Conv            Convert                     0.00   1        0.00   0.00
 Driver                driver_line                 500.71 1        500.71 99.98
 UnitCell              check_tau                   0.00   1        0.00   0.00
 ESolver_KS_LCAO       before_all_runners          2.10   1        2.10   0.42
 PW_Basis_Sup          setuptransform              0.03   1        0.03   0.01
 PW_Basis_Sup          distributeg                 0.02   1        0.02   0.00
 mymath                heapsort                    0.02   2016     0.00   0.00
 Symmetry              analy_sys                   0.17   1        0.17   0.03
 PW_Basis_K            setuptransform              0.01   1        0.01   0.00
 PW_Basis_K            distributeg                 0.01   1        0.01   0.00
 PW_Basis              setup_struc_factor          0.13   1        0.13   0.03
 NOrbital_Lm           extra_uniform               0.18   26       0.01   0.04
 Mathzone_Add1         SplineD2                    0.00   26       0.00   0.00
 Mathzone_Add1         Cubic_Spline_Interpolation  0.01   26       0.00   0.00
 Mathzone_Add1         Uni_Deriv_Phi               0.16   26       0.01   0.03
 ppcell_vl             init_vloc                   0.12   1        0.12   0.02
 Ions                  opt_ions                    498.27 1        498.27 99.50
 ESolver_KS_LCAO       runner                      373.62 1        373.62 74.60
 ESolver_KS_LCAO       before_scf                  1.64   1        1.64   0.33
 ESolver_KS_LCAO       beforesolver                0.76   1        0.76   0.15
 ESolver_KS_LCAO       set_matrix_grid             0.58   1        0.58   0.12
 atom_arrange          search                      0.00   1        0.00   0.00
 Grid_Technique        init                        0.52   1        0.52   0.10
 Grid_BigCell          grid_expansion_index        0.03   2        0.01   0.01
 Record_adj            for_2d                      0.02   1        0.02   0.00
 Grid_Driver           Find_atom                   0.03   792      0.00   0.01
 LCAO_domain           grid_prepare                0.00   1        0.00   0.00
 Veff                  initialize_HR               0.00   1        0.00   0.00
 Overlap               initialize_SR               0.00   1        0.00   0.00
 Ekinetic              initialize_HR               0.00   1        0.00   0.00
 Nonlocal              initialize_HR               0.00   1        0.00   0.00
 Charge                set_rho_core                0.14   1        0.14   0.03
 PW_Basis_Sup          recip2real                  3.44   122      0.03   0.69
 PW_Basis_Sup          gathers_scatterp            1.72   122      0.01   0.34
 Charge                atomic_rho                  0.14   1        0.14   0.03
 Potential             init_pot                    0.30   1        0.30   0.06
 Potential             update_from_charge          11.23  18       0.62   2.24
 Potential             cal_fixed_v                 0.01   1        0.01   0.00
 PotLocal              cal_fixed_v                 0.01   1        0.01   0.00
 Potential             cal_v_eff                   11.19  18       0.62   2.23
 H_Hartree_pw          v_hartree                   1.32   18       0.07   0.26
 PW_Basis_Sup          real2recip                  3.89   144      0.03   0.78
```

(continues on next page)

(continued from previous page)

```
PW_Basis_Sup          gatherp_scatters          1.92   144      0.01   0.38
PotXC                 cal_v_eff                 9.81   18       0.55   1.96
XC_Functional         v_xc                      11.76  20       0.59   2.35
Potential             interpolate_vrs           0.03   18       0.00   0.01
Symmetry              rhog_symmetry             12.41  18       0.69   2.48
Symmetry              group fft grids           2.64   18       0.15   0.53
H_Ewald_pw            compute_ewald             0.01   1        0.01   0.00
Charge_Mixing         init_mixing               0.00   1        0.00   0.00
HSolverLCAO           solve                     343.26 17       20.19  68.54
HamiltLCAO            updateHk                  120.31 17       7.08   24.02
OperatorLCAO          init                      119.59 51       2.34   23.88
Veff                  contributeHR              118.31 17       6.96   23.62
Gint_interface        cal_gint                  320.96 35       9.17   64.09
Gint_interface        cal_gint_vlocal           114.33 17       6.73   22.83
Gint_Tools            cal_psir_ylm              82.79  272000   0.00   16.53
Gint_k                transfer_pvpR             3.98   17       0.23   0.79
Overlap               calculate_SR              0.67   1        0.67   0.13
Overlap               contributeHk              0.06   17       0.00   0.01
Ekinetic              contributeHR              0.67   17       0.04   0.13
Ekinetic              calculate_HR              0.67   1        0.67   0.13
Nonlocal           contributeHR              0.49   17       0.03   0.10
Nonlocal           calculate_HR              0.45   1        0.45   0.09
OperatorLCAO          contributeHk              0.09   17       0.01   0.02
HSolverLCAO           hamiltSolvePsiK           90.50  17       5.32   18.07
ElecStateLCAO         psiToRho                  132.45 17       7.79   26.45
elecstate             cal_dm                    2.02   18       0.11   0.40
psiMulPsiMpi          pdgemm                    2.01   18       0.11   0.40
DensityMatrix         cal_DMR                   0.13   18       0.01   0.03
Gint                  transfer_DMR              2.98   17       0.18   0.60
Gint_interface        cal_gint_rho              127.22 17       7.48   25.40
Charge_Mixing         get_drho                  0.07   17       0.00   0.01
Charge                mix_rho                   0.74   16       0.05   0.15
Charge                Pulay_mixing              0.45   16       0.03   0.09
ESolver_KS_LCAO       out_deepks_labels         0.00   1        0.00   0.00
LCAO_Deepks_Interface out_deepks_labels         0.00   1        0.00   0.00
ESolver_KS_LCAO       cal_force                 124.65 1        124.65 24.89
Force_Stress_LCAO     getForceStress            124.65 1        124.65 24.89
Forces                cal_force_loc             0.62   1        0.62   0.12
Forces                cal_force_ew              0.53   1        0.53   0.11
Forces                cal_force_cc              1.67   1        1.67   0.33
Forces                cal_force_scc             1.20   1        1.20   0.24
Stress_Func           stress_loc                2.00   1        2.00   0.40
Stress_Func           stress_har                0.11   1        0.11   0.02
Stress_Func           stress_ewa                0.49   1        0.49   0.10
Stress_Func           stress_cc                 2.99   1        2.99   0.60
Stress_Func           stress_gga                0.64   1        0.64   0.13
Force_LCAO            ftable                    114.38 1        114.38 22.84
Force_LCAO            allocate                  0.00   1        0.00   0.00
LCAO_domain           build_ST_new              12.21  2        6.10   2.44
LCAO_domain           vnl_mu_new                6.91   1        6.91   1.38
Force_LCAO_k          allocate_k                0.00   1        0.00   0.00
Force_LCAO            cal_fedm                  1.59   1        1.59   0.32
```

(continues on next page)

(continued from previous page)

```
 Force_LCAO_k          cal_edm_2d                0.00   1        0.00   0.00
 Force_LCAO            cal_ftvnl_dphi            0.07   1        0.07   0.01
 Force_LCAO            cal_fvl_dphi              79.41  1        79.41  15.86
 Gint_interface        cal_gint_force            79.41  1        79.41  15.86
 Gint_Tools            cal_dpsir_ylm             40.76  8000     0.01   8.14
 Gint_Tools            cal_dpsirr_ylm            10.64  8000     0.00   2.12
 Force_LCAO            cal_fvnl_dbeta            11.73  1        11.73  2.34
 ESolver_KS_LCAO       cal_stress                0.00   1        0.00   0.00
 ESolver_KS_LCAO       after_all_runners         0.02   1        0.02   0.00
 ModuleIO              write_istate_info         0.02   1        0.02   0.00
----------------------------------------------------------------------------
// The breakdown of time statistics for various components of the ABACUS calculation.
```

MEMORY STATISTICS

```
NAME-------------------------|MEMORY(MB)--------
// A summary of the memory consumption by different parts of the calculation.
                         total      1473.1712
                  Stress::dSR       164.4884
                 Gint::hRGint       116.0392
                Gint::DMRGint       115.7267
                 pvpR_reduced       114.6401
                    LOC::DM_R       114.6401
       TwoCenterTable: Kinetic        91.1862
       TwoCenterTable: Overlap        91.1862
                  Force::dS_K        82.2442
                  Stress::dHr        82.2442
                 Force::dTVNL        82.2442
                    FFT::grid        54.4922
      TwoCenterTable: Nonlocal        42.0997
               HamiltLCAO::hR         28.1311
            DensityMatrix::DMR        28.1311
                  SF::strucFac        25.7154
              LOC::wfc_k_grid         21.6797
              GT::ind_bigcell         15.0464
         GT::in_this_processor        15.0464
               GT::index2normal        15.0464
               GT::index2ucell         15.0464
                HamiltLCAO::sR         14.9202
                      Chg::rho         13.1836
                 Chg::rho_save         13.1836
                 Chg::rho_core         13.1836
                 Pot::veff_fix         13.1836
                     Pot::veff         13.1836
              Pot::veff_smooth         13.1836
            DensityMatrix::DMK         12.0322
                      Chg::rhog          6.4288
                 Chg::rhog_save          6.4288
                 Chg::rhog_core          6.4288
                   meshball_pos          6.2642
```

(continues on next page)

(continued from previous page)

```
      GT::bigcell_on_processor         3.7616
                 RealGauntTable         3.6165
                 GT::which_atom         3.1321
              GT::which_bigcell         3.1321
             GT::which_unitcell         3.1321
                   PW_B_K::gcar         2.4168
                   SF::eigts123         2.1097
                 Grid::AtomLink         1.2822
                     index_ball         1.0440
 -------------   < 1.0 MB has been ignored ----------------
 ---------------------------------------------------------
// The list of memory allocations for various components, indicating the memory usage␣
↪efficiency.
```

Start and end times

```
Start  Time  : Thu Jul 18 11:34:56 2024
Finish Time  : Thu Jul 18 11:43:20 2024
Total  Time  : 0 h 8 mins 24 secs
// The start and end times of the calculation, along with the total duration.
```

13.3 Outputting Dipole Moment

ABACUS can output the dipole moment by adding the keyword `out_dipole` in the INPUT file:

```
out_dipole              1
```

13.3.1 Supported Calculations

This feature is available for all types of DFT calculations that use charge density:

- **KSDFT** (Kohn-Sham DFT)
  - Plane wave (PW) basis
  - Linear combination of atomic orbitals (LCAO) basis
- **SDFT** (Stochastic DFT)
- **OFDFT** (Orbital-Free DFT)
- **TDDFT** (Time-Dependent DFT)

13.3.2 Input Parameters

Parameter	Type	Description	Default
`out_dipole`	Integer	Whether to output dipole moment	0

- `out_dipole = 0`: Disable dipole output
- `out_dipole = 1`: Enable dipole output

13.3.3 Output Files

When `out_dipole` is set to 1, ABACUS will generate files named `dipole_s${spin}.txt` for each spin channel in the output directory.

For spin-polarized calculation (nspin=2):

- `dipole_s1.txt`
- `dipole_s2.txt`

For non-spin-polarized calculation (nspin=1):

- `dipole_s1.txt`

13.3.4 Output Format

The dipole output file contains one line for each ionic/electronic step:

```
step_index dipole_x dipole_y dipole_z
```

- `step_index`: The current step number (starts from 1)
- `dipole_x, dipole_y, dipole_z`: The x, y, z components of the dipole moment

Example output:

```
1 -0.00123456 0.00234567 -0.00345678
2 -0.00123457 0.00234568 -0.00345679
...
```

13.3.5 Additional Information

During the calculation, the dipole moment is also printed in the `running_*.log` file, including:

- Electronic dipole moment
- Ionic dipole moment
- Total dipole moment
- Total dipole moment norm

The dipole moment calculation includes both electronic and ionic contributions. The total dipole moment is the sum of electronic and ionic dipoles.

CHAPTER

FOURTEEN

INTERFACES TO OTHER SOFTWARES

14.1 PyABACUS

PyABACUS is the official Python interface for ABACUS, providing a convenient way to run DFT calculations directly from Python scripts.

14.1.1 Installation

From Source (Recommended)

```
cd /path/to/abacus-develop/python/pyabacus
pip install -e .
```

With C++ Driver Support

For full functionality including direct library calls (faster than subprocess), build ABACUS with Python bindings:

```
cmake -B build -DENABLE_PYABACUS=ON -DENABLE_LCAO=ON
cmake --build build -j8
pip install -e python/pyabacus
```

Note: The `pyabacus` package on PyPI is a different project and is NOT related to ABACUS. Please install from source as shown above.

14.1.2 Quick Start

Basic SCF Calculation

```
import pyabacus

# Run calculation from a directory containing INPUT, STRU, KPT files
result = pyabacus.abacus("./Si_scf/")

# Check results
print(f"Converged: {result.converged}")
print(f"Total energy: {result.etot_ev:.6f} eV")
print(result.summary())
```

Calculate Forces and Stress

```
result = pyabacus.abacus(
    "./Si_relax/",
    calculate_force=True,
    calculate_stress=True,
)

# Access forces (in eV/Angstrom)
if result.has_forces:
    forces = result.forces_ev_ang
    print(f"Max force: {forces.max():.6f} eV/Ang")

# Access stress tensor (in kbar)
if result.has_stress:
    print(f"Stress tensor:\n{result.stress}")
```

Parallel Calculation

```
# Run with MPI and OpenMP parallelization
result = pyabacus.abacus(
    "./Si_scf/",
    nprocs=4,       # 4 MPI processes (mpirun -np 4)
    nthreads=2,     # 2 OpenMP threads (OMP_NUM_THREADS=2)
)
```

This is equivalent to running:

```
OMP_NUM_THREADS=2 mpirun -np 4 abacus
```

Silent Mode

```
# Run without output
result = pyabacus.abacus("./Si_scf/", verbosity=0)
```

14.1.3 API Reference

pyabacus.abacus()

Main function for running ABACUS calculations.

```
def abacus(
    input_dir: str = None,
    *,
    input_file: str = None,
    stru_file: str = None,
    kpt_file: str = None,
    pseudo_dir: str = None,
    orbital_dir: str = None,
    output_dir: str = None,
    calculate_force: bool = True,
    calculate_stress: bool = False,
```

(continues on next page)

(continued from previous page)

```
    verbosity: int = 1,
    nprocs: int = 1,
    nthreads: int = 1,
) -> CalculationResult
```

Parameters:

Parameter	Type	Default	Description
`input_dir`	str	`"."`	Directory containing INPUT, STRU, KPT files
`input_file`	str	None	Explicit path to INPUT file
`stru_file`	str	None	Explicit path to STRU file
`kpt_file`	str	None	Explicit path to KPT file
`pseudo_dir`	str	None	Directory containing pseudopotentials
`orbital_dir`	str	None	Directory containing orbital files (LCAO)
`output_dir`	str	`"OUT.PYABACUS"`	Directory for output files
`calculate_force`	bool	True	Whether to calculate forces
`calculate_stress`	bool	False	Whether to calculate stress tensor
`verbosity`	int	1	Output level (0=silent, 1=normal, 2=verbose)
`nprocs`	int	1	Number of MPI processes
`nthreads`	int	1	Number of OpenMP threads

Returns: `CalculationResult` object

`CalculationResult`

Container for calculation results.

Attributes:

Attribute	Type	Description
`converged`	bool	Whether SCF converged
`niter`	int	Number of SCF iterations
`drho`	float	Final charge density difference
`etot`	float	Total energy (Ry)
`etot_ev`	float	Total energy (eV)
`forces`	ndarray	Forces on atoms (nat, 3) in Ry/Bohr
`forces_ev_ang`	ndarray	Forces in eV/Angstrom
`stress`	ndarray	Stress tensor (3, 3) in kbar
`fermi_energy`	float	Fermi energy (eV)
`bandgap`	float	Band gap (eV)
`nat`	int	Number of atoms
`ntype`	int	Number of atom types
`nbands`	int	Number of bands
`nks`	int	Number of k-points
`output_dir`	str	Path to output directory (OUT.$suffix)
`log_file`	str	Path to the main log file
`output_files`	dict	Dictionary of output files (filename -> path)

Methods:

Method	Description
`summary()`	Return a formatted summary string
`energies`	Dictionary of all energy components
`has_forces`	Whether forces are available
`has_stress`	Whether stress is available
`has_output_dir`	Whether output directory exists
`get_output_file(name)`	Get full path to specific output file
`list_output_files()`	List all output file names

14.1.4 Output File Tracking

PyABACUS automatically tracks output files generated during calculations:

```
result = pyabacus.abacus("./Si_scf/")

# Check output directory
print(f"Output directory: {result.output_dir}")
print(f"Log file: {result.log_file}")

# List all output files
print("Output files:")
for filename in result.list_output_files():
    print(f"  {filename}")

# Get path to specific file
bands_file = result.get_output_file("BANDS_1.dat")
if bands_file:
    # Read and process band structure data
    import numpy as np
    bands = np.loadtxt(bands_file)
```

Common Output Files

File	Description
`running_scf.log`	Main calculation log
`BANDS_1.dat`	Band structure data
`PDOS`	Projected density of states
`CHARGE.cube`	Charge density in cube format
`SPIN1_CHG.cube`	Spin-up charge density
`SPIN2_CHG.cube`	Spin-down charge density
`istate.info`	Band eigenvalues and occupations
`kpoints`	K-point information

14.1.5 Convenience Functions

`run_scf()`

Alias for `abacus()` with default SCF parameters:

```
result = pyabacus.run_scf("./Si_scf/")
```

run_relax()

Alias for `abacus()` with force calculation enabled:

```
result = pyabacus.run_relax("./Si_relax/")
```

14.1.6 Examples

Energy vs. Lattice Constant

```
import pyabacus
import numpy as np

lattice_constants = np.linspace(5.0, 5.5, 11)
energies = []

for a in lattice_constants:
    # Modify STRU file with new lattice constant
    # ... (file modification code)

    result = pyabacus.abacus("./Si_eos/", verbosity=0)
    energies.append(result.etot_ev)

# Plot equation of state
import matplotlib.pyplot as plt
plt.plot(lattice_constants, energies, 'o-')
plt.xlabel("Lattice constant (Ang)")
plt.ylabel("Energy (eV)")
plt.savefig("eos.png")
```

Parallel Batch Calculations

```
import pyabacus
from pathlib import Path

# Run calculations for multiple systems with parallelization
systems = ["Si", "Ge", "C"]
results = {}

for system in systems:
    input_dir = Path(f"./{system}_scf/")
    if input_dir.exists():
        result = pyabacus.abacus(
            str(input_dir),
            nprocs=4,
            nthreads=2,
        )
        results[system] = {
            "energy": result.etot_ev,
            "converged": result.converged,
            "bandgap": result.bandgap,
        }
```

(continues on next page)

(continued from previous page)

```
# Print summary
for system, data in results.items():
    print(f"{system}: E={data['energy']:.4f} eV, gap={data['bandgap']:.2f} eV")
```

14.1.7 Troubleshooting

ABACUS executable not found

If you see "ABACUS executable not found", ensure:

1. ABACUS is installed and in your PATH
2. Or build with C++ driver support (see Installation)

MPI not found

If you see "mpirun/mpiexec not found" when using `nprocs > 1`:

1. Install MPI (OpenMPI or MPICH)
2. Ensure `mpirun` or `mpiexec` is in your PATH
3. Or set `nprocs=1` to run without MPI

Import errors

If `import pyabacus` fails:

1. Ensure pyabacus is installed: `pip install pyabacus`
2. Check Python version compatibility (Python 3.8+)

Calculation not converging

Check the log file for details:

```
result = pyabacus.abacus("./problem_case/")
if not result.converged:
    log_path = result.get_output_file("running_scf.log")
    if log_path:
        with open(log_path) as f:
            print(f.read()[-2000:])  # Print last 2000 chars
```

Forces or stress not available

Forces and stress are parsed from the ABACUS output log. Ensure:

1. `cal_force` is set in your INPUT file for forces
2. `cal_stress` is set in your INPUT file for stress
3. The calculation completed successfully

14.2 DeePKS

DeePKS is a machine-learning (ML) 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. Common high-precision methods include hybrid functionals or CCSD-T, while common low-precision methods are LDA/GGA.

As such, the trained DeePKS model can provide highly accurate energetics (and forces/band gap/density) 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 J. Phys. Chem. A 126.49 (2022): 9154-9164 for the application of DeePKS in periodic systems.

Detailed instructions on installing and running DeePKS can be found on this website. The DeePKS-related keywords in `INPUT` file can be found here. An example for training DeePKS model with ABACUS is also provided. For practical applications, users can refer to a series of Notebooks. These Notebooks provide detailed instructions on how to train and use the DeePKS model using perovskite as an example. Currently, these tutorials are available in Chinese, but we plan to release corresponding English versions in the near future.

Note: DeePKS calculations can only be performed by the LCAO basis.

14.3 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.

14.3.1 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],
    "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,
                    40
                ],
                "resnet_dt": true,
                "axis_neuron": 12,
                "seed": 1
            },
            "fitting_net": {
                "neuron": [
                    25,
                    50,
                    100
                ],
                "resnet_dt": false,
                "seed": 1
            }
```

(continues on next page)

(continued from previous page)

```
        },
        "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
            ],
            "press": [
                1.0
            ],
            "trj_freq": 10,
            "nsteps": 300,
            "ensemble": "nvt",
            "_idx": "00"
        },
        {
            "sys_idx": [
```

(continues on next page)

(continued from previous page)

```
                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": "broyden",
    "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.
    }
}
```

14.3.2 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/"],
```

(continues on next page)

(continued from previous page)

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

(continues on next page)

(continued from previous page)

```
        ]
}
```

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",
          "password": "xxx",
```

(continues on next page)

(continued from previous page)

```
        "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,
```

(continues on next page)

(continued from previous page)

```
        "gpu_per_node": 0,
        "queue_name": "CPU",
        "group_size": 2,
        "local_root":"./",
        "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.

14.4 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
```

For ABACUS v3.9.0.25+: Files named `hrs1_nao.csr`, `hrs2_nao.csr` (for nspin=2), and `srs1_nao.csr` will be generated in `OUT.${suffix}/` directory, containing the Hamiltonian and overlap matrices in standard CSR format. You can optionally specify precision: `out_mat_hs2 1 8` (default 8 digits).

For ABACUS v3.8.x and earlier: Files named `data-HR-sparse_SPIN${x}.csr` and `data-SR-sparse_SPIN${x}.csr` will be generated, where `${x}` takes value of 0 or 1 based on the spin component.

> **Note:** DeepH v1.0.0+ is required to read the new CSR format from ABACUS v3.9.0.25+. For older DeepH versions, please use ABACUS v3.8.x or earlier.

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.

14.5 DeePTB

DeePTB is an innovative Python package that uses deep learning to accelerate ab initio electronic structure simulations. It offers versatile, accurate, and efficient simulations for a wide range of materials and phenomena. Trained on small systems, DeePTB can predict electronic structures of large systems, handle structural perturbations, and integrate with molecular dynamics for finite temperature simulations, providing comprehensive insights into atomic and electronic behavior. See more details in DeePTB-SK: Nat Commun 15, 6772 (2024) and DeePTB-E3: arXiv:2407.06053.

DeePTB trains the model based on the Structure, Eigenvalues, Hamiltonian, Density matrix, and Overlap matrix from first-principles calcualtions. DeePTB team provides the interfaces dftio with other first-principles softwares. dftio fully supports the interfaces with ABACUS, and can transfer the Structure, Eigenvalues, Hamiltonian, Density matrix, and Overlap matrix from ABACUS into the format used in DeePTB.

14.6 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`.

14.7 Phonopy

Phonopy is a powerful package to calculate phonon and related properties. The ABACUS interface has been added in Phonopy v.2.19.1. In the following, we take the FCC aluminum as an example:

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

```
phonopy -d --dim="2 2 2" --abacus
```

2. 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
```

3. Then create 'FORCE_SETS' file using ABACUS inteface:

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

4. 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.
```

14.8 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 suitable for nspin=1, 2, 4 (including lspinorb=1).

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_pw/` 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
   ```

(continues on next page)

(continued from previous page)

```
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
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

pseudo_dir              ../../../tests/PP_ORB
orbital_dir             ../../../tests/PP_ORB
ntype                   1
ecutwfc                 50
```

(continues on next page)

(continued from previous page)

```
nbands                  4
smearing_method         fixed
calculation             nscf
scf_nmax                50
pw_diag_thr             1.0e-12
scf_thr                 1.0e-13
init_chg                file
symmetry                -1
towannier90             1
nnkpfile                diamond.nnkp
basis_type              pw
out_wannier_unk         0
```

There are seven 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.
- *out_wannier_mmn*: control whether to output the "*.mmn" file.
- *out_wannier_amn*: control whether to output the "*.amn" file.
- *out_wannier_eig*: control whether to output the "*.eig" file.
- *out_wannier_unk*: control whether to output the "UNK.*" file.
- *out_wannier_wvfn_formatted*: control what format of the Wannier function file to output, `true`: output the formatted text file; `false`: output the binary file. Note that the `wvfn_formatted` option in `*.win` file (input file of Wannier90) has to be set accordingly with this option.

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. You also have to generate the `diamond.amn` and `UNK` files for the plot. Otherwise, the two types of file are not necessary.

14.9 ASE

14.9.1 Introduction

ASE (Atomic Simulation Environment) performs as a powerful Pythonic platform for atomistic simulations, in which there are plenty of functionalties supported, such as various geometry optimization algorithms for finding both the minimum energy point and the transition states, including BFGS, BFGSLineSearch, FIRE, NEB, AUTO-NEB, etc, and various molecular dynamics techniques, including thermostats (Langevin, CSVR, Nose-Hoover Chain, etc) and metadynamics (via the interface with Plumed).

Due to the growing number of softwares and machine-learning forcefields, we turn to maintain the interface with ASE by our own, while a legacy version of ASE interface can still be found at our GitLab repository of ase-abacus.

14.9.2 Installation

We strongly recommend you create a virtual environment for the installation of Python packages of abacus, such as `conda` or `venv`, to avoid conflicts with other packages, for example, with the `conda`:

```
conda create -n abacus python=3.10
conda activate abacus
```

Then, install the ASE interface by:

```
cd interfaces/ASE_interface
pip install .
```

14.9.3 ABACUS Calculator

Present calculator implementation requires a "profile" to act as an interface between the Python runtime and the file system. Instantiate an `AbacusProfile` object with proper settings:

```
from abacuslite import AbacusProfile
aprof = AbacusProfile(
    command='mpirun -np 4 abacus',
    omp_num_threads=1,
    pseudo_dir='/path/to/folder/of/pseudopotentials',
    orbital_dir='/path/to/folder/of/orbitals', # OPTIONAL!
)
```

, by such lines, you build the interface between the computational environment and the Python runtime. This interface can be reused in multiple calculations.

Then, you can instantiate the `Abacus` calculator with the profile by:

```
from abacuslite import Abacus
abacus = Abacus(
    profile=aprof,
    directory='/path/to/work/directory',
    pseudopotentials={
        'Si': 'Si_ONCV_PBE-1.0.upf',
```

(continues on next page)

(continued from previous page)

```
    },
    basissets={
        'Si': 'Si_gga_8au_100Ry_2s2p1d.orb',
    },
    inp={
        'calculation': 'scf',
        'nspin': 1,
        'basis_type': 'lcao',
        'ks_solver': 'genelpa',
        'ecutwfc': 100,
        'symmetry': 1,
        'kspacing': 0.1
    }
)
```

, where except the `directory`, you can focus on the setting of ABACUS itself. In `inp`, you can set everything as you do in INPUT file of ABACUS. The kpoint sampling can also be set by the `kpts` parameter, like:

```
abacus = Abacus(
    # all other parameters
    kpts={
        'mode': 'mp-sampling',
        'gamma-centered': True,
        'nk': (4, 4, 4),
        'kshift': (0, 0, 0),
    }
)
```

If with the `tempfile` module, you can create an abacus instance whose directory will be automatically removed when leaves from the context:

```
import tempfile
with tempfile.TemporaryDirectory() as tmpdir:
    abacus = Abacus(
        profile=aprof,
        directory=tmpdir,
        pseudopotentials={
            'Si': 'Si_ONCV_PBE-1.0.upf',
        },
        basissets={
            'Si': 'Si_gga_8au_100Ry_2s2p1d.orb',
        },
        inp={
            # the rest of input parameters
        }
    )
```

14.9.4 Perform Calculations

In the new implementation, we limit the range of functionalties supported to mainly include the necessary ones, such as the SCF calculation, the energy and force/stress evaluation. The other features, such as starting the molecule dynamics directly in ABACUS from Python, is not supported anymore. Instead, it is encouraged to use the ASE tools to perform the molecule dynamics.

Please read the examples in `interfaces/ASE_interface/examples/` for more details.

14.9.5 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`.

14.10 PYATB

14.10.1 Introduction

PYATB (Python ab initio tight binding simulation package) is an open-source software package designed for computing electronic structures and related properties based on the ab initio tight binding Hamiltonian. The Hamiltonian can be directly obtained after conducting self-consistent calculations with ABACUS using numerical atomic orbital (NAO) bases. The package comprises three modules - Bands, Geometric, and Optical, each providing a comprehensive set of tools for analyzing different aspects of a material's electronic structure.

14.10.2 Installation

```
git clone https://github.com/pyatb/pyatb.git
cd pyatb
python setup.py install --record log
```

To customize the `setup.py` file, you must make changes to the **CXX** and **LAPACK_DIR** variables in line with your environment. **CXX** denotes the C++ compiler you intend to use, for instance, icpc (note that it should not be the mpi version). Furthermore, **LAPACK_DIR** is used to specify the Intel MKL path.

After completing the installation process, you can access the `pyatb` executable and corresponding module, which can be imported using the `import pyatb` command.

14.10.3 How to use

We take Bi_2Se_3 as an example to illustrate how to use ABACUS to generate the tight binding Hamiltonian required for PYATB, and then perform calculations related to PYATB functions.

1. Perform ABACUS self consistent calculation:

```
INPUT_PARAMETERS

# System variables
suffix                  Bi2Se3
ntype                   2
calculation             scf
esolver_type            ksdft
symmetry                1
init_chg                atomic
```

(continues on next page)

(continued from previous page)

```
# Plane wave related variables
ecutwfc                100

# Electronic structure
basis_type             lcao
ks_solver              genelpa
nspin                  4
smearing_method        gauss
smearing_sigma         0.02
mixing_type            broyden
mixing_beta            0.7
scf_nmax               200
scf_thr                1e-8
lspinorb               1
noncolin               0

# Variables related to output information
out_chg                1
out_mat_hs2            1
out_mat_r              1
```

After the key parameters `out_mat_hs2` and `out_mat_r` are turned on, ABACUS will generate files containing the Hamiltonian matrix $H(R)$, overlap matrix $S(R)$, and dipole matrix $r(R)$ after completing the self-consistent calculation. These parameters can be found in the ABACUS `INPUT` file.

2. Copy the HR, SR, and rR files output by ABACUS's self-consistent calculation, which are located in the `OUT*` directory and named `data-HR-sparse_SPIN0.csr`, `data-SR-sparse_SPIN0.csr`, and `data-rR-sparse.csr`, respectively. Copy these files to the working directory and write the `Input` file for PYATB:

```
INPUT_PARAMETERS
{
    nspin                           4
    package                         ABACUS
    fermi_energy                    9.557219691497478
    fermi_energy_unit               eV
    HR_route                        data-HR-sparse_SPIN0.csr
    SR_route                        data-SR-sparse_SPIN0.csr
    rR_route                        data-rR-sparse.csr
    HR_unit                         Ry
    rR_unit                         Bohr
    max_kpoint_num                  8000
}

LATTICE
{
    lattice_constant                1.8897162
    lattice_constant_unit           Bohr
    lattice_vector
    -2.069  -3.583614  0.000000
     2.069  -3.583614  0.000000
     0.000   2.389075  9.546667
}
```

(continues on next page)

(continued from previous page)

```
BAND_STRUCTURE
{
    wf_collect                      0
    kpoint_mode                     line
    kpoint_num                      5
    high_symmetry_kpoint
    0.00000 0.00000 0.0000 100  # G
    0.00000 0.00000 0.5000 100  # Z
    0.50000 0.50000 0.0000 100  # F
    0.00000 0.00000 0.0000 100  # G
    0.50000 0.00000 0.0000 1    # L
}
```

For specific input file writing, please refer to PYATB's quick start.

3. Perform PYATB calculation:

```
export OMP_NUM_THREADS=2
mpirun -np 6 pyatb
```

After the calculation is completed, the band structure data and figures of Bi_2Se_3 can be found in the `Out/Band_Structure` folder.

14.11 ShengBTE

14.11.1 Introduction

This tutorial aims to introduce the process of performing density functional theory calculations using ABACUS and calculating lattice thermal conductivity using the ShengBTE software. During the entire calculation process, the following external software are also used: 1) Phonopy for calculating second-order force constants, 2) ASE for converting atomic structures, 3) ShengBTE's thirdorder program for calculating third-order force constants, and 4) finally using ShengBTE to calculate the material's lattice thermal conductivity.

Here is the announcement of ShengBTE with ABACUS: ShengBTE - The ABACUS DFT software can now be used with ShengBTE

Some external packages that need to be combined are mentioned above, and here it is recommended to read the relevant documentation and instructions of these packages:

ShengBTE：https://bitbucket.org/sousaw/shengbte/src/master/

phonopy：http://abacus.deepmodeling.com/en/latest/advanced/interface/phonopy.html

ASE：http://abacus.deepmodeling.com/en/latest/advanced/interface/ase.html

thirdorder: https://bitbucket.org/sousaw/thirdorder/src/master/

14.11.2 Prepare

The ABACUS software package provides an example of using ABACUS+ShengBTE to calculate the lattice thermal conductivity in the `examples/interface_ShengBTE` folder. The example includes two folders: `LCAO` (Linear Combination of Atomic Orbitals) which uses numerical atomic orbitals and `PW` (Plane wave) which uses plane wave basis vectors. Each folder contains three subfolders: `2nd`, `3rd`, and `shengbte`, which respectively store the relevant files for calculating second-order force constants using Phonopy (`2nd`), third-order force constants using the thirdorder program (`3rd`), and lattice thermal conductivity using ShengBTE (`shengbte`).

14.11.3 How to use

Taking the `LCAO` folder as an example, we provide the test case of a diamond structure Si structure containing 2 atoms with the norm-conserving pseudopotential `Si_ONCV_PBE-1.0.upf` and the atomic orbital file `Si_gga_7au_100Ry_2s2p1d.orb` (GGA functional, 7 a.u. cut-off radius, 100 Ry energy cut-off, and DZP orbitals containing 2s2p1d).

1. Calculating the second-order force constants

It would be best to combine Phonopy and ASE with ABACUS to calculate the second-order force constants. First, enter the `2nd` folder.

1.1 Structure optimization

Before performing lattice thermal conductivity calculations, it is necessary to optimize the atomic configuration of the simulated material system. The following is the atomic configuration file `STRU` obtained after structure optimization (relax) using ABACUS. In this example, for simplicity, a 2*2*2 Brillouin zone k-point sampling was used in the structure optimization process, with an energy cutoff value of 100 Ry for plane waves (the plane wave basis vector is also used in LCAO). Note that the actual calculation should use more convergent k-point sampling.

```
ATOMIC_SPECIES
Si 28.0855 Si_ONCV_PBE-1.0.upf

NUMERICAL_ORBITAL
Si_gga_7au_100Ry_2s2p1d.orb

LATTICE_CONSTANT
1.88972612546

LATTICE_VECTORS
0 2.81594778072 2.81594778072 #latvec1
2.81594778072 0 2.81594778072 #latvec2
2.81594778072 2.81594778072 0 #latvec3

ATOMIC_POSITIONS
Direct # direct coordinate

Si #label
0 #magnetism
2 #number of atoms
0.875  0.875  0.875  m  0  0  0
0.125  0.125  0.125  m  0  0  0
```

1.2 Calculating the second-order force constants

The Phonopy software is called to generate multiple atomic configurations of the supercell and corresponding perturbations needed for calculation with the following command:

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

where the `setting.conf` file reads:

```
DIM = 2 2 2
ATOM_NAME = Si
```

In this Si example, we only need to generate one perturbed atomic configuration, `STRU-001`. Perform SCF calculations (SCF stands for Self-Consistent Field and represents the iterative self-consistent calculation of density functional theory) on all perturbed configurations (in this case, there is only one for Si) to obtain the forces on the atoms. Afterward, use the following command to generate the `FORCE_SET` file:

```
phonopy -f OUT.DIA-50/running_scf.log
```

Tip: In the input file `INPUT` of ABACUS, you can set the variable `stru_file`, which corresponds to the atomic configuration file `STRU-001`, and ABACUS will read the structure file directly.

Next, set the `band.conf` file to calculate the phonon spectrum and the second-order force constants:

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

The `band.conf` file mentioned here contains the following contents (you can refer to the Phonopy documentation for specific parameter meanings):

```
ATOM_NAME = Si
DIM = 2 2 2
MESH = 8 8 8
PRIMITIVE_AXES = 1 0 0 0 1 0 0 0 1
BAND = 0.0 0.0 0.0  0.5 0.0 0.5  0.625  0.25  0.625, 0.375 0.375 0.75  00 0.0 0.0  0.
↪5 0.5 0.5
BAND_POINTS = 101
BAND_CONNECTION = .TRUE.
FORCE_CONSTANTS = WRITE
FULL_FORCE_CONSTANTS = .TRUE.
```

After this step, the Phonopy software will generate `band.yaml` (for plotting the phonon spectrum) and the `FORCE_CONSTANTS` file. The data contained in the `FORCE_CONSTANTS` file is the second-order force constants. It is important to set `FULL_FORCE_CONSTANTS = .TRUE.`, which outputs all the second-order force constants. Otherwise, there may be errors when ShengBTE reads the data.

In addition, you can use the following command to output the gnuplot format of the phonon spectrum for plotting:

```
phonopy-bandplot --gnuplot > pho.dat
```

1.3 Post-processing

Note that ShengBTE software requires the unit of the data in the `FORCE_CONSTANTS_2ND` file to be eV/Å^2, but the unit of the `FORCE_CONSTANTS` calculated by ABACUS combined with Phonopy is eV/(Å*au), where au is the atomic unit system and 1 au = 0.52918 Å. You can use the provided `au2si.py` script in the `2nd` directory to convert the units and generate the `FORCE_CONSTANTS_2ND` file. The command is as follows:

```
python au2si.py
```

The `FORCE_CONSTANTS_2ND` file is provided in the shengbte folder for reference to the calculation results.

2. Calculating the third-order force constants

To calculate the third-order force constants, you need to combine with the thirdorder program and output the third-order force constant file `FORCE_CONSTANTS_3RD`. However, thirdorder currently only supports reading input and output files from VASP and QE. Therefore, we are using thirdorder by converting ABACUS's structure and output force files to `POSCAR` and `vasprun.xml`, respectively. Please enter the `3rd` folder first, and the specific steps will be described below.

2.1 Obtaining perturbed configurations

First, convert the optimized `STRU` file from ABACUS software to `POSCAR` (the converted `POSCAR` file is already provided in the directory, or you can do this conversion by yourself).

Then, run the `thirdorder_vasp.py` program to generate a series of atomic configuration files `3RD.POSCAR.*` after perturbation. For example, in this example, a total of 40 configurations were generated:

```
thirdorder_vasp.py sow 2 2 2 -2
```

Run `pos2stru.py` to convert the above `POSCAR` to `STRU` file. Note that this script calls functions from the ASE software package (ASE needs to be installed in advance):

```
python pos2stru.py
```

Note: The dpdata software cannot be called here to perform the conversion. This is because the dpdata software forces the lattice to change into a lower triangular matrix, which is equivalent to rotating the lattice and leads to a corresponding rotation in the direction of the interatomic forces, which will cause errors.

2.2 Calculation of atomic forces for perturbation configurations

You can refer to the `run_stru.sh` script provided in the directory to batch generate `SCF-*` folders and submit calculations. Here, ABACUS needs to perform SCF calculations on 40 atomic configurations, which may take some time. It is recommended to run each SCF separately in the `SCF-*` folder. The `scf_thr` parameter in INPUT file should be set to at least 1e-8 to obtain converged results.

After the calculations are complete, run `aba2vasp.py` to package the atomic forces calculated by ABACUS into the `vasprun.xml` format and place them in each `SCF-\*` folder with the following command:

```
python aba2vasp.py
```

The `vasprun.xml` format is illustrated as follows:

```
<modeling>
    <calculation>
        <varray name="forces">
            <v>1.865e-05 -0.04644196 -0.00153852</v>
            <v>-1.77e-05 -0.00037715 -0.00149635</v>
            <v>1.973e-05 0.002213 -0.00149461</v>
            <v>-1.976e-05 0.00065303 -0.0014804</v>
            <v>8.31e-06 -0.0003306 -0.00024288</v>
            <v>-8.25e-06 -0.00038306 -0.00025385</v>
            <v>1.071e-05 0.00060621 -0.00025797</v>
            <v>-1.05e-05 -0.00014553 -0.00027532</v>
            <v>0.00668053 0.00645634 -0.04642593</v>
            <v>-0.00668085 0.00645595 -0.00040122</v>
            <v>-0.00650454 0.00628877 -0.00025123</v>
            <v>0.00650504 0.00628892 -0.00028948</v>
            <v>-0.00039591 2.479e-05 0.00223371</v>
            <v>0.00039608 2.426e-05 0.0006732</v>
            <v>0.0003264 3.122e-05 0.00052874</v>
            <v>-0.00032589 3.415e-05 -0.00023577</v>
            <v>-2.908e-05 -0.00832477 0.00635709</v>
            <v>3.737e-05 -0.00125057 -7.444e-05</v>
            <v>-2.582e-05 0.00656076 0.00636285</v>
            <v>2.566e-05 -0.00049974 -6.661e-05</v>
```

(continues on next page)

(continued from previous page)

```
            <v>-5.431e-05 0.00502637 0.00639077</v>
            <v>4.553e-05 -0.00180978 0.0001325</v>
            <v>-3.609e-05 -0.00676473 0.00638092</v>
            <v>3.806e-05 5.503e-05 0.00012759</v>
            <v>-0.00670704 0.00646596 0.01310437</v>
            <v>0.00670119 3.673e-05 0.00602948</v>
            <v>0.00036366 0.00627899 -0.00657272</v>
            <v>-0.00036508 2.288e-05 0.00026009</v>
            <v>0.00648649 0.0064463 -0.00036521</v>
            <v>-0.00648098 1.594e-05 0.00671469</v>
            <v>-0.00034493 0.00630074 0.00662932</v>
            <v>0.00034331 4.157e-05 -0.0002028</v>
        </varray>
    </calculation>
</modeling>
```

Finally, execute the following command:

```
find SCF-* -name vasprun.xml|sort -n|thirdorder_vasp.py reap 2 2 2 -2
```

Then, the third-order force constant file `FORCE_CONSTANTS_3RD` can be obtained by running the above command. The `FORCE_CONSTANTS_3RD` file is provided in the shengbte folder for reference in calculating the results.

3. Run ShengBTE to obtain lattice thermal conductivity

Enter the `shengbte` folder, in which the three files `CONTROL` (parameter file of ShengBTE), `FORCE_CONSTANTS_2ND` (second-order force constant file), and `FORCE_CONSTANTS_3RD` (third-order force constant file) have been prepared. Next, run ShengBTE with the following command to obtain the lattice thermal conductivity, where the calculation results are given in the Ref folder for reference:

```
mpirun -n 10 ShengBTE
```

14.11.4 Conclusion

For using plane wave (PW) in ABACUS to perform ShengBTE calculations, similar procedures should be followed. However, the `scf_thr` parameter in the `INPUT` file for calculating the third-order force constant needs to be set to at least 1e-12. The experimental lattice thermal conductivity of Si at 300 K is around 150 W/(m K), while the calculated thermal conductivity of Si at 300 K is around 100 W/(m K) by using the provided example. This is because, as a demo, a 2*2*2 expanded cell and a 2*2*2 K-point are used in the example, but the results are not converged yet with respect to the given system size and k-points. In actual research, the size of the supercell and the sampling scheme of K-points need to be tested to obtain converged results.

14.12 CANDELA

CANDELA is short for Collection of ANalysis DEsigned for Large-scale Atomic simulations. It is developed by MCresearch to conduct analyses on MD trajectory in different formats. Right now the program only supports analysis of pair distribution function (PDF), static structure factor (SSF) and mean square displacement (MSD). The minimum supported version of ABACUS is 3.2.0.

14.12.1 Requirements for using CANDELA

For Detailed usage of CANDELA, please refer to the official document. There are two things which need special attention in using CANDELA with ABACUS. First, the input file of CANDELA only takes the name of `INPUT`, the same as ABACUS input file, so you should not run CANDELA in the same folder where you run ABACUS. Second, to use CANDELA to postprocess ABACUS MD trajectory, the following parameters have to be specified in the `INPUT` file of CANDELA in addition to other required parameters:

1. `geo_in_type` has to be set to `ABACUS`;
2. `msd_dt` has to be specified in unit of picosecond, especially in the case of `calculation` = `msd`;
3. `geo_directory` has to be set to the path to the `MD_dump` file in the `OUT.xxx` folder. As a result, a CANDELA `INPUT` file for calculating PDF from ABACUS should be something like this:

```
calculation  pdf # Pair Distribution Function.
system Al
geo_in_type  LAMMPS
geo_directory ../geo/Al64.dump
geo_1        0
geo_2        20
geo_interval 1
geo_ignore   4

geo_out      pdf.txt # output pdf name.

ntype        1        # number of different types of atoms.
natom        64   # total number of atoms.
natom1       64
rcut         6
dr           0.01     # delta r in real space

struf_dgx   0.05
struf_ng    200
```

More examples of CANDELA `INPUT` with ABACUS can be found in the test directory.

14.13 TB2J

14.13.1 Introduction

TB2J is an open-source Python package for the automatic computation of magnetic interactions (including exchange and Dzyaloshinskii-Moriya) between atoms of magnetic crystals from density functional Hamiltonians based on Wannier functions or linear combinations of atomic orbitals. The program is based on Green's function method with the local rigid spin rotation treated as a perturbation. The ABACUS interface has been added since TB2J version 0.8.0.

For more information, see the documentation on https://tb2j.readthedocs.io/en/latest/

14.13.2 Installation

The most easy way to install TB2J is to use pip:

```
pip install TB2J
```

You can also download TB2J from the github page, and install with:

```
git clone https://github.com/mailhexu/TB2J.git
cd TB2J
python setup.py install
```

14.13.3 The Heisenberg model

The Heisenberg Hamiltonian in TB2J contains three different parts, which are:

$$E = -\sum_{i \neq j} \left[J_{\text{iso}}^{ij} \vec{S}_i \cdot \vec{S}_j + \vec{S}_i J_{\text{ani}}^{ij} \vec{S}_j + \vec{D}_{ij} \cdot (\vec{S}_i \times \vec{S}_j) \right],$$

where J_{iso}^{ij} represents the isotropic exchange, J_{ani}^{ij} represents the symmetric anisotropic exhcange which is a 3×3 tensor with $J^{\text{ani}} = J^{\text{ani,T}}$, $\vec{D}_{ij}$ represents the Dzyaloshinskii-Moriya interaction (DMI).

> **Note:** Exchange parameters conventions for other Heisenberg Hamiltonian can be found in Conventions of Heisenberg Model.

14.13.4 How to use

With the LCAO basis set, TB2J can directly take the output and compute the exchange parameters. For the PW and LCAO-in-PW basis set, the Wannier90 interace can be used instead. In this tutorial we will use LCAO.

Collinear calculation without SOC

We take Fe as an example to illustrate how to use ABACUS to generate the input files required for TB2J.

1. Perform ABACUS calculation.

`INPUT` file:

```
INPUT_PARAMETERS
#Parameters (1.General)
suffix                          Fe
stru_file                       STRU
kpoint_file                     KPT
pseudo_dir                      ./
orbital_dir                     ./
calculation                     scf
ntype                           1
nspin                           2
symmetry                        1
noncolin                        0
lspinorb                        0

#Parameters (2.PW)
ecutwfc                         100
scf_thr                         1.0e-6
init_chg                        atomic
out_mul                         1

#Parameters (4.Relaxation)
ks_solver                       genelpa
scf_nmax                        200
```

(continues on next page)

(continued from previous page)

```
#Parameters (5.LCAO)
basis_type                      lcao
gamma_only                      0

#Parameters (6.Smearing)
smearing_method                 gauss
smearing_sigma                  0.001

#Parameters (7.Charge Mixing)
mixing_type                     broyden
mixing_beta                     0.2

# Variables related to output information
out_mat_hs2                     1
```

`STRU` file:

```
ATOMIC_SPECIES
Fe  55.845         Fe_ONCV_PBE_FR-1.0.upf

NUMERICAL_ORBITAL
Fe_gga_8au_100Ry_4s2p2d1f.orb

LATTICE_CONSTANT
1.8897261258369282

LATTICE_VECTORS
2.8660000000      0.0000000000      0.0000000000
0.0000000000      2.8660000000      0.0000000000
0.0000000000      0.0000000000      2.8660000000

ATOMIC_POSITIONS
Direct

Fe
5.0000000000
2
0.0000000000 0.0000000000 0.0000000000 1 1 1 mag 2.5
0.5000000000 0.5000000000 0.5000000000 1 1 1 mag 2.5
```

and `KPT` file:

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

After the key parameter `out_mat_hs2` is turned on, the Hamiltonian matrix $H(R)$ (in Ry) and overlap matrix $S(R)$ will be written into files in the directory `OUT.${suffix}` . In the INPUT, the line:

```
suffix                          Fe
```

specifies the suffix of the output, in this calculation, we set the path to the directory of the DFT calculation, which is the

current directory (".") and the suffix to Fe.

> **Note (ABACUS v3.9.0.25+):** Starting from ABACUS v3.9.0.25, the output format has changed to standard CSR format with filenames `hrs1_nao.csr`, `hrs2_nao.csr` (for nspin=2), and `srs1_nao.csr`. The parameter `out_mat_hs2` now supports optional precision control: `out_mat_hs2 1 8` (default 8 digits). TB2J v0.9.0+ is required to read the new format. For older TB2J versions, please use ABACUS v3.8.x or earlier.

2. Perform TB2J calculation:

```
abacus2J.py --path . --suffix Fe --elements Fe  --kmesh 7 7 7
```

This first reads the atomic structures from the `STRU` file, then reads the Hamiltonian and overlap matrices. For ABACUS v3.9.0.25+, the matrices are stored in `hrs1_nao.csr`, `hrs2_nao.csr` (nspin=2), and `srs1_nao.csr` files. For older versions, they are in `data-HR-*` and `data-SR-*` files. It also reads the fermi energy from the `OUT.Fe/running_scf.log` file.

With the command above, we can calculate the J with a $7 \times 7 \times 7$ k-point grid. This allows for the calculation of exchange between spin pairs between $7 \times 7 \times 7$ supercell. Note: the kmesh is not dense enough for a practical calculation. For a very dense k-mesh, the `--rcut` option can be used to set the maximum distance of the magnetic interactions and thus reduce the computation cost. But be sure that the cutoff is not too small.

The description of the output files in `TB2J_results` can be found in the TB2J documentation. We can find the exchange parameters with `Fe` by :

```
grep "Fe" exchange.out
```

the following contents showing the first, second and third nearest neighbor exchange parameters as 18.6873, 9.9213 and 0.8963 meV, resoectively. More equivalent exchange parameters are also shown.

```
   Fe1    Fe2    ( -1,  -1,  -1) 18.6873    (-1.433, -1.433, -1.433)  2.482
   Fe1    Fe2    ( -1,  -1,   0) 18.6867    (-1.433, -1.433,  1.433)  2.482
   Fe1    Fe2    ( -1,   0,  -1) 18.6866    (-1.433,  1.433, -1.433)  2.482
   Fe1    Fe2    ( -1,   0,   0) 18.6873    (-1.433,  1.433,  1.433)  2.482
   Fe1    Fe2    (  0,  -1,  -1) 18.6873    ( 1.433, -1.433, -1.433)  2.482
   Fe1    Fe2    (  0,  -1,   0) 18.6866    ( 1.433, -1.433,  1.433)  2.482
   Fe1    Fe2    (  0,   0,  -1) 18.6867    ( 1.433,  1.433, -1.433)  2.482
   Fe1    Fe2    (  0,   0,   0) 18.6873    ( 1.433,  1.433,  1.433)  2.482
   Fe2    Fe1    (  0,   0,   0) 18.6873    (-1.433, -1.433, -1.433)  2.482
   Fe2    Fe1    (  0,   0,   1) 18.6867    (-1.433, -1.433,  1.433)  2.482
   Fe2    Fe1    (  0,   1,   0) 18.6866    (-1.433,  1.433, -1.433)  2.482
   Fe2    Fe1    (  0,   1,   1) 18.6873    (-1.433,  1.433,  1.433)  2.482
   Fe2    Fe1    (  1,   0,   0) 18.6873    ( 1.433, -1.433, -1.433)  2.482
   Fe2    Fe1    (  1,   0,   1) 18.6866    ( 1.433, -1.433,  1.433)  2.482
   Fe2    Fe1    (  1,   1,   0) 18.6867    ( 1.433,  1.433, -1.433)  2.482
   Fe2    Fe1    (  1,   1,   1) 18.6873    ( 1.433,  1.433,  1.433)  2.482
   Fe1    Fe1    ( -1,   0,   0)  9.9213    (-2.866,  0.000,  0.000)  2.866
   Fe2    Fe2    ( -1,   0,   0)  9.9213    (-2.866,  0.000,  0.000)  2.866
   Fe1    Fe1    (  0,  -1,   0)  9.9211    ( 0.000, -2.866,  0.000)  2.866
   Fe2    Fe2    (  0,  -1,   0)  9.9211    ( 0.000, -2.866,  0.000)  2.866
   Fe1    Fe1    (  0,   0,  -1)  9.9210    ( 0.000,  0.000, -2.866)  2.866
   Fe2    Fe2    (  0,   0,  -1)  9.9210    ( 0.000,  0.000, -2.866)  2.866
   Fe1    Fe1    (  0,   0,   1)  9.9210    ( 0.000,  0.000,  2.866)  2.866
   Fe2    Fe2    (  0,   0,   1)  9.9210    ( 0.000,  0.000,  2.866)  2.866
   Fe1    Fe1    (  0,   1,   0)  9.9211    ( 0.000,  2.866,  0.000)  2.866
```

(continues on next page)

(continued from previous page)

```
  Fe2   Fe2   (  0,   1,   0)  9.9211   ( 0.000,  2.866,  0.000)  2.866
  Fe1   Fe1   (  1,   0,   0)  9.9213   ( 2.866,  0.000,  0.000)  2.866
  Fe2   Fe2   (  1,   0,   0)  9.9213   ( 2.866,  0.000,  0.000)  2.866
  Fe1   Fe1   ( -1,  -1,   0)  0.8963   (-2.866, -2.866,  0.000)  4.053
  Fe2   Fe2   ( -1,  -1,   0)  0.8963   (-2.866, -2.866,  0.000)  4.053
  Fe1   Fe1   ( -1,   0,  -1)  0.8970   (-2.866,  0.000, -2.866)  4.053
  Fe2   Fe2   ( -1,   0,  -1)  0.8970   (-2.866,  0.000, -2.866)  4.053
```

Several other formats of the exchange parameters are also provided in the `TB2J_results` directory , which can be used in spin dynamics code, e.g. MULTIBINIT, Vampire.

Non-collinear calculation with SOC

The DMI and anisotropic exchange are result of the SOC, therefore requires the DFT calculation to be done with SOC enabled. To get the full set of exchange parameters, a "rotate and merge" procedure is needed, in which several DFT calculations with either the structure or the spin rotated are needed. For each of the non-collinear calcualtion, we compute the exchange parameters from the DFT calculation with the same command as in the collienar case.

```
abacus2J.py --path . --suffix Fe --elements Fe  --kmesh 7 7 7
```

And then the "TB2J_merge.py" command can be used to get the final spin interaction parameters.

Parameters of abacus2J.py

We can use the command

```
abacus2J.py --help
```

to view the parameters and the usage of them in abacus2J.py.

Acknowledgments

We thanks to Xu He to provide critical interface support.

CHAPTER

FIFTEEN

DETAILED INTRODUCTION OF THE INPUT FILES

15.1 Full List of INPUT Keywords

* *td_print_eij*
* *td_propagator*
* *td_vext*
* *td_vext_dire*
* *td_stype*
* *td_ttype*
* *td_tstart*
* *td_tend*
* *td_lcut1*
* *td_lcut2*
* *td_gauss_freq*
* *td_gauss_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*
* *init_vecpot_file*
* *ocp*
* *ocp_set*
* *out_dipole*
* *out_current*
* *out_current_k*
* *out_efield*
* *out_vecpot*

- *Reduced Density Matrix Functional Theory*
  - *rdmft*
  - *rdmft_power_alpha*

15.1.1 System variables

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 the calculation.
- **Default**: 0

calculation

- **Type**: String
- **Description**: Specify the type of calculation.
  - scf: perform self-consistent electronic structure calculations
  - nscf: perform non-self-consistent electronic structure calculations. A charge density file is required
  - relax: perform structure relaxation calculations, the relax_nmax parameter depicts the maximal number of ionic iterations
  - cell-relax: perform cell relaxation calculations
  - md: perform molecular dynamics simulations
  - get_pchg: obtain partial (band-decomposed) charge densities (for LCAO basis only). See out_pchg for more information
  - get_wf: obtain real space wave functions (for LCAO basis only). See out_wfc_norm and out_wfc_re_im for more information
  - get_s: obtain the overlap matrix formed by localized orbitals (for LCAO basis with multiple k points). the file name is SR.csr with file format being the same as that generated by out_mat_hs2
  - gen_bessel: generates projectors, i.e., a series of Bessel functions, for the DeePKS method (for LCAO basis only)
  - gen_opt_abfs: generate opt-ABFs as discussed in this article
  - test_memory: obtain a rough estimation of memory consumption for the calculation
  - test_neighbour: obtain information of neighboring atoms (for LCAO basis only), please specify a positive search_radius manually
- **Default**: scf

esolver_type

- **Type**: String
- **Description**: Choose the energy solver.
  - ksdft: Kohn-Sham density functional theory
  - ofdft: orbital-free density functional theory
  - tdofdft: time-dependent orbital-free density functional theory
  - sdft: stochastic density functional theory
  - tddft: real-time time-dependent density functional theory (RT-TDDFT)
  - lj: Leonard Jones potential
  - dp: DeeP potential
  - nep: Neuroevolution Potential
  - ks-lr: Kohn-Sham density functional theory + LR-TDDFT (Under Development Feature)
  - lr: LR-TDDFT with given KS orbitals (Under Development Feature)
- **Default**: ksdft

symmetry

- **Type**: String
- **Description**: Takes value 1, 0 or -1.
  - -1: No symmetry will be considered. It is recommended to set -1 for non-colinear + soc calculations, where time reversal symmetry is broken sometimes.
  - 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.
  - 1: Symmetry analysis will be performed to determine the type of Bravais lattice and associated symmetry operations. (point groups, space groups, primitive cells, and irreducible k-points)

    Note: When symmetry is enabled (value 1), k-points are reduced to the irreducible Brillouin zone (IBZ). For explicit k-point lists with custom weights (see KPT file), the custom weights are preserved during symmetry reduction. For Monkhorst-Pack grids, uniform weights are used.
- **Default**: default

symmetry_prec

- **Type**: Real
- **Description**: The accuracy for symmetry analysis. Typically, the default value is good enough, but if the lattice parameters or atom positions in STRU file are not accurate enough, this value should be enlarged.

  Note: if calculation==cell_relax, this value can be dynamically changed corresponding to the variation of accuracy of the lattice parameters and atom positions during the relaxation.
- **Default**: 1.0e-6
- **Unit**: Bohr

symmetry_autoclose

- **Type**: Boolean
- **Availability**: *symmetry==1*
- **Description**: Control how to deal with error in symmetry analysis due to inaccurate lattice parameters or atom positions in STRU file, especially useful when calculation==cell-relax
  - False: quit with an error message
  - True: automatically set symmetry to 0 and continue running without symmetry analysis
- **Default**: True

cal_force

- **Type**: Boolean
- **Description**: If set to True, calculate the force at the end of the electronic iteration.
- **Default**: False

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 processes.
- **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.

  Available options are:
  - none: free structure
  - sc: simple cubic
  - fcc: face-centered cubic
  - bcc: body-centered cubic
  - hexagonal: hexagonal
  - trigonal: trigonal
  - st: simple tetragonal
  - bct: body-centered tetragonal

  - so: orthorhombic
  - baco: base-centered orthorhombic
  - fco: face-centered orthorhombic
  - bco: body-centered orthorhombic
  - sm: simple monoclinic
  - bacm: base-centered monoclinic
  - triclinic: triclinic
- **Default**: none

init_wfc

- **Type**: String
- **Description**: The type of the starting wave functions.

  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 binary files wf*.dat, which are output by setting out_wfc_pw to 2.
  - random: random numbers
  - nao: from numerical atomic orbitals. If they are not enough, other wave functions are initialized with random numbers.
  - nao+random: add small random numbers on numerical atomic orbitals

    Note: Only the file option is useful for the lcao basis set, which is mostly used when calculation is set to get_wf and get_pchg.
- **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.
  - atomic: the density is starting from the summation of the atomic density of single atoms.
  - file: the density will be read in from a binary file charge-density.dat first. If it does not exist, the charge density will be read in from cube files.
  - wfc: the density will be calculated by wavefunctions and occupations.
  - dm: the density will be calculated by real space density matrix(DMR) of LCAO base.
  - hr: the real space Hamiltonian matrix(HR) will be read in from file hrs1_nao.csr in directory read_file_dir.
  - auto: Abacus first attempts to read the density from a file; if not found, it defaults to using atomic density.
- **Default**: atomic

init_vel

- **Type**: Boolean
- **Description**: - True: read the atom velocity (atomic unit : 1 a.u. = 21.877 Angstrom/fs) from the atom file (STRU) and determine the initial temperature md_tfirst. If md_tfirst is unset or less than zero, init_vel is autoset to be true.
  - False: assign value to atom velocity using Gaussian distributed random numbers.
- **Default**: False

mem_saver

- **Type**: Integer
- **Availability**: *Used only for nscf calculations with plane wave basis set.*
- **Description**: Save memory when performing nscf calculations.
  - 0: no memory saving techniques are used.
  - 1: a memory saving technique will be used for many k point calculations.
- **Default**: 0

cal_stress

- **Type**: Boolean
- **Description**: If set to True, calculate the stress at the end of the electronic iteration.
- **Default**: False

diago_proc

- **Type**: Integer
- **Availability**: *Used only for plane wave basis set.*
- **Description**: - 0: it will be set to the number of MPI processes.
  - >0: it specifies the number of processes used for carrying out diagonalization. Must be less than or equal to total number of MPI processes.
- **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**: Vector of Real (1 or 3 values)
- **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. The default value 0.0 means that ABACUS will read the applied KPT file.

  Note: If gamma_only is set to be true, kspacing is invalid.

- **Default**: 0.0

koffset

- **Type**: Vector of Real (3 values)
- **Description**: Set offsets for automatic k-point mesh generated by kspacing, in each reciprocal direction. This parameter is only effective when kspacing > 0.0 and gamma_only is false.
- **Default**: 0.0 0.0 0.0

kmesh_type

- **Type**: String
- **Description**: Set mesh type used for automatic k-point mesh generated by kspacing. Available options are gamma and mp. This parameter is only effective when kspacing > 0.0 and gamma_only is false.
- **Default**: gamma

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.
- **Default**: 0.2

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 or ROCm.

    Note: ks_solver must also be set to the algorithms supported. lcao_in_pw currently does not support gpu.
- **Default**: cpu

precision

- **Type**: String
- **Availability**: *Used only for plane wave basis set.*
- **Description**: Specifies the precision when performing scf calculation.
  - single: single precision
  - double: double precision
- **Default**: double

gint_precision

- **Type**: String
- **Availability**: *Used only for LCAO basis set.*
- **Description**: Specifies the precision when performing grid integral in LCAO calculations.
  - single: single precision
  - double: double precision
  - mix: mixed precision, starting from single precision and switching to double precision when the SCF residual becomes small enough
- **Default**: double

timer_enable_nvtx

- **Type**: Boolean
- **Description**: Controls whether NVTX profiling labels are emitted by the timer. This feature is only effective on CUDA platforms.
  - True: Enable NVTX profiling labels in the timer.
  - False: Disable NVTX profiling labels in the timer.
- **Default**: False

cell_factor

- **Type**: Real
- **Description**: Used in the construction of the pseudopotential tables. For cell-relax calculations, this is automatically set to 2.0.
- **Default**: 1.2

dm_to_rho

- **Type**: Boolean
- **Description**: Reads density matrix in npz format and calculates electron density.
- **Default**: False

chg_extrap

- **Type**: String
- **Description**: Charge extrapolation method for MD and relaxation calculations.
- **Default**: default

nb2d

- **Type**: Integer
- **Description**: In LCAO calculations, the Hamiltonian and overlap matrices are distributed across 2D processor grid. This parameter controls the 2D block size for distribution.
- **Default**: 0

cal_symm_repr

- **Type**: Integer [Integer](optional)
- **Description**: Whether to print the matrix representation of symmetry operation to running log file. If the first value is given as 1, then all matrix representations will be printed. The second optional parameter controls the precision (number of digits) to print, default is 3, which is enough for a quick check.
- **Default**: 1 3

back to top

15.1.2 Input files

stru_file

- **Type**: String
- **Description**: The name of the structure file containing 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**: The name of the k-point file that includes the k-point information of Brillouin zone.
- **Default**: KPT

pseudo_dir

- **Type**: String
- **Description**: The directory of pseudopotential files. This parameter is combined with the pseudopotential filenames in the STRU file to form the complete pseudopotential file paths.
- **Default**: “”

orbital_dir

- **Type**: String
- **Description**: The directory to save numerical atomic orbitals. This parameter is combined with orbital filenames in the STRU file to form the complete orbital file paths.
- **Default**: “”

read_file_dir

- **Type**: String
- **Description**: Location of files, such as the electron density (chgs1.cube), required as a starting point.
- **Default**: OUT.$suffix

restart_load

- **Type**: Boolean
- **Availability**: *Used only when numerical atomic orbitals are employed as basis set.*

- **Description**: If restart_save is set to true and an electronic iteration is finished, calculations can be restarted from the charge density file, which are saved in the former calculation.
- **Default**: False

spillage_outdir

- **Type**: String
- **Availability**: *Used only for plane wave basis set.*
- **Description**: The directory to save the spillage files.
- **Default**: "./"

back to top

15.1.3 Plane wave related variables

ecutwfc

- **Type**: Real
- **Description**: Energy cutoff for plane wave functions. 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.

  Note: ecutwfc and ecutrho can be set simultaneously. If only one parameter is set, abacus will automatically set another parameter based on the 4-time relationship.
- **Default**: 50 for PW basis, 100 for LCAO basis
- **Unit**: Ry

ecutrho

- **Type**: Real
- **Description**: Energy cutoff for charge density and potential. For norm-conserving pseudopotential you should stick to the default value, you can reduce it by a little but it will introduce noise especially on forces and stress.
- **Default**: 4*ecutwfc
- **Unit**: Ry

nx

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points in x direction. If set to 0, the number will be calculated from ecutrho.

  Note: You must specify all three dimensions (nx, ny, nz) for this setting to be used.
- **Default**: 0

ny

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points in y direction. If set to 0, the number will be calculated from ecutrho.

  Note: You must specify all three dimensions (nx, ny, nz) for this setting to be used.

- **Default**: 0

nz

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points in z direction. If set to 0, the number will be calculated from ecutrho.

  Note: You must specify all three dimensions (nx, ny, nz) for this setting to be used.

- **Default**: 0

ndx

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points for the dense part of charge density in x direction. If set to 0, the number will be calculated from ecutwfc.

  Note: You must specify all three dimensions (ndx, ndy, ndz) for this setting to be used. These parameters must be used combined with nx, ny, nz. If nx, ny, nz are unset, ndx, ndy, ndz are used as nx, ny, nz.

- **Default**: 0

ndy

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points for the dense part of charge density in y direction. If set to 0, the number will be calculated from ecutwfc.

  Note: You must specify all three dimensions (ndx, ndy, ndz) for this setting to be used. These parameters must be used combined with nx, ny, nz. If nx, ny, nz are unset, ndx, ndy, ndz are used as nx, ny, nz.

- **Default**: 0

ndz

- **Type**: Integer
- **Description**: If set to a positive number, specifies the number of FFT grid points for the dense part of charge density in z direction. If set to 0, the number will be calculated from ecutwfc.

  Note: You must specify all three dimensions (ndx, ndy, ndz) for this setting to be used. These parameters must be used combined with nx, ny, nz. If nx, ny, nz are unset, ndx, ndy, ndz are used as nx, ny, nz.

- **Default**: 0

pw_seed

- **Type**: Integer
- **Availability**: *Only used for plane wave basis.*
- **Description**: Specify the random seed to initialize wave functions. Only positive integers are available.
- **Default**: 0

diag_subspace

- **Type**: Integer
- **Description**: The method to diagonalize subspace in dav_subspace method.
  - 0: by LAPACK
  - 1: by GenELPA
  - 2: by ScaLAPACK
- **Default**: 0

erf_ecut

- **Type**: Real
- **Description**: Used in variable-cell molecular dynamics (or in stress calculation). See erf_sigma for details.
- **Default**: 0.0
- **Unit**: Ry

fft_mode

- **Type**: Integer
- **Description**: Set the mode of FFTW.
  - 0: FFTW_ESTIMATE
  - 1: FFTW_MEASURE
  - 2: FFTW_PATIENT
  - 3: FFTW_EXHAUSTIVE
- **Default**: 0

erf_height

- **Type**: Real
- **Description**: Used in variable-cell molecular dynamics (or in stress calculation). See erf_sigma for details.
- **Default**: 0.0
- **Unit**: Ry

erf_sigma

- **Type**: Real
- **Description**: In order to recover the accuracy of a constant energy cutoff calculation, the kinetic functional is modified, which is used in variable-cell molecular dynamics (or in stress calculation).
- **Default**: 0.1
- **Unit**: Ry

pw_diag_thr

- **Type**: Real
- **Description**: Only used when you use ks_solver = cg/dav/dav_subspace/bpcg. 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

diago_smooth_ethr

- **Type**: Boolean
- **Description**: If TRUE, the smooth threshold strategy, which applies a larger threshold (10e-5) for the empty states, will be implemented in the diagonalization methods. (This strategy should not affect total energy, forces, and other ground-state properties, but computational efficiency will be improved.) If FALSE, the smooth threshold strategy will not be applied.
- **Default**: false

use_k_continuity

- **Type**: Boolean
- **Availability**: *Used only for plane wave basis set.*
- **Description**: If TRUE, the wavefunctions at k-point will be initialized from the converged wavefunctions at the nearest k-point, which can speed up the SCF convergence. Only works for PW basis.
- **Default**: false

pw_diag_nmax

- **Type**: Integer
- **Availability**: *basis_type==pw, ks_solver==cg/dav/dav_subspace/bpcg*
- **Description**: Only useful when you use ks_solver = cg/dav/dav_subspace/bpcg. It indicates the maximal iteration number for cg/david/dav_subspace/bpcg method.
- **Default**: 50

pw_diag_ndim

- **Type**: Integer
- **Description**: Only useful when you use ks_solver = dav or ks_solver = dav_subspace. It indicates dimension of workspace(number of wavefunction packets, at least 2 needed) for the Davidson method. A larger value may yield a smaller number of iterations in the algorithm but uses more memory and more CPU time in subspace diagonalization.
- **Default**: 4

diago_cg_prec

- **Type**: Integer
- **Description**: Preconditioner type for conjugate gradient diagonalization method.
- **Default**: 1

back to top

15.1.4 Numerical atomic orbitals related variables

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 (in Ry) for two-center integrals in LCAO. The two-center integration table are obtained via a k space integral whose upper limit is about sqrt(lcao_ecut).
- **Default**: ecutwfc

lcao_dk

- **Type**: Real
- **Description**: the interval of k points for two-center integrals. The two-center integration table are obtained via a k space integral on a uniform grid with spacing lcao_dk.
- **Default**: 0.01
- **Unit**: Bohr

lcao_dr

- **Type**: Real
- **Description**: r spacing of the integration table of two-center integrals.
- **Default**: 0.01
- **Unit**: Bohr

lcao_rmax

- **Type**: Real
- **Description**: Maximum distance for the two-center integration table.
- **Default**: 30
- **Unit**: Bohr

search_radius

- **Type**: Real
- **Description**: Searching radius in finding the neighbouring atoms. By default the radius will be automatically determined by the cutoffs of orbitals and nonlocal beta projectors.
- **Default**: -1
- **Unit**: Bohr

bx

- **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. The default is 0, which means abacus will automatically calculate these values.
- **Default**: 0

by

- **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. The default is 0, which means abacus will automatically calculate these values.
- **Default**: 0

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. The default is 0, which means abacus will automatically calculate these values.
- **Default**: 0

elpa_num_thread

- **Type**: Integer
- **Description**: Number of threads used in one elpa calculation.

  If the number is below 0 or 0 or beyond the max number of threads, all elpa calculation will be using all mpi threads
- **Default**: -1

num_stream

- **Type**: Integer
- **Description**: The number of CUDA streams used in LCAO calculations with GPU acceleration.
- **Default**: 4

back to top

15.1.5 Electronic structure

basis_type

- **Type**: String
- **Description**: Choose the basis set.
  - pw: Using plane-wave basis set only.
  - lcao: Using localized atomic orbital sets.
  - lcao_in_pw: Expand the localized atomic set in plane-wave basis, non-self-consistent field calculation not tested.

- **Default**: pw

ks_solver

- **Type**: String
- **Description**: Choose the diagonalization methods for the Hamiltonian matrix expanded in a certain basis set.

  For plane-wave basis,

  - cg: The conjugate-gradient (CG) method.
  - bpcg: The BPCG method, which is a block-parallel Conjugate Gradient (CG) method, typically exhibits higher acceleration in a GPU environment.
  - dav: The Davidson algorithm.
  - dav_subspace: The Davidson algorithm without orthogonalization operation, this method is the most recommended for efficiency. `pw_diag_ndim` can be set to 2 for this method.

  For numerical atomic orbitals basis,

  - lapack: Use LAPACK to diagonalize the Hamiltonian, only used for serial version
  - genelpa: Use GEN-ELPA to diagonalize the Hamiltonian.
  - scalapack_gvx: Use Scalapack to diagonalize the Hamiltonian.
  - cusolver: Use CUSOLVER to diagonalize the Hamiltonian, at least one GPU is needed.
  - cusolvermp: Use CUSOLVER to diagonalize the Hamiltonian, supporting multi-GPU devices. Note that you should set the number of MPI processes equal to the number of GPUs.
  - elpa: The ELPA solver supports both CPU and GPU. By setting the `device` to GPU, you can launch the ELPA solver with GPU acceleration (provided that you have installed a GPU-supported version of ELPA, which requires you to manually compile and install ELPA, and the ABACUS should be compiled with -DUSE_ELPA=ON and -DUSE_CUDA=ON). The ELPA solver also supports multi-GPU acceleration.

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

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

  Then the user has to correct the input file and restart the calculation.

- **Default**:
  - PW basis: cg.
  - LCAO basis:
    * genelpa (if compiling option `USE_ELPA` has been set)
    * lapack (if compiling option `ENABLE_MPI` has not been set)
    * scalapack_gvx (if compiling option `USE_ELPA` has not been set and compiling option `ENABLE_MPI` has been set)
    * cusolver (if compiling option `USE_CUDA` has been set)

nbands

- **Type**: Integer
- **Description**: The number of Kohn-Sham orbitals to calculate. It is recommended to setup this value, especially when smearing techniques are utilized, more bands should be included.

nelec

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

nelec_delta

- **Type**: Real
- **Description**: The total number of electrons will be calculated by nelec+nelec_delta.
- **Default**: 0.0

nupdown

- **Type**: Real
- **Description**: - 0.0: no constrain apply to system.
  - >0.0: The different number of electrons between spin-up and spin-down channels. The range of value must be in [-nelec ~ nelec]. It is one type of constrainted DFT method, two Fermi energies will be calculated.
- **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. A complete list of 'short-hand' expressions can be found in the source code. Supported density functionals are:
  - LDA functionals
  - LDA (equivalent with PZ and SLAPZNOGXNOGC), PWLDA
  - GGA functionals
  - PBE (equivalent with SLAPWPBXPBC), PBESOL, REVPBE, WC, BLYP, BP(referred to BP86), PW91, HCTH, OLYP, BLYP_LR
  - meta-GGA functionals
  - SCAN (require LIBXC)
  - Hybrid functionals
  - PBE0, HF
  - If LIBXC is available, additional short-hand names of hybrid functionals are supported: HSE(referred to HSE06), B3LYP, LC_PBE, LC_WPBE, LRC_WPBE, LRC_WPBEH, CAM_PBEH, WP22, CWP22, MULLER (equivalent with POWER)
  - Hybrid meta-GGA functionals
  - SCAN0 (require LIBXC)

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'.

- **Default**: Used the same as DFT functional as specified in the pseudopotential files.

xc_temperature

- **Type**: Real
- **Description**: Specifies temperature when using temperature-dependent XC functionals (KSDT and so on).
- **Default**: 0.0
- **Unit**: Ry

xc_exch_ext

- **Type**: Integer followed by Real values
- **Description**: Customized parameterization on the exchange part of XC functional. The first value should be the LibXC ID of the original functional, and latter values are external parameters. Default values are those of Perdew-Burke-Ernzerhof (PBE) functional. For more information on LibXC ID of functionals, please refer to LibXC. For parameters of functionals of interest, please refer to the source code of LibXC, such as PBE functional interface in LibXC: gga_x_pbe.c.

  Note: Solely setting this keyword will take no effect on XC functionals. One should also set dft_functional to the corresponding functional to apply the customized parameterization. Presently this feature can only support parameterization on one exchange functional.

- **Default**: 101 0.8040 0.2195149727645171

xc_corr_ext

- **Type**: Integer followed by Real values
- **Description**: Customized parameterization on the correlation part of XC functional. The first value should be the LibXC ID of the original functional, and latter values are external parameters. Default values are those of Perdew-Burke-Ernzerhof (PBE) functional. For more information on LibXC ID of functionals, please refer to LibXC. For parameters of functionals of interest, please refer to the source code of LibXC, such as PBE functional interface in LibXC: gga_c_pbe.c.

  Note: Solely setting this keyword will take no effect on XC functionals. One should also set dft_functional to the corresponding functional to apply the customized parameterization. Presently this feature can only support parameterization on one correlation functional.

- **Default**: 130 0.06672455060314922 0.031090690869654895034 1.0

pseudo_rcut

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

pseudo_mesh

- **Type**: Boolean
- **Description**: - 0: Use a mesh for radial integration of pseudopotentials.
  - 1: Use the mesh that is consistent with quantum espresso
- **Default**: 0

nspin

- **Type**: Integer
- **Description**: The number of spin components of wave functions.
  - 1: Spin degeneracy
  - 2: Collinear spin polarized.
  - 4: For the case of noncollinear polarized, nspin will be automatically set to 4 without being specified by the user.
- **Default**: 1

smearing_method

- **Type**: String
- **Description**: It indicates which occupation and smearing method is used in the calculation.
  - fixed: fixed occupations (available for non-coductors only)
  - gauss or gaussian: Gaussian smearing method.
  - mp: methfessel-paxton smearing method; recommended for metals.
  - mp2: 2-nd methfessel-paxton smearing method; recommended for metals.
  - mv or cold: marzari-vanderbilt smearing method.
  - fd: Fermi-Dirac smearing method: and smearing_sigma below is the temperature (in Ry).
- **Default**: gauss

smearing_sigma

- **Type**: Real
- **Description**: Energy range for smearing.
- **Default**: 0.015
- **Unit**: Ry

smearing_sigma_temp

- **Type**: Real
- **Description**: Energy range for smearing, smearing_sigma = 1/2 kB smearing_sigma_temp.
- **Default**: 2 * smearing_sigma / kB.
- **Unit**: K

mixing_type

- **Type**: String
- **Description**: Charge mixing methods.
  - plain: Just simple mixing.
  - pulay: Standard Pulay method. P. Pulay Chemical Physics Letters, (1980)
  - broyden: Simplified modified Broyden method. D.D. Johnson Physical Review B (1988)

  In general, the convergence of the Broyden method is slightly faster than that of the Pulay method.
- **Default**: broyden

mixing_beta

- **Type**: Real
- **Description**: In general, the formula of charge mixing can be written as rho_new = rho_old + mixing_beta * drho, where rho_new represents the new charge density after charge mixing, rho_old represents the charge density in previous step, drho is obtained through various mixing methods, and mixing_beta is set by this parameter. A lower value of 'mixing_beta' results in less influence of drho on rho_new, making the self-consistent field (SCF) calculation more stable. However, it may require more steps to achieve convergence. We recommend the following options:
  - 0.8: nspin=1
  - 0.4: nspin=2 and nspin=4
  - 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.

  Note: For low-dimensional large systems, the setup of mixing_beta=0.1, mixing_ndim=20, and mixing_gg0=1.0 usually works well.
- **Default**: 0.8 for nspin=1, 0.4 for nspin=2 and nspin=4.

mixing_beta_mag

- **Type**: Real
- **Description**: Mixing parameter of magnetic density.
- **Default**: 4*mixing_beta, but the maximum value is 1.6.

mixing_ndim

- **Type**: Integer
- **Description**: It indicates the mixing dimensions in Pulay or Broyden. Pulay and Broyden method use the density from previous mixing_ndim steps and do a charge mixing based on this density.

  For systems that are difficult to converge, one could try increasing the value of 'mixing_ndim' to enhance the stability of the self-consistent field (SCF) calculation.
- **Default**: 8

mixing_restart

- **Type**: Real
- **Description**: If the density difference between input and output drho is smaller than mixing_restart, SCF will restart at next step which means SCF will restart by using output charge density from perivos iteration as input charge density directly, and start a new mixing. Notice that mixing_restart will only take effect once in one SCF.
- **Default**: 0

mixing_dmr

- **Type**: Boolean
- **Availability**: *Only for mixing_restart >= 0.0*
- **Description**: At n-th iteration which is calculated by drho<mixing_restart, SCF will start a mixing for real-space density matrix by using the same coefficiences as the mixing of charge density.
- **Default**: false

mixing_gg0

- **Type**: Real
- **Description**: Whether to perfom Kerker scaling for charge density.
  - >0: The high frequency wave vectors will be suppressed by multiplying a scaling factor. Setting mixing_gg0 = 1.0 is normally a good starting point. Kerker preconditioner will be automatically turned off if mixing_beta <= 0.1.
  - 0: No Kerker scaling is performed.

  For systems that are difficult to converge, particularly metallic systems, enabling Kerker scaling may aid in achieving convergence.
- **Default**: 1.0

mixing_gg0_mag

- **Type**: Real
- **Description**: Whether to perfom Kerker preconditioner of magnetic density. Note: we do not recommand to open Kerker preconditioner of magnetic density unless the system is too hard to converge.
- **Default**: 0.0

mixing_gg0_min

- **Type**: Real
- **Description**: The minimum kerker coefficient.
- **Default**: 0.1

mixing_angle

- **Type**: Real
- **Availability**: *Only relevant for non-colinear calculations nspin=4.*

- **Description**: Normal broyden mixing can give the converged result for a given magnetic configuration. If one is not interested in the energies of a given magnetic configuration but wants to determine the ground state by relaxing the magnetic moments' directions, one cannot rely on the standard Broyden mixing algorithm. To enhance the ability to find correct magnetic configuration for non-colinear calculations, ABACUS implements a promising mixing method proposed by J. Phys. Soc. Jpn. 82 (2013) 114706. Here, mixing_angle is the angle mixing parameter. In fact, only mixing_angle=1.0 is implemented currently.
  - <=0: Normal broyden mixing
  - >0: Angle mixing for the modulus with mixing_angle=1.0
- **Default**: -10.0

mixing_tau

- **Type**: Boolean
- **Availability**: *Only relevant for meta-GGA calculations.*
- **Description**: Whether to mix the kinetic energy density.
  - True: 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.
  - False: The kinetic energy density will not be mixed.
- **Default**: False

mixing_dftu

- **Type**: Boolean
- **Availability**: *Only relevant for DFT+U calculations.*
- **Description**: Whether to mix the occupation matrices.
  - True: The occupation matrices will also be mixed by plain mixing. From experience this is not very helpful if the +U calculation does not converge.
  - False: The occupation matrices will not be mixed.
- **Default**: False

gamma_only

- **Type**: Boolean
- **Availability**: *Only used in localized orbitals set*
- **Description**: Whether to use gamma_only algorithm.
  - 0: more than one k-point is used and the ABACUS is slower compared to the gamma only algorithm.
  - 1: ABACUS uses gamma only, the algorithm is faster and you don't need to specify the k-points file.

  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

scf_nmax

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

scf_thr

- **Type**: Real
- **Description**: It's the density threshold for electronic iteration. It represents the charge density error between two sequential densities from electronic iterations. This criterion is always enabled. If `scf_ene_thr` is set, the total-energy criterion (`scf_ene_thr`) is evaluated conditionally after the charge-density criterion (`scf_thr`) is satisfied, and not on the first iteration. For local-orbital calculations, 1e-6 is usually accurate enough.
- **Default**: 1.0e-9 (plane-wave basis), or 1.0e-7 (localized atomic orbital basis).
- **Unit**: Ry if scf_thr_type=1, dimensionless if scf_thr_type=2

scf_ene_thr

- **Type**: Real
- **Description**: It's the energy threshold for electronic iteration. The compared quantity is the total-energy difference evaluated from the charge densities before and after the `Hpsi` operation in one SCF step. It is not the same as the screen-output `EDIFF`, which is the energy difference before `Hpsi` and after charge mixing (i.e., across both `Hpsi` and charge-mixing operations).
- **Default**: -1.0. If the user does not set this parameter, it will not take effect.
- **Unit**: eV

scf_thr_type

- **Type**: Integer
- **Description**: Choose the calculation method of convergence criterion.
  - 1: the criterion is defined in reciprocal space, which is used in SCF of PW basis with unit Ry.
  - 2: the criterion is defined in real space, where is the number of electron, which is used in SCF of LCAO with unit dimensionless.
- **Default**: 1 (plane-wave basis), or 2 (localized atomic orbital basis).

scf_os_stop

- **Type**: Boolean
- **Description**: For systems that are difficult to converge, the SCF process may exhibit oscillations in charge density, preventing further progress toward the specified convergence criteria and resulting in continuous oscillation until the maximum number of steps is reached; this greatly wastes computational resources. To address this issue, this function allows ABACUS to terminate the SCF process early upon detecting oscillations, thus reducing subsequent meaningless calculations. The detection of oscillations is based on the slope of the logarithm of historical drho values. To this end, Least Squares Method is used to calculate the slope of the logarithmically taken drho for the previous scf_os_ndim iterations. If the calculated slope is larger than scf_os_thr, stop the SCF.
  - 0: The SCF will continue to run regardless of whether there is oscillation or not.
  - 1: If the calculated slope is larger than scf_os_thr, stop the SCF.

- **Default**: false

scf_os_thr

- **Type**: Real
- **Description**: The slope threshold to determine if the SCF is stuck in a charge density oscillation. If the calculated slope is larger than scf_os_thr, stop the SCF.
- **Default**: -0.01

scf_os_ndim

- **Type**: Integer
- **Description**: To determine the number of old iterations' drho used in slope calculations.
- **Default**: mixing_ndim

sc_os_ndim

- **Type**: Integer
- **Description**: To determine the number of old iterations to judge oscillation, it occured, more accurate lambda with DeltaSpin method would be calculated, only for PW base.
- **Default**: 5

lspinorb

- **Type**: Boolean
- **Description**: Whether to consider spin-orbit coupling (SOC) effect in the calculation.
  - True: Consider spin-orbit coupling effect. When enabled:
  - nspin is automatically set to 4 (noncollinear spin representation)
  - Symmetry is automatically disabled (SOC breaks inversion symmetry)
  - Requires full-relativistic pseudopotentials with has_so=true in the UPF header
  - False: Do not consider spin-orbit coupling effect.
  - Common Error: "no soc upf used for lspinorb calculation" - ensure you are using full-relativistic pseudopotentials
- **Default**: False

noncolin

- **Type**: Boolean
- **Description**: Whether to allow non-collinear magnetic moments, where magnetization can point in arbitrary directions (x, y, z components) rather than being constrained to the z-axis.
  - True: Allow non-collinear polarization. When enabled:
  - nspin is automatically set to 4
  - Wave function dimension is doubled (npol=2), and the number of occupied states is doubled
  - Charge density has 4 components (Pauli spin matrices)
  - Cannot be used with gamma_only=true

  - Can be combined with lspinorb=true for SOC effects with non-collinear magnetism
  - False: Do not allow non-collinear polarization (magnetization constrained to z-axis).
  - Relationship with lspinorb:
  - noncolin=0, lspinorb=1: SOC with z-axis magnetism only (for non-magnetic materials with SOC)
  - noncolin=1, lspinorb=0: Non-collinear magnetism without SOC
  - noncolin=1, lspinorb=1: Both non-collinear magnetism and SOC
- **Default**: False

soc_lambda

- **Type**: Real
- **Availability**: *Only works when lspinorb=true*
- **Description**: Modulates the strength of spin-orbit coupling effect. Sometimes, for some real materials, both scalar-relativistic and full-relativistic pseudopotentials cannot 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 to modulate SOC effect:

  - soc_lambda 0.0: Scalar-relativistic case (no SOC)
  - soc_lambda 1.0: Full-relativistic case (full SOC)
  - Intermediate values: Partial-relativistic SOC (interpolation between scalar and full)

  Use case: When experimental or high-level theoretical results suggest that the SOC effect is weaker or stronger than what full-relativistic pseudopotentials predict, you can adjust this parameter to match the target behavior.
- **Default**: 1.0

dfthalf_type

- **Type**: Integer
- **Description**: DFT-1/2 type:
  - 0: DFT-1/2 is off.
  - 1: Shell DFT-1/2 method is used.
- **Default**: 0

back to top

15.1.6 Electronic structure (SDFT)

method_sto

- **Type**: Integer
- **Availability**: *esolver_type = sdft*
- **Description**: Different methods to do stochastic DFT
  - 1: Calculate twice, this method cost less memory but is slower.
  - 2: Calculate once but needs much more memory. This method is much faster. Besides, it calculates with a smaller nche_sto. However, when the memory is not enough, only method 1 can be used.
  - other: use 2

- **Default**: 2

nbands_sto

- **Type**: Integer or string
- **Availability**: *esolver_type* = *sdft*
- **Description**: The number of stochastic orbitals
  - > 0: Perform stochastic DFT. Increasing the number of bands improves accuracy and reduces stochastic errors; To perform mixed stochastic-deterministic DFT, you should set nbands, which represents the number of KS orbitals.
  - 0: Perform Kohn-Sham DFT.
  - all: All complete basis sets are used to replace stochastic orbitals with the Chebyshev method (CT), resulting in the same results as KSDFT without stochastic errors.
- **Default**: 256

nche_sto

- **Type**: Integer
- **Availability**: *esolver_type* = *sdft*
- **Description**: Chebyshev expansion orders for stochastic DFT.
- **Default**: 100

emin_sto

- **Type**: Real
- **Availability**: *esolver_type* = *sdft*
- **Description**: Trial energy to guess the lower bound of eigen energies of the Hamiltonian Operator.
- **Default**: 0.0
- **Unit**: Ry

emax_sto

- **Type**: Real
- **Availability**: *esolver_type* = *sdft*
- **Description**: Trial energy to guess the upper bound of eigen energies of the Hamiltonian Operator.
- **Default**: 0.0
- **Unit**: Ry

seed_sto

- **Type**: Integer
- **Availability**: *esolver_type* = *sdft*
- **Description**: The random seed to generate stochastic orbitals.
  - >= 0: Stochastic orbitals have the form of exp(i*theta), *where theta is a uniform distribution in [0, 2*pi).
  - 0: the seed is decided by time(NULL).

  - <= -1: Stochastic orbitals have the form of +1 or -1 with equal probability.
  - -1: the seed is decided by time(NULL).
- **Default**: 0

initsto_ecut

- **Type**: Real
- **Availability**: *esolver_type = sdft*
- **Description**: Stochastic wave functions are initialized in a large box generated by "4*initsto_ecut". initsto_ecut should be larger than ecutwfc. In this method, SDFT results are the same when using different cores. Besides, coefficients of the same G are the same when ecutwfc is rising to initsto_ecut. If it is smaller than ecutwfc, it will be turned off.
- **Default**: 0.0
- **Unit**: Ry

initsto_freq

- **Type**: Integer
- **Availability**: *esolver_type = sdft*
- **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
- **Availability**: *method_sto = 2 and out_dos = 1 or cal_cond = True*
- **Description**: Make memory cost to 1/npart_sto times of the previous one when running the post process of SDFT like DOS or conductivities.
- **Default**: 1

back to top

15.1.7 Geometry relaxation

relax_method

- **Type**: Vector of string
- **Description**: The methods to do geometry optimization. The available algorithms depend on the relax_new setting.

  First element (algorithm selection):
  - cg: Conjugate gradient (CG) algorithm. Available for both relax_new = True (default, simultaneous optimization) and relax_new = False (nested optimization). See relax_new for implementation details.
  - bfgs: Broyden–Fletcher–Goldfarb–Shanno (BFGS) quasi-Newton algorithm. Only available when relax_new = False.
  - lbfgs: Limited-memory BFGS algorithm, suitable for large systems. Only available when relax_new = False.

  - cg_bfgs: Mixed method starting with CG and switching to BFGS when force convergence reaches relax_cg_thr. Only available when relax_new = False.
  - sd: Steepest descent algorithm. Only available when relax_new = False. Not recommended for production use.
  - fire: Fast Inertial Relaxation Engine method, a molecular-dynamics-based relaxation algorithm. Use by setting calculation to md and md_type to fire. Ionic velocities must be set in STRU file. See fire for details.

  Second element (BFGS variant, only when first element is bfgs):

  - 1: Traditional BFGS that updates the Hessian matrix B and then inverts it.
  - 2 or omitted: Default BFGS that directly updates the inverse Hessian (recommended).

    Note: In the 3.10-LTS version, the type of this parameter is std::string. It can be set to "cg", "bfgs", "cg_bfgs", "bfgs_trad", "lbfgs", "sd", "fire".

- **Default**: cg 1

relax_new

- **Type**: Boolean
- **Description**: Controls which implementation of geometry relaxation to use. At the end of 2022, a new implementation of the Conjugate Gradient (CG) method was introduced for relax and cell-relax calculations, while the old implementation was kept for backward compatibility.
  - True (default): Use the new CG implementation with the following features:
  - Simultaneous optimization of ionic positions and cell parameters (for cell-relax)
  - Line search algorithm for step size determination
  - Only CG algorithm is available (relax_method must be cg)
  - Supports advanced cell constraints: fixed_axes = "shape", "volume", "a", "b", "c", etc.
  - Supports fixed_ibrav to maintain lattice type
  - More efficient for variable-cell relaxation
  - Step size controlled by relax_scale_force
  - False: Use the old implementation with the following features:
  - Nested optimization procedure: ionic positions optimized first, then cell parameters (for cell-relax)
  - Multiple algorithms available: cg, bfgs, lbfgs, sd, cg_bfgs
  - Limited cell constraints: only fixed_axes = "volume" is supported
  - Traditional approach with separate ionic and cell optimization steps
- **Default**: True

relax_scale_force

- **Type**: Real
- **Availability**: *Only used when relax_new set to True*
- **Description**: The parameter controls the size of the first conjugate gradient 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

relax_nmax

- **Type**: Integer
- **Description**: The maximal number of ionic iteration steps. If set to 0, the code performs a quick "dry run", stopping just after initialization. This is useful to check for input correctness and to have the summary printed.
- **Default**: 1 for SCF, 50 for relax and cell-relax calcualtions

relax_cg_thr

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method = cg_bfgs*
- **Description**: When relax_method is set to cg_bfgs, a mixed algorithm of conjugate gradient (CG) and Broyden–Fletcher–Goldfarb–Shanno (BFGS) is used. The ions first move according to the CG method, then switch to the BFGS method when the maximum force on atoms is reduced below this threshold.
- **Default**: 0.5
- **Unit**: eV/Angstrom

force_thr

- **Type**: Real
- **Description**: Threshold of the force convergence. The threshold is compared with the largest force among all of the atoms. The recommended value for using atomic orbitals is 0.04 eV/Angstrom (0.0016 Ry/Bohr). The parameter is equivalent to force_thr_ev except for the unit, you can choose either you like.
- **Default**: 0.001
- **Unit**: Ry/Bohr (25.7112 eV/Angstrom)

force_thr_ev

- **Type**: Real
- **Description**: Threshold of the force convergence. The threshold is compared with the largest force among all of the atoms. The recommended value for using atomic orbitals is 0.04 eV/Angstrom (0.0016 Ry/Bohr). The parameter is equivalent to force_thr except for the unit. You may choose either you like.
- **Default**: 0.0257112
- **Unit**: eV/Angstrom (0.03889 Ry/Bohr)

force_zero_out

- **Type**: Real
- **Description**: The atomic forces that are smaller than force_zero_out will be treated as zero.
- **Default**: 0.0
- **Unit**: eV/Angstrom

relax_bfgs_w1

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method is bfgs or cg_bfgs*

- **Description**: Controls the Wolfe condition for the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm used in geometry relaxation. This parameter sets the sufficient decrease condition (c1 in Wolfe conditions). For more information, see Phys. Chem. Chem. Phys., 2000, 2, 2177.
- **Default**: 0.01

relax_bfgs_w2

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method is bfgs or cg_bfgs*
- **Description**: Controls the Wolfe condition for the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm used in geometry relaxation. This parameter sets the curvature condition (c2 in Wolfe conditions). For more information, see Phys. Chem. Chem. Phys., 2000, 2, 2177.
- **Default**: 0.5

relax_bfgs_rmax

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method is bfgs or cg_bfgs*
- **Description**: Maximum allowed total displacement of all atoms during geometry optimization. The sum of atomic displacements can increase during optimization steps but cannot exceed this value.
- **Default**: 0.8
- **Unit**: Bohr

relax_bfgs_rmin

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method = bfgs 1 (traditional BFGS)*
- **Description**: Minimum allowed total displacement of all atoms. When the total atomic displacement falls below this value and force convergence is not achieved, the calculation will terminate. Note: This parameter is not used in the default BFGS algorithm (relax_method = bfgs 2 or bfgs).
- **Default**: 1e-5
- **Unit**: Bohr

relax_bfgs_init

- **Type**: Real
- **Availability**: *Only used when relax_new = False and relax_method is bfgs or cg_bfgs*
- **Description**: Initial total displacement of all atoms in the first BFGS step. This sets the scale for the initial movement.
- **Default**: 0.5
- **Unit**: Bohr

stress_thr

- **Type**: Real
- **Description**: The threshold of the stress convergence. The threshold is compared with the largest component of the stress tensor.
- **Default**: 0.5
- **Unit**: kbar

press1

- **Type**: Real
- **Description**: The external pressures along three axes. Positive input value is taken as compressive stress.
- **Default**: 0
- **Unit**: kbar

press2

- **Type**: Real
- **Description**: The external pressures along three axes. Positive input value is taken as compressive stress.
- **Default**: 0
- **Unit**: kbar

press3

- **Type**: Real
- **Description**: The external pressures along three axes. Positive input value is taken as compressive stress.
- **Default**: 0
- **Unit**: kbar

fixed_axes

- **Type**: String
- **Availability**: *Only used when calculation is set to cell-relax*
- **Description**: Specifies which cell degrees of freedom are fixed during variable-cell relaxation. The available options depend on the relax_new setting:

  When relax_new = True (default), all options are available:

  - None: Default; all cell parameters can relax freely
  - volume: Relaxation with fixed volume (allows shape changes)
  - shape: Fix shape but allow volume changes (hydrostatic pressure only)
  - a: Fix the a-axis lattice vector during relaxation
  - b: Fix the b-axis lattice vector during relaxation
  - c: Fix the c-axis lattice vector 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

  When relax_new = False, all options are now available:

  - None: Default; all cell parameters can relax freely
  - volume: Relaxation with fixed volume (allows shape changes). Volume is preserved by rescaling the lattice after each update.
  - shape: Fix shape but allow volume changes (hydrostatic pressure only). Stress tensor is replaced with isotropic pressure.
  - a, b, c, ab, ac, bc: Fix specific lattice vectors. Gradients for fixed vectors are set to zero.

    Note: For VASP users, see the ISIF correspondence table in the geometry optimization documentation. Both implementations now support all constraint types.

- **Default**: None

fixed_ibrav

- **Type**: Boolean
- **Availability**: *Can be used with both relax_new = True and relax_new = False. A specific latname must be provided.*
- **Description**: - True: the lattice type will be preserved during relaxation. The lattice vectors are reconstructed to match the specified Bravais lattice type after each update.
  - False: No restrictions are exerted during relaxation in terms of lattice type

    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. When both are used, fixed_ibrav is applied first, then fixed_axes = "volume" rescaling is applied.
- **Default**: False

fixed_atoms

- **Type**: Boolean
- **Description**: - True: The direct coordinates of atoms will be preserved during variable-cell relaxation.
  - False: No restrictions are exerted on positions of all atoms. However, 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

back to top

15.1.8 Output information

out_freq_ion

- **Type**: Integer
- **Description**: Controls the output interval in ionic steps. When set to a positive integer, information such as charge density, local potential, electrostatic potential, Hamiltonian matrix, overlap matrix, density matrix, and Mulliken population analysis is printed every n ionic steps.

  Note: In RT-TDDFT calculations, this parameter is inactive; output frequency is instead controlled by out_freq_td.

- **Default**: 0

out_freq_td

- **Type**: Integer
- **Description**: Controls the output interval in completed electronic evolution steps during RT-TDDFT calculations. When set to a positive integer n, detailed information (see out_freq_ion) is printed every n electron time-evolution steps (i.e., every STEP OF ELECTRON EVOLVE). For example, if you wish to output information once per ionic step, you should set out_freq_td equal to estep_per_md, since one ionic step corresponds to estep_per_md electronic evolution steps.

  Note: This parameter is only active in RT-TDDFT mode (esolver_type = tddft). It has no effect in ground-state calculations.

- **Default**: 0

out_freq_elec

- **Type**: Integer
- **Description**: Output the charge density (only binary format, controlled by out_chg), wavefunction (controlled by out_wfc_pw) per out_freq_elec electronic iterations. Note that they are always output when converged or reach the maximum iterations scf_nmax.
- **Default**: scf_nmax

out_chg

- **Type**: Integer [Integer](optional)
- **Description**: The first integer controls whether to output the charge density on real space grids:
  - 1: Output the charge density (in Bohr^-3) on real space grids into the density files in the folder `OUT.${suffix}`. The files are named as:
    - nspin = 1: `chg.cube`;
    - nspin = 2: `chgs1.cube`, and `chgs2.cube`;
    - nspin = 4: `chgs1.cube`, `chgs2.cube`, `chgs3.cube`, and `chgs4.cube`;
    - When using the Meta-GGA functional, additional files containing the kinetic energy density are also output:
      - nspin = 1: `tau.cube`;
      - nspin = 2: `taus1.cube`, and `taus2.cube`;
      - nspin = 4: `taus1.cube`, `taus2.cube`, `taus3.cube`, and `taus4.cube`;
  - 2: On top of 1, also output the initial charge density files with a suffix name as '_ini', such as `taus1_ini.cube`, etc.
  - -1: Disable the charge density auto-back-up file `{suffix}-CHARGE-DENSITY.restart`, useful for large systems.

  The second integer controls the precision of the charge density output. If not given, `3` is used as default. For restarting from this file and other high-precision calculations, `10` is recommended.

  In molecular dynamics simulations, the output frequency is controlled by out_freq_ion.

  Note: In the 3.10-LTS version, the file names are SPIN1_CHG.cube and SPIN1_CHG_INI.cube, etc.

- **Default**: 0 3

out_pot

- **Type**: Integer [Integer](optional)
- **Description**: - 1: Output the total local potential (i.e., local pseudopotential + Hartree potential + XC potential + external electric field (if exists) + dipole correction potential (if exists) + …) on real space grids (in Ry) into files in the folder OUT.{suffix}. The files are named as:
  - nspin = 1: pots1.cube;
  - nspin = 2: pots1.cube and pots2.cube;
  - nspin = 4: pots1.cube, pots2.cube, pots3.cube, and pots4.cube
  - 2: Output the electrostatic potential on real space grids into OUT.{suffix}/pot_es.cube. The Python script named tools/average_pot/aveElecStatPot.py can be used to calculate the average electrostatic potential along the z-axis and outputs it into ElecStaticPot_AVE. Please note that the total local potential refers to the local component of the self-consistent potential, excluding the non-local pseudopotential. The distinction between the local potential and the electrostatic potential is as follows: local potential = electrostatic potential + XC potential.
  - 3: Apart from 1, also output the total local potential of the initial charge density. The files are named as:
  - nspin = 1: pots1_ini.cube;
  - nspin = 2: pots1_ini.cube and pots2_ini.cube;
  - nspin = 4: pots1_ini.cube, pots2_ini.cube, pots3_ini.cube, and pots4_ini.cube

  The optional second integer controls the output precision. If not provided, the default precision is 8.

  In molecular dynamics calculations, the output frequency is controlled by out_freq_ion.

  Note: In the 3.10-LTS version, the file names are SPIN1_POT.cube and SPIN1_POT_INI.cube, etc.
- **Default**: 0

out_dmk

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to output the density matrix for each k-point into files in the folder OUT.${suffix}. For current develop versions, out_dmk writes *_nao.txt files and includes a g{istep} index in the file name:
  - For gamma only case:
  - nspin = 1 and 4: dmg1_nao.txt;
  - nspin = 2: dms1g1_nao.txt and dms2g1_nao.txt for the two spin channels.
  - For multi-k points case:
  - nspin = 1 and 4: dmk1g1_nao.txt, dmk2g1_nao.txt, …;
  - nspin = 2: dmk1s1g1_nao.txt… and dmk1s2g1_nao.txt… for the two spin channels. Here, g{istep} denotes the geometry/step index in the output file name.

    Note: Version difference (develop vs 3.10-LTS):
  - In develop, out_dmk supports both gamma-only and multi-k-point density-matrix output.
  - In 3.10-LTS, the corresponding keyword is out_dm, and the output files are SPIN1_DM and SPIN2_DM, etc.
- **Default**: False

out_dmr

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis (multi-k points)*
- **Description**: Whether to output the density matrix with Bravias lattice vector R index into files in the folder OUT.${suffix}. The files are named as dmr{s}{spin index}{g}{geometry index}{_nao} + {".csr"}. Here, 's' refers to spin, where s1 means spin up channel while s2 means spin down channel, and the sparse matrix format 'csr' is mentioned in out_mat_hs2. Finally, if out_app_flag is set to false, the file name contains the optional 'g' index for each ionic step that may have different geometries, and if out_app_flag is set to true, the density matrix with respect to Bravias lattice vector R accumulates during ionic steps:
  - nspin = 1: dmrs1_nao.csr;
  - nspin = 2: dmrs1_nao.csr and dmrs2_nao.csr for the two spin channels.

    Note: In the 3.10-LTS version, the parameter is named out_dm1, and the file names are data-DMR-sparse_SPIN0.csr and data-DMR-sparse_SPIN1.csr, etc.
- **Default**: False

out_wfc_pw

- **Type**: Integer
- **Availability**: *Output electronic wave functions in plane wave basis, or transform the real-space electronic wave function into plane wave basis (see get_wf option in calculation with NAO basis)*
- **Description**: Whether to output the electronic wavefunction coefficients into files and store them in the folder OUT.${suffix}. The files are named as wf{k}{k-point index}{s}{spin index}{g}{geometry index}{e}{electronic iteration index}{_pw} + {".txt"/".dat"}. Here, the s index refers to spin but the label will not show up for non-spin-polarized calculations, where s1 means spin up channel while s2 means spin down channel, and s4 refers to spinor wave functions that contains both spin channels with spin-orbital coupling or noncollinear calculations enabled. For scf or nscf calculations, g index will not appear, but the g index appears for geometry relaxation and molecular dynamics, where one can use the out_freq_ion command to control. To print out the electroinc wave functions every few SCF iterations, use the out_freq_elec command and the e index will appear in the file name.
  - 0: no output
  - 1: (txt format)
  - non-gamma-only with nspin=1: wfk1_pw.txt, wfk2_pw.txt, …;
  - non-gamma-only with nspin=2: wfk1s1_pw.txt, wfk1s2_pw.txt, wfk2s1_pw.txt, wfk2s2_pw.txt, …;
  - non-gamma-only with nspin=4: wfk1s4_pw.txt, wfk2s4_pw.txt, …;
  - 2: (binary format)
  - non-gamma-only with nspin=1: wfk1_pw.dat, wfk2_pw.dat, …;
  - non-gamma-only with nspin=2: wfk1s1_pw.dat, wfk1s2_pw.dat, wfk2s1_pw.dat, wfk2s2_pw.dat, …;
  - non-gamma-only with nspin=4: wfk1s4_pw.dat, wfk2s4_pw.dat, …;

    Note: In the 3.10-LTS version, the file names are WAVEFUNC1.dat, WAVEFUNC2.dat, etc.
- **Default**: 0

out_wfc_lcao

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to output the electronic wavefunction coefficients into files and store them in the folder OUT.${suffix}. The files are named as wf{s}{spin index}{k(optional)}{k-point index}{g(optional)}{geometry index1}{_nao} + {".txt"/".dat"}. Here, 's' refers to spin, where s1 means spin up channel while s2 means spin down channel, and 's12' refer to spinor wave functions that contains both spin channels with spin-orbital coupling or noncollinear calculations enabled. In addition, if 'gamma_only' is set to 0, then the optinoal k-point sampling index appears with the k-point index attached to the electronic wave function file names. Finally, if out_app_flag is set to false, the file name contains the optional 'g' index for each ionic step that may have different geometries, and if out_app_flag is set to true, the wave functions accumulate during ionic steps. If the out_app_flag is set to false, a new folder named WFC will be created, and the wave function files will be saved into it.
  - 0: no output
  - 1: (txt format)
  - gamma-only: wfs1_nao.txt or wfs2_nao.txt, …;
  - non-gamma-only: wfs1k1_nao.txt or wfs1k2_nao.txt, …;
  - 2: (binary format)
  - gamma-only: wfs1_nao.dat or wfs2_nao.dat, …;
  - non-gamma-only: wfs1k1_nao.dat or wfs1k2_nao.dat, …

  The corresponding sequence of the orbitals can be seen in Basis Set.

  Also controled by out_freq_ion and out_app_flag.

  > Note: In the 3.10-LTS version, the file names are WFC_NAO_GAMMA1_ION1.txt and WFC_NAO_K1_ION1.txt, etc.
- **Default**: 0

out_dos

- **Type**: Integer
- **Description**: Whether to output the density of states (DOS). For more information, refer to the dos.md.
  - 0: no output
  - 1: output the density of states (DOS)
  - nspin=1 or 4: doss1g{geom}_{basis}.txt, where geom is the geometry index when cell changes or ions move while basis is either pw or nao.
  - nspin=2: doss1g{geom}*{basis}.txt and doss2g{geom}*{basis}.txt for two spin channles.
  - 2: (LCAO) output the density of states (DOS) and the projected density of states (PDOS)
  - 3: output the Fermi surface file (fermi.bxsf) in BXSF format that can be visualized by XCrySDen
- **Default**: 0

out_ldos

- **Type**: Integer [Integer](optional)
- **Description**: Whether to output the local density of states (LDOS), optionally output precision can be set by a second parameter, default is 3.
  - 0: no output
  - 1: output the partial charge density for given bias (controlled by stm_bias) in cube file format, which can be used to plot scanning tunneling spectroscopys to mimick STM images using the Python script plot.py.
  - 2: output LDOS along a line in real space (controlled by ldos_line). Parameters used to control DOS output are also valid for LDOS.
  - 3: output both two LDOS modes above.
- **Default**: 0

out_band

- **Type**: Boolean [Integer](optional)
- **Description**: Whether to output the eigenvalues of the Hamiltonian matrix (in eV) into the running log during electronic iterations and into a file at the end of calculations. The former can be used with the 'out_freq_elec' parameter while the latter option allows the output precision to be set via a second parameter, with a default value of 8. The output file names are:
  - nspin = 1 or 4: eig.txt;
  - nspin = 2: eigs1.txt and eigs2.txt;
  - For more information, refer to the band.md
- **Default**: False

out_proj_band

- **Type**: Boolean
- **Description**: Whether to output the projected band structure. For more information, refer to the band.md
- **Default**: False

out_stru

- **Type**: Boolean
- **Description**: Whether to output structure files per ionic step in geometry relaxation calculations into OUT.{istep}_D, where ${istep} is the ionic step.
- **Default**: False

out_level

- **Type**: String
- **Description**: Control the output level of information in OUT.{calculation}.log.
  - ie: electronic iteration level, which prints useful information for electronic iterations;
  - i: geometry relaxation level, which prints some information for geometry relaxations additionally;
  - m: molecular dynamics level, which does not print some information for simplicity.

- **Default**: ie

out_mat_hs

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to print the upper triangular part of the Hamiltonian matrices and overlap matrices for each k-point into files in the directory OUT.${suffix}. The second number controls precision. For more information, please refer to hs_matrix.md. Also controled by out_freq_ion and out_app_flag.
  - For gamma only case:
  - nspin = 1: hks1_nao.txt for the Hamiltonian matrix and sks1_nao.txt for the overlap matrix;
  - nspin = 2: hks1_nao.txt and hks2_nao.txt for the Hamiltonian matrix and sks1_nao.txt for the overlap matrix. Note that the code will not output sks2_nao.txt because it is the same as sks1_nao.txt;
  - nspin = 4: hks12_nao.txt for the Hamiltonian matrix and sks12_nao.txt for the overlap matrix.
  - For multi-k points case:
  - nspin = 1: hks1k1_nao.txt for the Hamiltonian matrix at the 1st k-point, and sks1k1_nao.txt for the overlap matrix for the 1st k-point, …;
  - nspin = 2: hks1k1_nao.txt and hks2k1_nao.txt for the two spin channels of the Hamiltonian matrix at the 1st k-point, and sks1k1_nao.txt for the overlap matrix for the 1st k-point. Note that the code will not output sks2k1_nao.txt because it is the same as sks1k1_nao.txt, …;
  - nspin = 4: hks12k1_nao.txt for the Hamiltonian matrix at the 1st k-point, and sks12k1_nao.txt for the overlap matrix for the 1st k-point, …;

    Note: In the 3.10-LTS version, the file names are data-0-H and data-0-S, etc.
- **Default**: False 8
- **Unit**: Ry

out_mat_hs2

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Whether to print files containing the Hamiltonian matrix and overlap matrix into files in the directory OUT.${suffix}. For more information, please refer to hs_matrix.md.

  Note: In the 3.10-LTS version, the file names are data-HR-sparse_SPIN0.csr and data-SR-sparse_SPIN0.csr, etc.
- **Default**: False [8]
- **Unit**: Ry

out_mat_tk

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to print the upper triangular part of the kinetic matrices for each k-point into OUT.${suffix}/tks1ki_nao.txt, where i is the index of k points. One may optionally provide a second parameter to specify the precision.

Note: In the 3.10-LTS version, the file names are data-TR-sparse_SPIN0.csr, etc.

- **Default**: False [8]
- **Unit**: Ry

out_mat_r

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Whether to print the matrix representation of the position matrix into files named rxrs1_nao.csr, ryrs1_nao.csr, rzrs1_nao.csr in the directory OUT.${suffix}. If calculation is set to get_s, the position matrix can be obtained without scf iterations. For more information, please refer to position_matrix.md.

  Note: In the 3.10-LTS version, the file name is data-rR-sparse.csr.

- **Default**: False 8
- **Unit**: Bohr

out_mat_t

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Generate files containing the kinetic energy matrix. The format will be the same as the Hamiltonian matrix and overlap matrix as mentioned in out_mat_hs2. The name of the files will be trs1_nao.csr and so on. Also controled by out_freq_ion and out_app_flag.

  Note: In the 3.10-LTS version, the file name is data-TR-sparse_SPIN0.csr.

- **Default**: False 8
- **Unit**: Ry

out_mat_dh

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Whether to print files containing the derivatives of the Hamiltonian matrix. The format will be the same as the Hamiltonian matrix and overlap matrix as mentioned in out_mat_hs2. The name of the files will be dhrxs1_nao.csr, dhrys1_nao.csr, dhrzs1_nao.csr and so on. Also controled by out_freq_ion and out_app_flag.

  Note: In the 3.10-LTS version, the file name is data-dHRx-sparse_SPIN0.csr and so on.

- **Default**: 0 8
- **Unit**: Ry/Bohr

out_mat_ds

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Whether to print files containing the derivatives of the overlap matrix. The format will be the same as the overlap matrix as mentioned in out_mat_dh. The name of the files will be dsxrs1_nao.csr and so on. Also controled by out_freq_ion and out_app_flag. This feature can be used with calculation get_s.

  Note: In the 3.10-LTS version, the file name is data-dSRx-sparse_SPIN0.csr and so on.

- **Default**: False 8
- **Unit**: Ry/Bohr

out_mat_xc

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital (NAO) and NAO-in-PW basis*
- **Description**: Whether to print the upper triangular part of the exchange-correlation matrices in Kohn-Sham orbital representation: for each k point into files in the directory OUT.i_nao.txt, where {suffix}/vxc_out.dat. If EXX is calculated, the local and EXX part of band energy will also be printed in OUT.{suffix}/vxc_exx_out.dat, respectively. All the vxc_out.dat files contains 3 integers (nk, nspin, nband) followed by nk*nspin*nband lines of energy Hartree and eV.

  Note: In the 3.10-LTS version, the file name is k-$k-Vxc and so on.

- **Default**: False
- **Unit**: Ry

out_mat_xc2

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital (NAO) basis*
- **Description**: Whether to print the exchange-correlation matrices in numerical orbital representation: in CSR format in the directory OUT.${suffix}. The name of the files will be vxcrs1_nao.csr and so on.

  Note: In the 3.10-LTS version, the file name is Vxc_R_spin$s and so on.

- **Default**: False 8
- **Unit**: Ry

out_mat_l

- **Type**: Boolean [Integer](optional)
- **Availability**: *Numerical atomic orbital (NAO) basis*
- **Description**: Whether to print the expectation value of the angular momentum operator , , and in the basis of the localized atomic orbitals. The files are named OUT.{suffix}_Lx.dat, OUT.{suffix}_Ly.dat, and OUT.{suffix}_Lz.dat. The second integer controls the precision of the output.
- **Default**: False 8

out_xc_r

- **Type**: Integer [Integer](optional)
- **Description**: The first integer controls whether to output the exchange-correlation (in Bohr^-3) on real space grids using Libxc to folder OUT.${suffix}:
  - 0: rho, amag, sigma, exc
  - 1: vrho, vsigma
  - 2: v2rho2, v2rhosigma, v2sigma2
  - 3: v3rho3, v3rho2sigma, v3rhosigma2, v3sigma3

  - 4: v4rho4, v4rho3sigma, v4rho2sigma2, v4rhosigma3, v4sigma4 The meaning of the files is presented in Libxc

  The second integer controls the precision of the charge density output, if not given, will use 3 as default.

  The circle order of the charge density on real space grids is: x is the outer loop, then y and finally z (z is moving fastest).

- **Default**: -1 3

out_eband_terms

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to print the band energy terms separately in the file OUT.{term}_out.dat. The terms include the kinetic, pseudopotential (local + nonlocal), Hartree and exchange-correlation (including exact exchange if calculated).
- **Default**: False

out_mul

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to print the Mulliken population analysis result into OUT.${suffix}/mulliken.txt. In molecular dynamics calculations, the output frequency is controlled by out_freq_ion.
- **Default**: False

out_app_flag

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis (not gamma-only algorithm)*
- **Description**: Whether to output r®, H®, S®, T®, dH®, dS®, and wfc matrices in an append manner during molecular dynamics calculations. Check input parameters out_mat_r, out_mat_hs2, out_mat_t, out_mat_dh, out_mat_hs and out_wfc_lcao for more information.
- **Default**: true

out_ndigits

- **Type**: Integer
- **Availability**: *out_mat_hs 1 case presently.*
- **Description**: Controls the length of decimal part of output data, such as charge density, Hamiltonian matrix, Overlap matrix and so on.
- **Default**: 8

out_element_info

- **Type**: Boolean
- **Description**: Whether to print element information into files in the directory OUT.{element_label}, including pseudopotential and orbital information of the element (in atomic Ryberg units).
- **Default**: False

restart_save

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Whether to save charge density files per ionic step, which are used to restart calculations. According to the value of read_file_dir:
  - auto: These files are saved in folder OUT.{read_file_dir}/restart/.

  If EXX(exact exchange) is calculated (i.e. dft_fuctional==hse/hf/pbe0/scan0 or rpa==True), the Hexx® files for each processor will also be saved in the above folder, which can be read in EXX calculation with restart_load==True.
- **Default**: False

rpa

- **Type**: Boolean
- **Description**: Generate output files used in rpa calculations.

  Note: If symmetry is set to 1, additional files containing the necessary information for exploiting symmetry in the subsequent rpa calculation will be output: irreducible_sector.txt, symrot_k.txt and symrot_R.txt.
- **Default**: False

out_pchg

- **Type**: String
- **Availability**: *For both PW and LCAO. When basis_type = lcao, used when calculation = get_pchg.*
- **Description**: Specifies the electronic states to calculate the charge densities with state index for, using a space-separated string of 0s and 1s. Each digit in the string corresponds to a state, starting from the first state. A 1 indicates that the charge density should be calculated for that state, while a 0 means the state will be ignored. The parameter allows a compact and flexible notation (similar to ocp_set), for example the syntax 1 4*0 5*1 0 is used to denote the selection of states: 1 means calculate for the first state, 4*0 skips the next four states,* 5*1 means calculate for the following five states, and the final 0 skips the next state. It's essential that the total count of states does not exceed the total number of states (nbands); otherwise, it results in an error, and the process exits. The input string must contain only numbers and the asterisk (*) for repetition, ensuring correct format and intention of state selection. The outputs comprise multiple .cube files following the naming convention pchgi[state]s[spin]k[kpoint].cube.
- **Default**: none

out_wfc_norm

- **Type**: String
- **Availability**: *For both PW and LCAO. When basis_type = lcao, used when calculation = get_wf.*
- **Description**: Specifies the electronic states to calculate the real-space wave function modulus (norm, or known as the envelope function) with state index. The syntax and state selection rules are identical to out_pchg, but the output is the norm of the wave function. The outputs comprise multiple .cube files following the naming convention wfi[state]s[spin]k[kpoint].cube.
- **Default**: none

out_wfc_re_im

- **Type**: String
- **Availability**: *For both PW and LCAO. When basis_type = lcao, used when calculation = get_wf.*
- **Description**: Specifies the electronic states to calculate the real and imaginary parts of the wave function with state index. The syntax and state selection rules are identical to out_pchg, but the output contains both the real and imaginary components of the wave function. The outputs comprise multiple .cube files following the naming convention wfi[state]s[spin]k[kpoint][re/im].cube.
- **Default**: none

if_separate_k

- **Type**: Boolean
- **Availability**: *For both PW and LCAO. When basis_type = pw, used if out_pchg is set. When basis_type = lcao, used only when calculation = get_pchg and gamma_only = 0.*
- **Description**: Specifies whether to write the partial charge densities for all k-points to individual files or merge them. Warning: Enabling symmetry may produce unwanted results due to reduced k-point weights and symmetry operations in real space. Therefore when calculating partial charge densities, if you are not sure what you want exactly, it is strongly recommended to set symmetry = -1. It is noteworthy that your symmetry setting should remain the same as that in the SCF procedure.
- **Default**: false

out_elf

- **Type**: Integer [Integer](optional)
- **Availability**: *Only for Kohn-Sham DFT and Orbital Free DFT.*
- **Description**: Whether to output the electron localization function (ELF) in the folder `OUT.${suffix}`. The files are named as
  - nspin = 1:
    - elf.cube: $\mathrm{ELF} = \frac{1}{1+\chi^2}$, $\chi = \frac{\frac{1}{2}\sum_i f_i|\nabla\psi_i|^2 - \frac{|\nabla\rho|^2}{8\rho}}{\frac{3}{10}(3\pi^2)^{2/3}\rho^{5/3}}$;
  - nspin = 2:
    - elf1.cube, elf2.cube: $\mathrm{ELF}_\sigma = \frac{1}{1+\chi_\sigma^2}$, $\chi_\sigma = \frac{\frac{1}{2}\sum_i f_i|\nabla\psi_{i,\sigma}|^2 - \frac{|\nabla\rho_\sigma|^2}{8\rho_\sigma}}{\frac{3}{10}(6\pi^2)^{2/3}\rho_\sigma^{5/3}}$;
    - elf.cube: $\mathrm{ELF} = \frac{1}{1+\chi^2}$, $\chi = \frac{\frac{1}{2}\sum_{i,\sigma} f_i|\nabla\psi_{i,\sigma}|^2 - \sum_\sigma \frac{|\nabla\rho_\sigma|^2}{8\rho_\sigma}}{\sum_\sigma \frac{3}{10}(6\pi^2)^{2/3}\rho_\sigma^{5/3}}$;
  - nspin = 4 (noncollinear):
    - elf.cube: ELF for total charge density, $\mathrm{ELF} = \frac{1}{1+\chi^2}$, $\chi = \frac{\frac{1}{2}\sum_i f_i|\nabla\psi_i|^2 - \frac{|\nabla\rho|^2}{8\rho}}{\frac{3}{10}(3\pi^2)^{2/3}\rho^{5/3}}$

  The second integer controls the precision of the kinetic energy density output, if not given, will use 3 as default. For purpose restarting from this file and other high-precision involved calculation, recommend to use 10.

  In molecular dynamics calculations, the output frequency is controlled by out_freq_ion.
- **Default**: 0 3

out_spillage

- **Type**: Integer
- **Availability**: *Only for Kohn-Sham DFT with plane-wave basis.*
- **Description**: This output is only intentively needed by the ABACUS numerical atomic orbital generation workflow. This parameter is used to control whether to output the overlap integrals between truncated spherical Bessel functions (TSBFs) and plane-wave basis expanded wavefunctions (named as OVERLAP_Q), and between TSBFs (named as OVERLAP_Sq), also their first order derivatives. The output files are named starting with orb_matrix. A value of 2 would enable the output.
- **Default**: 0

out_alllog

- **Type**: Boolean
- **Description**: Whether to print information into individual logs from all ranks in an MPI run.
  - True: Information from each rank will be written into individual files named OUT.{calculation}*{suffix}/running*${calculation}.log.
- **Default**: False

back to top

15.1.9 Density of states

dos_edelta_ev

- **Type**: Real
- **Description**: The step size in writing Density of States (DOS)
- **Default**: 0.01
- **Unit**: eV

dos_sigma

- **Type**: Real
- **Description**: The width of the Gaussian factor when obtaining smeared Density of States (DOS)
- **Default**: 0.07
- **Unit**: eV

dos_scale

- **Type**: Real
- **Description**: Defines the energy range of DOS output as (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
- **Unit**: eV

dos_emin_ev

- **Type**: Real
- **Description**: The minimal range for Density of States (DOS)
  - If set, “dos_scale” will be ignored.
- **Default**: Minimal eigenenergy of
- **Unit**: eV

dos_emax_ev

- **Type**: Real
- **Description**: The maximal range for Density of States (DOS)
  - If set, “dos_scale” will be ignored.
- **Default**: Maximal eigenenergy of
- **Unit**: eV

dos_nche

- **Type**: Integer
- **Description**: The order of Chebyshev expansions when using Stochastic Density Functional Theory (SDFT) to calculate DOS.
- **Default**: 100

stm_bias

- **Type**: Real Real(optional) Integer(optional)
- **Description**: The bias voltage used to calculate local density of states to simulate scanning tunneling microscope, see details in out_ldos. When using three parameters:
  - The first parameter specifies the initial bias voltage value.
  - The second parameter defines the voltage increment (step size between consecutive bias values).
  - The third parameter determines the total number of voltage points
- **Default**: 1.0
- **Unit**: V

ldos_line

- **Type**: Real*6 Integer(optional)
- **Description**: Specify the path of the three-dimensional space and display LDOS in the form of a two-dimensional color chart, see details in out_ldos. The first three paramenters are the direct coordinates of the start point, the next three paramenters are the direct coordinates of the end point, and the final one is the number of points along the path, whose default is 100.
- **Default**: 0.0 0.0 0.0 0.0 0.0 1.0 100

back to top

15.1.10 NAOs

bessel_nao_ecut

- **Type**: String
- **Description**: "Energy cutoff" (in Ry) of spherical Bessel functions. The number of spherical Bessel functions that constitute the radial parts of NAOs is determined by sqrt(bessel_nao_ecut)*bessel_nao_rcut/.
- **Default**: ecutwfc

bessel_nao_tolerence

- **Type**: Real
- **Description**: Tolerance when searching for the zeros of spherical Bessel functions.
- **Default**: 1.0e-12

bessel_nao_rcut

- **Type**: Vector of Real (N values)
- **Description**: Cutoff radius (in Bohr) and the common node of spherical Bessel functions used to construct the NAOs.
- **Default**: 6.0

bessel_nao_smooth

- **Type**: Boolean
- **Description**: If True, NAOs will be smoothed near the cutoff radius. See bessel_nao_rcut and bessel_nao_sigma for parameters.
- **Default**: True

bessel_nao_sigma

- **Type**: Real
- **Description**: Smoothing range (in Bohr). See also bessel_nao_smooth.
- **Default**: 0.1

back to top

15.1.11 DeePKS

deepks_out_labels

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Print labels and descriptors for DeePKS in OUT.${suffix}. The names of these files start with "deepks".
  - 0 : No output.
  - 1 : Output intermediate files needed during DeePKS training.

  - 2 : Output target labels for label preperation. The label files are named as deepks_<property>.npy or deepks_<property>.csr, where the units and formats are the same as label files <property>.npy or <property>.csr required for training, except that the first dimension (nframes) is excluded. System structrue files are also given in deepks_atom.npy and deepks_box.npy in the unit of Bohr, which means lattice_constant should be set to 1 when training.

    Note: When deepks_out_labels equals 1, the path of a numerical descriptor (an orb file) is needed to be specified under the NUMERICAL_DESCRIPTOR tag in the STRU file. This is not needed when deepks_out_labels equals 2.

- **Default**: 0

deepks_out_freq_elec

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis*
- **Description**: When deepks_out_freq_elec is greater than 0, print labels and descriptors for DeePKS in OUT.${suffix}/DeePKS_Labels_Elec per deepks_out_freq_elec electronic iterations, with suffix _e* to distinguish different steps. Often used with deepks_out_labels equals 1.
- **Default**: 0

deepks_out_base

- **Type**: String
- **Availability**: *Numerical atomic orbital basis and deepks_out_freq_elec is greater than 0*
- **Description**: Print labels and descriptors calculated by base functional (determined by deepks_out_base) and target functional (determined by dft_functional) for DeePKS in per deepks_out_freq_elec electronic iterations. The SCF process, labels and descriptors output of the target functional are all consistent with those when the target functional is used alone. The only additional output under this configuration is the labels of the base functional. Often used with deepks_out_labels equals 1.
- **Default**: None

deepks_scf

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis*
- **Description**: perform self-consistent field iteration in DeePKS method

    Note: A trained, traced model file is needed.

- **Default**: False

deepks_equiv

- **Type**: Boolean
- **Availability**: *Numerical atomic orbital basis*
- **Description**: whether to use equivariant version of DeePKS

    Note: The equivariant version of DeePKS-kit is still under development, so this feature is currently only intended for internal usage.

- **Default**: False

deepks_model

- **Type**: String
- **Availability**: *Numerical atomic orbital basis and deepks_scf is true*
- **Description**: the path of the trained, traced neural network model file generated by deepks-kit
- **Default**: None

bessel_descriptor_lmax

- **Type**: Integer
- **Availability**: *gen_bessel calculation*
- **Description**: the maximum angular momentum of the Bessel functions generated as the projectors in DeePKS - NOte: To generate such projectors, set calculation type to gen_bessel in ABACUS. See also calculation.
- **Default**: 2

bessel_descriptor_ecut

- **Type**: String
- **Availability**: *gen_bessel calculation*
- **Description**: energy cutoff of Bessel functions
- **Default**: same as ecutwfc
- **Unit**: Ry

bessel_descriptor_tolerence

- **Type**: Real
- **Availability**: *gen_bessel calculation*
- **Description**: tolerance for searching the zeros of Bessel functions
- **Default**: 1.0e-12

bessel_descriptor_rcut

- **Type**: Real
- **Availability**: *gen_bessel calculation*
- **Description**: cutoff radius of Bessel functions
- **Default**: 6.0
- **Unit**: Bohr

bessel_descriptor_smooth

- **Type**: Boolean
- **Availability**: *gen_bessel calculation*
- **Description**: smooth the Bessel functions at radius cutoff
- **Default**: False

bessel_descriptor_sigma

- **Type**: Real
- **Availability**: *gen_bessel calculation*
- **Description**: smooth parameter at the cutoff radius of projectors
- **Default**: 0.1
- **Unit**: Bohr

deepks_bandgap

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis and deepks_scf is true*
- **Description**: include bandgap label for DeePKS training
  - 0: Don't include bandgap label
  - 1: Include target bandgap label (see deepks_band_range for more details)
  - 2: Include multiple bandgap label (see deepks_band_range for more details)
  - 3: Used for systems containing H atoms. Here HOMO is defined as the max occupation except H atoms and the bandgap label is the energy between HOMO and (HOMO + 1)
- **Default**: 0

deepks_band_range

- **Type**: Integer*2
- **Availability**: *Numerical atomic orbital basis, deepks_scf is true, and deepks_bandgap is 1 or 2*
- **Description**: The first value should not be larger than the second one and the meaning differs in different cases below
  - deepks_bandgap is 1: Bandgap label is the energy between LUMO + deepks_band_range[0] and LUMO + deepks_band_range[1]. If not set, it will calculate energy between HOMO and LUMO states.
  - deepks_bandgap is 2: Bandgap labels are energies between HOMO and all states in range [LUMO + deepks_band_range[0], LUMO + deepks_band_range[1]] (Thus there are deepks_band_range[1] - deepks_band_range[0] + 1 bandgaps in total). If HOMO is included in the setting range, it will be ignored since it will always be zero and has no valuable messages (deepks_band_range[1] - deepks_band_range[0] bandgaps in this case). NOTICE: The set range can be greater than, less than, or include the value of HOMO. In the bandgap label, we always calculate the energy of the state in the set range minus the energy of HOMO state, so the bandgap can be negative if the state is lower than HOMO.
- **Default**: -1 0

deepks_v_delta

- **Type**: Integer
- **Availability**: *Numerical atomic orbital basis*
- **Description**: Include V_delta/V_delta_R (Hamiltonian in k/real space) label for DeePKS training. When deepks_out_labels is true and deepks_v_delta > 0 (k space), ABACUS will output deepks_hbase.npy, deepks_vdelta.npy and deepks_htot.npy(htot=hbase+vdelta). When deepks_out_labels is true and deepks_v_delta < 0 (real space), ABACUS will output deepks_hrtot.csr, deepks_hrdelta.csr. Some more files output for different

settings. NOTICE: To match the unit Normally used in DeePKS, the unit of Hamiltonian in k space is Hartree. However, currently in R space the unit is still Ry.

- deepks_v_delta = 1: deepks_vdpre.npy, which is used to calculate V_delta during DeePKS training.
- deepks_v_delta = 2: deepks_phialpha.npy and deepks_gevdm.npy, which can be used to calculate deepks_vdpre.npy. A recommanded method for memory saving.
- deepks_v_delta = -1: deepks_vdrpre.npy, which is used to calculate V_delta_R during DeePKS training.
- deepks_v_delta = -2: deepks_phialpha_r.npy and deepks_gevdm.npy, which can be used to calculate deepks_vdrpre.npy. A recommanded method for memory saving.

- **Default**: 0

deepks_out_unittest

- **Type**: Boolean
- **Description**: generate files for constructing DeePKS unit test

  Note: Not relevant when running actual calculations. When set to 1, ABACUS needs to be run with only 1 process.

- **Default**: False

back to top

15.1.12 OFDFT: orbital free density functional theory

of_kinetic

- **Type**: String
- **Availability**: *OFDFT*
- **Description**: Kinetic energy functional type:
  - tf: Thomas-Fermi (TF) functional
  - vw: von Weizsacker (vW) functional
  - tf+: TF + vW functional
  - wt: Wang-Teter (WT) functional
  - ext-wt: Extended Wang-Teter functional
  - xwm: XWM functional
  - lkt: Luo-Karasiev-Trickey (LKT) functional
  - ml: Machine learning KEDF
  - mpn: MPN KEDF (automatically sets ml parameters)
  - cpn5: CPN5 KEDF (automatically sets ml parameters)
- **Default**: wt

of_method

- **Type**: String
- **Availability**: *OFDFT*
- **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
- **Availability**: *OFDFT*
- **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**: Real
- **Availability**: *OFDFT*
- **Description**: Tolerance of the energy change for determining the convergence.
- **Default**: 2e-6
- **Unit**: Ry

of_tolp

- **Type**: Real
- **Availability**: *OFDFT*
- **Description**: Tolerance of potential for determining the convergence.
- **Default**: 1e-5
- **Unit**: Ry

of_tf_weight

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=tf, tf+, wt, ext-wt, xwm*
- **Description**: Weight of TF KEDF (kinetic energy density functional).
- **Default**: 1.0

of_vw_weight

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=vw, tf+, wt, ext-wt, lkt, xwm*
- **Description**: Weight of vW KEDF (kinetic energy density functional).
- **Default**: 1.0

of_wt_alpha

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=wt, ext-wt*
- **Description**: Parameter alpha of WT KEDF (kinetic energy density functional).

of_wt_beta

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=wt, ext-wt*
- **Description**: Parameter beta of WT KEDF (kinetic energy density functional).

of_extwt_kappa

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=ext-wt*
- **Description**: Parameter kappa for EXT-WT KEDF.
- **Default**: 1.0 / (2.0 * std::pow(4./3., 1./3.) - 1.0)

of_wt_rho0

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=wt*
- **Description**: The average density of system.
- **Default**: 0.0
- **Unit**: Bohr^-3

of_hold_rho0

- **Type**: Boolean
- **Availability**: *OFDFT with of_kinetic=wt*
- **Description**: Whether to fix the average density rho0.
  - True: rho0 will be fixed even if the volume of system has changed, it will be set to True automatically if of_wt_rho0 is not zero.
  - False: rho0 will change if volume of system has changed.
- **Default**: False

of_lkt_a

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=lkt*
- **Description**: Parameter a of LKT KEDF (kinetic energy density functional).
- **Default**: 1.3

of_xwm_rho_ref

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=xwm*
- **Description**: Reference charge density for XWM kinetic energy functional. If set to 0, the program will use average charge density.
- **Default**: 0.0

of_xwm_kappa

- **Type**: Real
- **Availability**: *OFDFT with of_kinetic=xwm*
- **Description**: Parameter for XWM kinetic energy functional. See PHYSICAL REVIEW B 100, 205132 (2019) for optimal values.
- **Default**: 0.0

of_read_kernel

- **Type**: Boolean
- **Availability**: *OFDFT with of_kinetic=wt*
- **Description**: Whether to read in the kernel file.
  - True: The kernel of WT KEDF (kinetic energy density functional) will be filled from the file specified by of_kernel_file.
  - False: The kernel of WT KEDF (kinetic energy density functional) will be filled from formula.
- **Default**: False

of_kernel_file

- **Type**: String
- **Availability**: *OFDFT with of_read_kernel=True*
- **Description**: The name of WT kernel file.
- **Default**: WTkernel.txt

of_full_pw

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Whether to use full planewaves.
  - True: Ecut will be ignored while collecting planewaves, so that all planewaves will be used in FFT.
  - False: Only use the planewaves inside ecut, the same as KSDFT.
- **Default**: True

of_full_pw_dim

- **Type**: Integer
- **Availability**: *OFDFT with of_full_pw = True*
- **Description**: Specify the parity of FFT dimensions.
  - 0: either odd or even.
  - 1: odd only.
  - 2: even only.

  Note: 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 please set of_full_pw_dim = 1 if nbspline != -1.
- **Default**: 0

back to top

15.1.13 ML-KEDF: machine learning based kinetic energy density functional for OFDFT

of_ml_gene_data

- **Type**: Boolean
- **Availability**: *Used only for KSDFT with plane wave basis*
- **Description**: Controls the generation of machine learning training data. When enabled, training data in .npy format will be saved in the directory OUT.${suffix}/.
- **Default**: False

of_ml_device

- **Type**: String
- **Availability**: *OFDFT*
- **Description**: Run Neural Network on GPU or CPU.
  - cpu: CPU
  - gpu: GPU
- **Default**: cpu

of_ml_feg

- **Type**: Integer
- **Availability**: *OFDFT*
- **Description**: The method to incorporate the Free Electron Gas (FEG) limit.
  - 0: Do not incorporate the FEG limit.
  - 1: Incorporate the FEG limit by translation.
  - 3: Incorporate the FEG limit by nonlinear transformation using softplus function.
- **Default**: 0

of_ml_nkernel

- **Type**: Integer
- **Availability**: *OFDFT*
- **Description**: Number of kernel functions.
- **Default**: 1

of_ml_kernel

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the type of the i-th kernel function.
  - 1: Wang-Teter kernel function.
  - 2: Modified Yukawa function, and alpha is specified by of_ml_yukawa_alpha.
  - 3: Truncated kinetic kernel (TKK), the file containing TKK is specified by of_ml_kernel_file.
- **Default**: 1

of_ml_kernel_scaling

- **Type**: Vector of Real
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the RECIPROCAL of scaling parameter of the i-th kernel function.
- **Default**: 1.0

of_ml_yukawa_alpha

- **Type**: Vector of Real
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the parameter alpha of i-th kernel function. ONLY used for Yukawa kernel function.
- **Default**: 1.0

of_ml_kernel_file

- **Type**: Vector of String
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the file containing the i-th kernel function. ONLY used for TKK.
- **Default**: none

of_ml_gamma

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Local descriptor: gamma = (rho / rho0)^(1/3).
- **Default**: False

of_ml_p

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Semi-local descriptor: p = |nabla rho|^2 / [2 (3 pi^2)^(1/3) rho^(4/3)]^2.
- **Default**: False

of_ml_q

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Semi-local descriptor: q = nabla^2 rho / [4 (3 pi^2)^(2/3) rho^(5/3)].
- **Default**: False

of_ml_tanhp

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Semi-local descriptor: tanhp = tanh(chi_p * p).
- **Default**: False

of_ml_tanhq

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: Semi-local descriptor: tanhq = tanh(chi_q * q).
- **Default**: False

of_ml_chi_p

- **Type**: Real
- **Availability**: *OFDFT*
- **Description**: Hyperparameter chi_p: tanhp = tanh(chi_p * p).
- **Default**: 1.0

of_ml_chi_q

- **Type**: Real
- **Availability**: *OFDFT*
- **Description**: Hyperparameter chi_q: tanhq = tanh(chi_q * q).
- **Default**: 1.0

of_ml_gammanl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor gammanl defined by the i-th kernel function.
- **Default**: 0

of_ml_pnl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor pnl defined by the i-th kernel function.
- **Default**: 0

of_ml_qnl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor qnl defined by the i-th kernel function.
- **Default**: 0

of_ml_xi

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor xi defined by the i-th kernel function.
- **Default**: 0

of_ml_tanhxi

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanhxi defined by the i-th kernel function.
- **Default**: 0

of_ml_tanhxi_nl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanhxi_nl defined by the i-th kernel function.
- **Default**: 0

of_ml_tanh_pnl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanh_pnl defined by the i-th kernel function.
- **Default**: 0

of_ml_tanh_qnl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanh_qnl defined by the i-th kernel function.
- **Default**: 0

of_ml_tanhp_nl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanhp_nl defined by the i-th kernel function.
- **Default**: 0

of_ml_tanhq_nl

- **Type**: Vector of Integer
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element controls the non-local descriptor tanhq_nl defined by the i-th kernel function.
- **Default**: 0

of_ml_chi_xi

- **Type**: Vector of Real
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the hyperparameter chi_xi of non-local descriptor tanhxi defined by the i-th kernel function.
- **Default**: 1.0

of_ml_chi_pnl

- **Type**: Vector of Real
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the hyperparameter chi_pnl of non-local descriptor tanh_pnl defined by the i-th kernel function.
- **Default**: 1.0

of_ml_chi_qnl

- **Type**: Vector of Real
- **Availability**: *OFDFT*
- **Description**: Containing nkernel (see of_ml_nkernel) elements. The i-th element specifies the hyperparameter chi_qnl of non-local descriptor tanh_qnl defined by the i-th kernel function.
- **Default**: 1.0

of_ml_local_test

- **Type**: Boolean
- **Availability**: *OFDFT*
- **Description**: FOR TEST. Read in the density, and output the F and Pauli potential.
- **Default**: False

ml_exx

- **Type**: Boolean
- **Description**: Whether to use machine learning based exact exchange (ML-EXX).
- **Default**: False

back to top

15.1.14 TDOFDFT: time dependent orbital free density functional theory

of_cd

- **Type**: Boolean
- **Availability**: *TDOFDFT*
- **Description**: Added the current dependent(CD) potential. (https://doi.org/10.1103/PhysRevB.98.144302)
  - True: Added the CD potential.
  - False: Not added the CD potential.
- **Default**: False

of_mcd_alpha

- **Type**: Real
- **Availability**: *TDOFDFT*

- **Description**: The value of the parameter alpha in modified CD potential method. mCDPotential=alpha*CDPotential (proposed in paper PhysRevB.98.144302)
- **Default**: 1.0

back to top

15.1.15 Electric field and dipole correction

efield_flag

- **Type**: Boolean
- **Description**: Added the electric field.
  - True: A saw-like potential simulating an electric field is added to the bare ionic potential.
  - False: Not added the electric field.
- **Default**: False

dip_cor_flag

- **Type**: Boolean
- **Availability**: *With dip_cor_flag = True and efield_flag = True.*
- **Description**: Added a dipole correction to the bare ionic potential.
  - True: A dipole correction is also added to the bare ionic potential.
  - False: A dipole correction is not added to the bare ionic potential.

    Note: If you do not want any electric field, the parameter efield_amp should be set to zero. This should ONLY be used in a slab geometry for surface calculations, with the discontinuity FALLING IN THE EMPTY SPACE.
- **Default**: False

efield_dir

- **Type**: Integer
- **Availability**: *with efield_flag = True.*
- **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 can set to 0, 1 or 2.
  - 0: parallel to the first reciprocal lattice vector
  - 1: parallel to the second reciprocal lattice vector
  - 2: parallel to the third reciprocal lattice vector
- **Default**: 2

efield_pos_max

- **Type**: Real
- **Availability**: *with efield_flag = True.*
- **Description**: Position of the maximum of the saw-like potential along crystal axis efield_dir, within the unit cell, 0 <= efield_pos_max < 1.
- **Default**: Autoset to center of vacuum - width of vacuum / 20

efield_pos_dec

- **Type**: Real
- **Availability**: *with efield_flag = True.*
- **Description**: Zone in the unit cell where the saw-like potential decreases, 0 < efield_pos_dec < 1.
- **Default**: Autoset to width of vacuum / 10

efield_amp

- **Type**: Real
- **Availability**: *with efield_flag = True.*
- **Description**: Amplitude of the electric field. 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.

  Note: The change of slope of this potential must be located in the empty region, or else unphysical forces will result.

- **Default**: 0.0
- **Unit**: a.u., 1 a.u. = 51.4220632*10^10 V/m.

back to top

15.1.16 Gate field (compensating charge)

gate_flag

- **Type**: Boolean
- **Description**: Controls the addition of compensating charge by a charged plate for charged cells.
  - true: A charged plate is placed at the zgate position to add compensating charge. The direction is determined by efield_dir.
  - false: No compensating charge is added.
- **Default**: false

zgate

- **Type**: Real
- **Description**: Position of the charged plate in the unit cell
- **Default**: 0.5
- **Unit**: Unit cell size

block

- **Type**: Boolean
- **Description**: Controls the addition of a potential barrier to prevent electron spillover.
  - true: A potential barrier is added from block_down to block_up with a height of block_height. If dip_cor_flag is set to true, efield_pos_dec is used to smoothly increase and decrease the potential barrier.
  - false: No potential barrier is added.
- **Default**: false

block_down

- **Type**: Real
- **Description**: Lower beginning of the potential barrier
- **Default**: 0.45
- **Unit**: Unit cell size

block_up

- **Type**: Real
- **Description**: Upper beginning of the potential barrier
- **Default**: 0.55
- **Unit**: Unit cell size

block_height

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

back to top

15.1.17 Exact Exchange (Common)

exx_fock_alpha

- **Type**: Real
- **Description**: Fraction of full-ranged Fock exchange $1/r$ in range-separated hybrid functionals.
- **Default**: see hybrid_func_params

exx_erfc_alpha

- **Type**: Real
- **Description**: Fraction of short-ranged Fock exchange $\mathrm{erfc}(\omega r)/r$ in range-separated hybrid functionals.
- **Default**: see hybrid_func_params

exx_erfc_omega

- **Type**: Real
- **Description**: Range-separation parameter ω in the short-ranged Fock term $\mathrm{erfc}(\omega r)/r$.
- **Default**: see hybrid_func_params

exx_separate_loop

- **Type**: Boolean
- **Description**: There are two types of iterative approaches provided by ABACUS to evaluate Fock exchange.
  - False: Start with a GGA-Loop, and then Hybrid-Loop, in which EXX Hamiltonian is updated with electronic iterations.
  - True: A two-step method is employed, i.e. in the inner iterations, density matrix is updated, while in the outer iterations, is calculated based on density matrix that converges in the inner iteration.
- **Default**: True

exx_hybrid_step

- **Type**: Integer
- **Availability**: *exx_separate_loop==1*
- **Description**: The maximal iteration number of the outer-loop, where the Fock exchange is calculated
- **Default**: 100

exx_mixing_beta

- **Type**: Real
- **Availability**: *exx_separate_loop==1*
- **Description**: Mixing parameter for densty matrix in each iteration of the outer-loop
- **Default**: 1.0

back to top

15.1.18 Exact Exchange (LCAO in PW)

exx_fock_lambda

- **Type**: Real
- **Availability**: *basis_type==lcao_in_pw*
- **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

back to top

15.1.19 Exact Exchange (LCAO)

exx_pca_threshold

- **Type**: Real
- **Description**: To accelerate the evaluation of four-center integrals (), the product of atomic orbitals are expanded in the basis of auxiliary basis functions (ABF): . The size of the ABF (i.e. number of) 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 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_cs_inv_thr

- **Type**: Real
- **Description**: By default, the Coulomb matrix inversion required for obtaining LRI coefficients is performed using LU decomposition. However, this approach may suffer from numerical instabilities when a large set of auxiliary basis functions (ABFs) is employed. When exx_cs_inv_thr > 0, the inversion is instead carried out via matrix diagonalization. Eigenvalues smaller than exx_cs_inv_thr are discarded to improve numerical stability. A relatively safe and commonly recommended value is 1e-5.
- **Default**: -1

exx_v_threshold

- **Type**: Real
- **Description**: See also the entry exx_pca_threshold. With the approximation , the four-center integral in Fock exchange is expressed as , where 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 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. is used in force. Smaller components (less than exx_c_grad_threshold) of the 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 , the four-center integral in Fock exchange is expressed as , where is a double-center integral. is used in force. 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_c_grad_r_threshold

- **Type**: Real
- **Description**: See also the entry exx_pca_threshold. is used in stress. Smaller components (less than exx_c_grad_r_threshold) of the 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_r_threshold

- **Type**: Real
- **Description**: See also the entry exx_pca_threshold. With the approximation , the four-center integral in Fock exchange is expressed as , where is a double-center integral. 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_ccp_rmesh_times

- **Type**: String
- **Description**: This parameter determines how many times larger the radial mesh required for calculating Columb potential is to that of atomic orbitals. The value should be larger than 0. Reducing this value can effectively increase the speed of self-consistent calculations using hybrid functionals.

exx_opt_orb_lmax

- **Type**: Integer
- **Availability**: *calculation==gen_opt_abfs*
- **Description**: The maximum l of the spherical Bessel functions, when the radial part of opt-ABFs are generated as linear combinations of spherical Bessel functions. A reasonable choice is 2.
- **Default**: 0

exx_opt_orb_ecut

- **Type**: Real
- **Availability**: *calculation==gen_opt_abfs*
- **Description**: The cut-off of plane wave expansion, when the plane wave basis is used to optimize the radial ABFs. A reasonable choice is 60.
- **Default**: 0
- **Unit**: Ry

exx_opt_orb_tolerence

- **Type**: Real
- **Availability**: *calculation==gen_opt_abfs*
- **Description**: The threshold when solving for the zeros of spherical Bessel functions. A reasonable choice is 1e-12.

- **Default**: 1E-12

exx_real_number

- **Type**: String
- **Description**: - True: Enforce LibRI to use double data type.
  - False: Enforce LibRI to use complex data type. Setting it to True can effectively improve the speed of self-consistent calculations with hybrid functionals.
- **Default**: depends on the gamma_only option

exx_singularity_correction

- **Type**: String
- **Description**: - spencer: see Phys. Rev. B 77, 193110 (2008).
  - revised_spencer: see Phys. Rev. Mater. 5, 013807 (2021). Set the scheme of Coulomb singularity correction.
- **Default**: default

rpa_ccp_rmesh_times

- **Type**: Real
- **Description**: How many times larger the radial mesh required is to that of atomic orbitals in the postprocess calculation of the bare Coulomb matrix for RPA, GW, etc.
- **Default**: 10

exx_symmetry_realspace

- **Type**: Boolean
- **Availability**: *symmetry==1 and exx calculation (dft_fuctional==hse/hf/pbe0/scan0 or rpa==True)*
- **Description**: - False: only rotate k-space density matrix D(k) from irreducible k-points to accelerate diagonalization
  - True: rotate both D(k) and Hexx® to accelerate both diagonalization and EXX calculation
- **Default**: True

out_ri_cv

- **Type**: Boolean
- **Description**: Whether to output the coefficient tensor C® and ABFs-representation Coulomb matrix V® for each atom pair and cell in real space.
- **Default**: false

back to top

15.1.20 Exact Exchange (PW)

exxace

- **Type**: Boolean
- **Availability**: *exx_separate_loop==True.*

- **Description**: Whether to use the ACE method (https://doi.org/10.1021/acs.jctc.6b00092) to accelerate the calculation the Fock exchange matrix. Should be set to true most of the time.
  - True: Use the ACE method to calculate the Fock exchange operator.
  - False: Use the traditional method to calculate the Fock exchange operator.
- **Default**: True

exx_gamma_extrapolation

- **Type**: Boolean
- **Description**: Whether to use the gamma point extrapolation method to calculate the Fock exchange operator. See https://doi.org/10.1103/PhysRevB.79.205114 for details. Should be set to true most of the time.
- **Default**: True

ecutexx

- **Type**: Real
- **Description**: The energy cutoff for EXX (Fock) exchange operator in plane wave basis calculations. Reducing ecutexx below ecutrho may significantly accelerate EXX computations. This speed improvement comes with a reduced numerical accuracy in the exchange energy calculation.
- **Default**: same as ecutrho
- **Unit**: Ry

exx_thr_type

- **Type**: String
- **Description**: The type of threshold used to judge whether the outer loop has converged in the separate loop EXX calculation.
  - energy: use the change of exact exchange energy to judge convergence.
  - density: if the change of charge density difference between two successive outer loop iterations is seen as converged according to scf_thr, then the outer loop is seen as converged.
- **Default**: density

exx_ene_thr

- **Type**: Real
- **Availability**: *exx_thr_type==energy*
- **Description**: The threshold for the change of exact exchange energy to judge convergence of the outer loop in the separate loop EXX calculation.
- **Default**: 1e-5
- **Unit**: Ry

back to top

15.1.21 Molecular dynamics

md_type

- **Type**: String
- **Description**: Control the algorithm to integrate the equation of motion for molecular dynamics (MD), see md.md in detail.
  - fire: a MD-based relaxation algorithm, named fast inertial relaxation engine.
  - nve: NVE ensemble with velocity Verlet algorithm.
  - nvt: NVT ensemble, see md_thermostat in detail.
  - npt: Nose-Hoover style NPT ensemble, see md_pmode in detail.
  - langevin: NVT ensemble with Langevin thermostat, see md_damp in detail.
  - msst: MSST method, see msst_direction, msst_vel, msst_qmass, msst_vis, msst_tscale in detail.
- **Default**: nvt

md_nstep

- **Type**: Integer
- **Description**: The total number of molecular dynamics steps.
- **Default**: 10

md_dt

- **Type**: Real
- **Description**: The time step used in molecular dynamics calculations.
- **Default**: 1.0
- **Unit**: fs

md_thermostat

- **Type**: String
- **Description**: Specify the temperature control method used in NVT ensemble.
  - nhc: Nose-Hoover chain, see md_tfreq and md_tchain in detail.
  - anderson: Anderson thermostat, see md_nraise in detail.
  - berendsen: Berendsen thermostat, see md_nraise in detail.
  - rescaling: velocity Rescaling method 1, see md_tolerance in detail.
  - rescale_v: velocity Rescaling method 2, see md_nraise in detail.
- **Default**: nhc

md_tfirst

- **Type**: Real

- **Description**: The temperature used in molecular dynamics calculations.

  If md_tfirst is unset or less than zero, init_vel is autoset to be true. If init_vel is true, the initial temperature will be determined by the velocities read from STRU. In this case, if velocities are unspecified in STRU, the initial temperature is set to zero.

  If md_tfirst is set to a positive value and init_vel is true simultaneously, please make sure they are consistent, otherwise abacus will exit immediately.

  Note that md_tlast is only used in NVT/NPT simulations. If md_tlast is unset or less than zero, md_tlast is set to 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
- **Unit**: K

md_tlast

- **Type**: Real
- **Description**: The temperature used in molecular dynamics calculations.

  If md_tfirst is unset or less than zero, init_vel is autoset to be true. If init_vel is true, the initial temperature will be determined by the velocities read from STRU. In this case, if velocities are unspecified in STRU, the initial temperature is set to zero.

  If md_tfirst is set to a positive value and init_vel is true simultaneously, please make sure they are consistent, otherwise abacus will exit immediately.

  Note that md_tlast is only used in NVT/NPT simulations. If md_tlast is unset or less than zero, md_tlast is set to 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
- **Unit**: K

md_prec_level

- **Type**: Integer
- **Description**: Determine the precision level of variable-cell molecular dynamics calculations.
  - 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.
  - 2: FFT grids change per 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.
- **Default**: 0

md_restart

- **Type**: Boolean
- **Description**: Control whether to restart molecular dynamics calculations and time-dependent density functional theory calculations.
  - True: ABACUS will read in {md_step}, then read in the corresponding STRU_MD_suffix/STRU/ automatically. For tddft, ABACUS will also read in WFC_NAO_K${kpoint} of the last step (You need to set out_wfc_lcao=1 and out_app_flag=0 to obtain this file).

  - False: ABACUS will start molecular dynamics calculations normally from the first step.

- **Default**: False

md_restartfreq

- **Type**: Integer
- **Description**: The output frequency of OUT.{suffix}/STRIU/, which are used to restart molecular dynamics calculations, see md_restart in detail.
- **Default**: 5

md_dumpfreq

- **Type**: Integer
- **Description**: The output frequency of OUT.${suffix}/MD_dump in molecular dynamics calculations, which including the information of lattices and atoms.
- **Default**: 1

dump_force

- **Type**: Boolean
- **Description**: Whether to output atomic forces into the file OUT.${suffix}/MD_dump.
- **Default**: True

dump_vel

- **Type**: Boolean
- **Description**: Whether to output atomic velocities into the file OUT.${suffix}/MD_dump.
- **Default**: True

dump_virial

- **Type**: Boolean
- **Description**: Whether to output lattice virials into the file OUT.${suffix}/MD_dump.
- **Default**: True

md_seed

- **Type**: Integer
- **Description**: The random seed to initialize random numbers used in molecular dynamics calculations.
  - < 0: No srand() function is called.
  - >= 0: The function srand(md_seed) is called.
- **Default**: -1

md_tfreq

- **Type**: Real
- **Description**: Control the frequency of 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.

  Note: It is a system-dependent empirical parameter, ranging from 1/(40*md_dt) to 1/(100*md_dt). An improper choice might lead to the failure of jobs.
- **Default**: 1/40/md_dt

md_tchain

- **Type**: Integer
- **Description**: Number of thermostats coupled with the particles in the NVT/NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.
- **Default**: 1

md_pmode

- **Type**: String
- **Description**: Determine the precision level of variable-cell molecular dynamics calculations.
  - 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.
  - 2: FFT grids change per 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.
- **Default**: iso

ref_cell_factor

- **Type**: Real
- **Description**: Construct a reference cell bigger than the initial cell. 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 erf_ecut, erf_height, and erf_sigma.
- **Default**: 1.0

md_pcouple

- **Type**: String
- **Description**: The coupled lattice vectors will scale proportionally in NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.
  - 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

- **Type**: Real
- **Description**: The target pressure used in NPT ensemble simulations, 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**: -1.0
- **Unit**: kbar

md_plast

- **Type**: Real
- **Description**: The target pressure used in NPT ensemble simulations, 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**: -1.0
- **Unit**: kbar

md_pfreq

- **Type**: Real
- **Description**: The frequency of 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.

  Note: It is a system-dependent empirical parameter. An improper choice might lead to the failure of jobs.
- **Default**: 1/400/md_dt

md_pchain

- **Type**: Integer
- **Description**: The number of thermostats coupled with the barostat in the NPT ensemble based on the Nose-Hoover style non-Hamiltonian equations of motion.
- **Default**: 1

lj_rule

- **Type**: Integer
- **Description**: The Lennard-Jones potential between two atoms equals: $\sigma_k\sigma(i,j)$
- **Default**: 2

lj_eshift

- **Type**: Boolean
- **Description**: It True, the LJ potential is shifted by a constant such that it is zero at the cut-off distance.
- **Default**: False

lj_rcut

- **Type**: Real
- **Description**: Cut-off radius for Leonard Jones potential, beyond which the interaction will be neglected. It can be a single value, which means that all pairs of atoms types share the same cut-off radius. Otherwise, it should be a multiple-component vector, containing values, see details in lj_rule.
- **Default**: No default
- **Unit**: Angstrom

lj_epsilon

- **Type**: Real
- **Description**: The vector representing the matrix for Leonard Jones potential. See details in lj_rule.
- **Default**: No default
- **Unit**: eV

lj_sigma

- **Type**: Real
- **Description**: The vector representing the matrix for Leonard Jones potential. See details in lj_rule.
- **Default**: No default
- **Unit**: Angstrom

pot_file

- **Type**: String
- **Description**: The filename of DP/NEP potential files, see md.md in detail.
- **Default**: graph.pb

dp_rescaling

- **Type**: Real
- **Availability**: *esolver_type = dp.*
- **Description**: Rescaling factor to use a temperature-dependent DP. Energy, stress and force calculated by DP will be multiplied by this factor.
- **Default**: 1.0

dp_fparam

- **Type**: Real
- **Availability**: *esolver_type = dp.*
- **Description**: The frame parameter for dp potential. The array size is dim_fparam, then all frames are assumed to be provided with the same fparam.
- **Default**: {}

dp_aparam

- **Type**: Real
- **Availability**: *esolver_type = dp.*
- **Description**: The atomic parameter for dp potential. The array size can be (1) natoms x dim_aparam, then all frames are assumed to be provided with the same aparam; (2) dim_aparam, then all frames and atoms are assumed to be provided with the same aparam.
- **Default**: {}

msst_direction

- **Type**: Integer
- **Description**: The direction of the shock wave in the MSST method.
  - 0: x direction
  - 1: y direction
  - 2: z direction
- **Default**: 2

msst_vel

- **Type**: Real
- **Description**: The velocity of the shock wave in the MSST method.
- **Default**: 0.0
- **Unit**: Angstrom/fs

msst_vis

- **Type**: Real
- **Description**: Artificial viscosity in the MSST method.
- **Default**: 0.0
- **Unit**: g/(mol*Angstrom*fs)

msst_tscale

- **Type**: Real
- **Description**: The reduction percentage of the initial temperature used to compress volume in the MSST method.
- **Default**: 0.01

msst_qmass

- **Type**: Real
- **Description**: Inertia of the extended system variable. You should set a number larger than 0.
- **Default**: No default

md_damp

- **Type**: Real
- **Description**: The damping parameter used to add fictitious force in the Langevin method.
- **Default**: 1.0
- **Unit**: fs

md_tolerance

- **Type**: Real
- **Description**: The temperature tolerance for velocity rescaling. Velocities are rescaled if the current and target temperature differ more than md_tolerance.
- **Default**: 100.0
- **Unit**: K

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

cal_syns

- **Type**: Boolean *Integer*
- **Description**: Whether to calculate and output asynchronous overlap matrix for Hefei-NAMD interface. When enabled, calculates <phi(t-1)|phi(t)> by computing overlap between basis functions at atomic positions from previous time step and current time step. The overlap is calculated by shifting atom positions backward by velocity x md_dt. Output file: OUT.*/syns_nao.csr in CSR format.
  - 0 or false: disable
  - 1 or true: enable with default precision (8 digits)
  - 1 5: enable with custom precision (5 digits)

    Note: Only works with LCAO basis and molecular dynamics calculations. Requires atomic velocities. Output starts from the second MD step (istep > 0).
- **Default**: False

dmax

- **Type**: Real
- **Description**: The maximum displacement of all atoms in one step. This parameter is useful when cal_syns = True.
- **Default**: 0.01
- **Unit**: bohr

back to top

15.1.22 DFT+U correction

dft_plus_u

- **Type**: Integer
- **Description**: Determines whether to calculate the plus U correction, which is especially important for correlated electrons.
  - 1: Calculate plus U correction with radius-adjustable localized projections (with parameter onsite_radius).
  - 2: Calculate plus U correction using first zeta of NAOs as projections (this is old method for testing).
  - 0: Do not calculate plus U correction.
- **Default**: 0

dft_plus_dmft

- **Type**: Boolean
- **Availability**: *basis_type==lcao*
- **Description**: Whether to enable DFT+DMFT calculation. True: DFT+DMFT; False: standard DFT calculation.
- **Default**: False

orbital_corr

- **Type**: Vector of Integer (n values where n is the number of atomic types)
- **Description**: Specifies which orbits need plus U correction for each atom type (for atom type 1, 2, 3, respectively).
  - -1: The plus U correction will not be calculated for this atom.
  - 1: For p-electron orbits, the plus U correction is needed.
  - 2: For d-electron orbits, the plus U correction is needed.
  - 3: For f-electron orbits, the plus U correction is needed.
- **Default**: -1

hubbard_u

- **Type**: Vector of Real (n values where n is the number of atomic types)
- **Description**: Specifies the Hubbard Coulomb interaction parameter U (eV) in plus U correction, which should be specified for each atom unless the Yukawa potential is used.

  Note: Since only the simplified scheme by Duradev is implemented, the 'U' here is actually U-effective, which is given by Hubbard U minus Hund J.
- **Default**: 0.0

yukawa_potential

- **Type**: Boolean
- **Description**: Determines whether to use the local screen Coulomb potential method to calculate the values of U and J.
  - True: hubbard_u does not need to be specified.
  - False: hubbard_u does need to be specified.

- **Default**: False

yukawa_lambda

- **Type**: Real
- **Availability**: *DFT+U with yukawa_potential = True.*
- **Description**: The screen length of Yukawa potential. If left to default, the screen length will be calculated 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.

uramping

- **Type**: Real
- **Availability**: *DFT+U calculations with mixing_restart > 0.*
- **Description**: Once uramping > 0.15 eV. DFT+U calculations will start SCF with U = 0 eV, namely normal LDA/PBE calculations. Once SCF restarts when drho<mixing_restart, U value will increase by uramping eV. SCF will repeat above calcuations until U values reach target defined in hubbard_u. As for uramping=1.0 eV, the recommendations of mixing_restart is around 5e-4.
- **Default**: -1.0.
- **Unit**: eV

omc

- **Type**: Integer
- **Description**: The parameter controls the form of occupation matrix control used.
  - 0: No occupation matrix control is performed, and the onsite density matrix will be calculated from wavefunctions in each SCF step.
  - 1: 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.
  - 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

onsite_radius

- **Type**: Real
- **Availability**: *dft_plus_u is set to 1*
- **Description**: - The onsite_radius parameter facilitates modulation of the single-zeta portion of numerical atomic orbitals used for DFT+U projections.
  - The modulation algorithm applies a smooth truncation to the orbital tail followed by normalization. A representative profile is $f(r) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{r_c - r}{\sigma}\right)\right]$, where r_c is the cutoff radius and $\sigma = \gamma r_c$ controls smoothness.
- **Default**: 3.0
- **Unit**: Bohr

back to top

15.1.23 Spin-Constrained DFT

sc_mag_switch

- **Type**: Boolean
- **Description**: Switch to control spin-constrained DFT calculation
- **Default**: False

decay_grad_switch

- **Type**: Boolean
- **Description**: Switch to control gradient break condition in spin-constrained DFT
- **Default**: False

sc_thr

- **Type**: Real
- **Availability**: *sc_mag_switch is true*
- **Description**: Convergence criterion of spin-constrained iteration (RMS) in uB
- **Default**: 1.0e-6
- **Unit**: uB

nsc

- **Type**: Integer
- **Availability**: *sc_mag_switch is true*
- **Description**: Maximal number of spin-constrained iteration
- **Default**: 100

nsc_min

- **Type**: Integer
- **Availability**: *sc_mag_switch is true*
- **Description**: Minimum number of spin-constrained iteration
- **Default**: 2

sc_scf_nmin

- **Type**: Integer
- **Availability**: *sc_mag_switch is true*
- **Description**: Minimum number of outer scf loop before initializing lambda loop
- **Default**: 2

alpha_trial

- **Type**: Real
- **Availability**: *sc_mag_switch is true*
- **Description**: Initial trial step size for lambda in eV/uB^2
- **Default**: 0.01
- **Unit**: eV/uB^2

sccut

- **Type**: Real
- **Availability**: *sc_mag_switch is true*
- **Description**: Maximal step size for lambda in eV/uB
- **Default**: 3.0
- **Unit**: eV/uB

sc_drop_thr

- **Type**: Real
- **Availability**: *sc_mag_switch is true*
- **Description**: Convergence criterion ratio of lambda iteration in Spin-constrained DFT
- **Default**: 1.0e-2

sc_scf_thr

- **Type**: Real
- **Availability**: *sc_mag_switch is true*
- **Description**: Density error threshold for inner loop of spin-constrained SCF
- **Default**: 1.0e-4

back to top

15.1.24 vdW correction

vdw_method

- **Type**: String
- **Description**: Specifies the method used for Van der Waals (VdW) correction. Available options are:
  - d2: Grimme's D2 dispersion correction method
  - d3_0: Grimme's DFT-D3(0) dispersion correction method (zero-damping)
  - d3_bj: Grimme's DFTD3(BJ) dispersion correction method (BJ-damping)
  - d4: Grimme's DFT-D4 dispersion correction method using the external DFT-D4 library
  - none: no vdW correction

Note: ABACUS supports automatic setting of DFT-D3 parameters for common functionals. To benefit from this feature, please specify the parameter dft_functional explicitly, otherwise the autoset procedure will crash. If not satisfied with the built-in parameters, any manual setting on vdw_s6, vdw_s8, vdw_a1 and vdw_a2 will overwrite the automatic values.

Note: DFT-D4 support requires ABACUS to be configured with ENABLE_DFTD4=ON and a CMake-installed dftd4 library exporting `dftd4-config.cmake`. DFT-D4 damping parameters are loaded from the external library.

- **Default**: none

vdw_d4_xc

- **Type**: String
- **Availability**: *vdw_method is set to d4*
- **Description**: Functional name passed to the DFT-D4 library to load its internal damping parameters. If set to default, ABACUS infers the functional name from dft_functional or pseudopotential metadata.
- **Default**: default

vdw_d4_model

- **Type**: String
- **Availability**: *vdw_method is set to d4*
- **Description**: DFT-D4 dispersion model used by the external DFT-D4 library. Available options are d4 for the standard D4 model and d4s for the smooth D4S model.
- **Default**: d4

vdw_s6

- **Type**: String
- **Availability**: *vdw_method is set to d2, d3_0, or d3_bj*
- **Description**: This scale factor is used to optimize the interaction energy deviations in van der Waals (vdW) corrected calculations. The recommended values of this parameter are dependent on the chosen vdW correction method and the DFT functional being used. For DFT-D2, the recommended values are 0.75 (PBE), 1.2 (BLYP), 1.05 (B-P86), 1.0 (TPSS), and 1.05 (B3LYP). If not set, will use values of PBE functional. For DFT-D3, recommended values with different DFT functionals can be found on the here. If not set, will search in ABACUS built-in dataset based on the dft_functional keywords. User set value will overwrite the searched value.

vdw_s8

- **Type**: String
- **Availability**: *vdw_method is set to d3_0 or d3_bj*
- **Description**: This scale factor is relevant for D3(0) and D3(BJ) van der Waals (vdW) correction methods. The recommended values of this parameter with different DFT functionals can be found on the webpage. If not set, will search in ABACUS built-in dataset based on the dft_functional keywords. User set value will overwrite the searched value.

vdw_a1

- **Type**: String
- **Availability**: *vdw_method is set to d3_0 or d3_bj*
- **Description**: This damping function parameter is relevant for D3(0) and D3(BJ) van der Waals (vdW) correction methods. The recommended values of this parameter with different DFT functionals can be found on the webpage. If not set, will search in ABACUS built-in dataset based on the dft_functional keywords. User set value will overwrite the searched value.

vdw_a2

- **Type**: String
- **Availability**: *vdw_method is set to d3_0 or d3_bj*
- **Description**: This damping function parameter is only relevant for D3(0) and D3(BJ) van der Waals (vdW) correction methods. The recommended values of this parameter with different DFT functionals can be found on the webpage. If not set, will search in ABACUS built-in dataset based on the dft_functional keywords. User set value will overwrite the searched value.

vdw_d

- **Type**: Real
- **Availability**: *vdw_method is set to d2*
- **Description**: Controls the damping rate of the damping function in the DFT-D2 method.
- **Default**: 20

vdw_abc

- **Type**: Boolean
- **Availability**: *vdw_method is set to d3_0 or d3_bj*
- **Description**: Determines whether three-body terms are calculated for DFT-D3 methods.
  - True: ABACUS will calculate the three-body term.
  - False: The three-body term is not included.
- **Default**: False

vdw_c6_file

- **Type**: String
- **Availability**: *vdw_method is set to d2*
- **Description**: Specifies the name of the file containing parameters for each element when using the D2 method. If not set, ABACUS uses the default parameters (Jnm6/mol) stored in the program. To manually set the parameters, provide a file containing the parameters. An example is given by:

  H 0.1 Si 9.0

  Namely, each line contains the element name and the corresponding parameter.
- **Default**: default

vdw_c6_unit

- **Type**: String
- **Availability**: *vdw_C6_file is not default*
- **Description**: Specifies the unit of the provided parameters in the D2 method. Available options are:
  - Jnm6/mol (J nm^6/mol)
  - eVA (eV Angstrom)
- **Default**: Jnm6/mol

vdw_r0_file

- **Type**: String
- **Availability**: *vdw_method is set to d2*
- **Description**: Specifies the name of the file containing parameters for each element when using the D2 method. If not set, ABACUS uses the default parameters (Angstrom) stored in the program. To manually set the parameters, provide a file containing the parameters. An example is given by:

  Li 1.0 Cl 2.0

  Namely, each line contains the element name and the corresponding parameter.
- **Default**: default

vdw_r0_unit

- **Type**: String
- **Availability**: *vdw_R0_file is not default*
- **Description**: Specifies the unit for the parameters in the D2 method when manually set by the user. Available options are:
  - A (Angstrom)
  - Bohr
- **Default**: A

vdw_cutoff_type

- **Type**: String
- **Description**: Determines the method used for specifying the cutoff radius in periodic systems when applying Van der Waals correction. Available options are:
  - radius: The supercell is selected within a sphere centered at the origin with a radius defined by vdw_cutoff_radius.
  - period: The extent of the supercell is explicitly specified using the vdw_cutoff_period keyword.
- **Default**: radius

vdw_cutoff_radius

- **Type**: String
- **Availability**: *vdw_cutoff_type is set to radius*

- **Description**: Defines the cutoff radius when vdw_cutoff_type is set to radius. The default values depend on the chosen vdw_method. For DFT-D4, this controls the two-body dispersion cutoff, while the three-body cutoff is internally limited to the DFT-D4 default value of 40 Bohr.
- **Unit**: defined by vdw_radius_unit (default Bohr)

vdw_radius_unit

- **Type**: String
- **Availability**: *vdw_cutoff_type is set to radius*
- **Description**: Specify the unit of vdw_cutoff_radius. Available options are:
  - A(Angstrom)
  - Bohr
- **Default**: Bohr

vdw_cutoff_period

- **Type**: Integer Integer Integer
- **Availability**: *vdw_cutoff_type is set to period*
- **Description**: The three integers supplied here 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
- **Availability**: *vdw_method is set to d3_0, d3_bj, or d4*
- **Description**: The cutoff radius when calculating coordination numbers. The default is 40 Bohr for DFT-D3 and 30 Bohr for DFT-D4.
- **Default**: 40
- **Unit**: defined by vdw_cn_thr_unit (default: Bohr)

vdw_cn_thr_unit

- **Type**: String
- **Description**: Unit of the coordination number cutoff (vdw_cn_thr). Available options are:
  - A(Angstrom)
  - Bohr
- **Default**: Bohr

back to top

15.1.25 Berry phase and wannier90 interface

berry_phase

- **Type**: Boolean
- **Description**: Controls the calculation of Berry phase

  - true: Calculate Berry phase.
  - false: Do not calculate Berry phase.

- **Default**: false

gdir

- **Type**: Integer
- **Description**: The direction of the polarization in the lattice vector for Berry phase calculation
  - 1: Calculate the polarization in the direction of the lattice vector a_1 defined in the STRU file.
  - 2: Calculate the polarization in the direction of the lattice vector a_2 defined in the STRU file.
  - 3: Calculate the polarization in the direction of the lattice vector a_3 defined in the STRU file.
- **Default**: 3

towannier90

- **Type**: Boolean
- **Description**: Controls the generation of files for the Wannier90 code.
  - 1: Generate files for the Wannier90 code.
  - 0: Do not generate files for the Wannier90 code.
- **Default**: 0

nnkpfile

- **Type**: String
- **Description**: The file name generated when running "wannier90 -pp …" command
- **Default**: seedname.nnkp

wannier_method

- **Type**: Integer
- **Description**: Only available on LCAO basis, using different methods to generate ".mmn" file and ".amn" file.
  - 1: Calculated using the lcao_in_pw method, the calculation accuracy can be improved by increasing ecutwfc to maintain consistency with the pw basis set results.
  - 2: The overlap between atomic orbitals is calculated using grid integration. The radial grid points are generated using the Gauss-Legendre method, while the spherical grid points are generated using the Lebedev-Laikov method.
- **Default**: 1

wannier_spin

- **Type**: String
- **Description**: The spin direction for the Wannier function calculation when nspin is set to 2
  - up: Calculate spin up for the Wannier function.
  - down: Calculate spin down for the Wannier function.
- **Default**: up

out_wannier_mmn

- **Type**: Boolean
- **Description**: Write the "*.mmn" file or not.
  - 0: don't write the "*.mmn" file.
  - 1: write the "*.mmn" file.
- **Default**: 1

out_wannier_amn

- **Type**: Boolean
- **Description**: Write the "*.amn" file or not.
  - 0: don't write the "*.amn" file.
  - 1: write the "*.amn" file.
- **Default**: 1

out_wannier_eig

- **Type**: Boolean
- **Description**: Write the "*.eig" file or not.
  - 0: don't write the "*.eig" file.
  - 1: write the "*.eig" file.
- **Default**: 1

out_wannier_unk

- **Type**: Boolean
- **Description**: Write the "UNK.*" file or not.
  - 0: don't write the "UNK.*" file.
  - 1: write the "UNK.*" file.
- **Default**: 0

out_wannier_wvfn_formatted

- **Type**: Boolean
- **Description**: Write the "UNK.*" file in ASCII format or binary format.
  - 0: write the "UNK.*" file in binary format.
  - 1: write the "UNK.*" file in ASCII format (text file format).
- **Default**: 1

back to top

15.1.26 RT-TDDFT: Real-Time Time-Dependent Density Functional Theory

estep_per_md

- **Type**: Integer
- **Description**: The number of electronic propagation steps between two ionic steps.
- **Default**: 1

td_dt

- **Type**: Real
- **Description**: The time step used in electronic propagation. Setting td_dt will reset the value of md_dt to td_dt * estep_per_md.
- **Default**: md_dt / estep_per_md
- **Unit**: fs

td_edm

- **Type**: Integer
- **Description**: Method to calculate the energy-density matrix, mainly affects the calculation of force and stress.
  - 0: Using the original formula.
  - 1: Using the formula for ground state (deprecated). Note that this usually does not hold if wave function is not the eigenstate of the Hamiltonian.
- **Default**: 0

td_print_eij

- **Type**: Real
- **Description**: Controls the printing of Hamiltonian matrix elements.
  - < 0: Suppress all output.
  - >= 0: Print only elements with either i or j exceeding td_print_eij.
- **Default**: -1
- **Unit**: Ry

td_propagator

- **Type**: Integer
- **Description**: Methods of electronic propagation.
  - 0: Crank-Nicolson, based on matrix inversion.
  - 1: 4th-order Taylor expansion of exponential.
  - 2: Enforced time-reversal symmetry (ETRS).
  - 3: Crank-Nicolson, based on solving linear equation.
- **Default**: 0

td_vext

- **Type**: Boolean
- **Description**: - True: Add a laser-material interaction (external electric field).
  - False: No external electric field.
- **Default**: False

td_vext_dire

- **Type**: String
- **Description**: Specifies the direction(s) of the external electric field when td_vext is enabled. For example, td_vext_dire 1 2 indicates that external electric fields are applied to both the x and y directions simultaneously. Electric field parameters can also be written as strings. For example, td_gauss_phase 0 1.5707963 indicates that the Gaussian type electric fields in the x and y directions have a phase delay of pi/2.
  - 1: The external field direction is along the x-axis.
  - 2: The external field direction is along the y-axis.
  - 3: The external field direction is along the z-axis.
- **Default**: 1

td_stype

- **Type**: Integer
- **Description**: Type of electric field in the space domain, i.e. the gauge of the electric field.
  - 0: Length gauge.
  - 1: Velocity gauge.
  - 2: Hybrid gauge. See J. Chem. Theory Comput. 2025, 21, 3335-3341 for more information.
- **Default**: 0

td_ttype

- **Type**: String
- **Description**: Type of electric field in the time domain.
  - 0: Gaussian type function.
  - 1: Trapezoid type function.
  - 2: Trigonometric type function.
  - 3: Heaviside type function.
- **Default**: 0

td_tstart

- **Type**: Integer
- **Description**: The initial time step when the time-dependent electric field is activated.
- **Default**: 1

td_tend

- **Type**: Integer
- **Description**: The final time step when the time-dependent electric field is deactivated. The field remains active between td_tstart and td_tend.
- **Default**: 1000

td_lcut1

- **Type**: Real
- **Description**: The lower bound of the interval in the length gauge RT-TDDFT, where the coordinate is the fractional coordinate.
- **Default**: 0.05

td_lcut2

- **Type**: Real
- **Description**: The upper bound of the interval in the length gauge RT-TDDFT, where the coordinate is the fractional coordinate.
- **Default**: 0.95

td_gauss_freq

- **Type**: String
- **Description**: Frequency of the Gaussian type electric field.
- **Default**: 22.13
- **Unit**: 1/fs

td_gauss_phase

- **Type**: String
- **Description**: Phase of the Gaussian type electric field.
- **Default**: 0.0

td_gauss_sigma

- **Type**: String
- **Description**: Pulse width (standard deviation) of the Gaussian type electric field.
- **Default**: 30.0
- **Unit**: fs

td_gauss_t0

- **Type**: String
- **Description**: Step number of the time center of the Gaussian type electric field.
- **Default**: 100

td_gauss_amp

- **Type**: String
- **Description**: Amplitude of the Gaussian type electric field.
- **Default**: 0.25
- **Unit**: V/Angstrom

td_trape_freq

- **Type**: String
- **Description**: Frequency of the trapezoid type electric field.
- **Default**: 1.60
- **Unit**: 1/fs

td_trape_phase

- **Type**: String
- **Description**: Phase of the trapezoid type electric field.
- **Default**: 0.0

td_trape_t1

- **Type**: String
- **Description**: Step number of the time interval t1 of the trapezoid type electric field.
- **Default**: 1875

td_trape_t2

- **Type**: String
- **Description**: Step number of the time interval t2 of the trapezoid type electric field.
- **Default**: 5625

td_trape_t3

- **Type**: String
- **Description**: Step number of the time interval t3 of the trapezoid type electric field.
- **Default**: 7500

td_trape_amp

- **Type**: String
- **Description**: Amplitude of the trapezoid type electric field.
- **Default**: 2.74
- **Unit**: V/Angstrom

td_trigo_freq1

- **Type**: String
- **Description**: Frequency 1 of the trigonometric type electric field.
- **Default**: 1.164656
- **Unit**: 1/fs

td_trigo_freq2

- **Type**: String
- **Description**: Frequency 2 of the trigonometric type electric field.
- **Default**: 0.029116
- **Unit**: 1/fs

td_trigo_phase1

- **Type**: String
- **Description**: Phase 1 of the trigonometric type electric field.
- **Default**: 0.0

td_trigo_phase2

- **Type**: String
- **Description**: Phase 2 of the trigonometric type electric field.
- **Default**: 0.0

td_trigo_amp

- **Type**: String
- **Description**: Amplitude of the trigonometric type electric field.
- **Default**: 2.74
- **Unit**: V/Angstrom

td_heavi_t0

- **Type**: String
- **Description**: Step number of the switch time of the Heaviside type electric field.
- **Default**: 100

td_heavi_amp

- **Type**: String
- **Description**: Amplitude of the Heaviside type electric field.
- **Default**: 1.0
- **Unit**: V/Angstrom

init_vecpot_file

- **Type**: Boolean
- **Description**: Initialize vector potential through file or not.
  - True: Initialize vector potential from file At.dat (unit: a.u.). It consists of four columns, representing the step number and vector potential on each direction.
  - False: Calculate vector potential by integrating the electric field.
- **Default**: False

ocp

- **Type**: Boolean
- **Description**: - True: Fixes the band occupations based on the values specified in ocp_set.
  - False: Does not fix the band occupations.
- **Default**: False

ocp_set

- **Type**: String
- **Description**: If ocp is set to 1, ocp_set must be provided as a string specifying the occupation numbers for each band across all k-points. The format follows a space-separated pattern, where occupations are assigned sequentially to bands for each k-point. A shorthand notation Nx can be used to repeat a value x for N bands.
  - Example: 1 10*1 0 1 represents occupations for 13 bands, where the 12th band is fully unoccupied (0), and all others are occupied (1).
  - For a system with multiple k-points, the occupations must be specified for all k-points, following their order in the output file kpoints (may lead to fractional occupations).
  - Incorrect specification of ocp_set could lead to inconsistencies in electron counting, causing the calculation to terminate with an error.
- **Default**: None

out_dipole

- **Type**: Boolean
- **Description**: - True: Output electric dipole moment.
  - False: Do not output electric dipole moment.
- **Default**: False

out_current

- **Type**: Integer
- **Description**: - 0: Do not output current.
  - 1: Output current using the two-center integral, faster.
  - 2: Output current using the matrix commutation, more precise.
- **Default**: 0

out_current_k

- **Type**: Boolean
- **Description**: - True: Output current for each k-points separately.
  - False: Output current in total.
- **Default**: False

out_efield

- **Type**: Boolean
- **Description**: Whether to output the electric field data to files. When enabled, writes real-time electric field values (unit: V/A) into files named efield_[num].txt, where [num] is the sequential index of the electric field ranges from 0 to N-1 for N configured fields. It is noteworthy that the field type sequence follows td_ttype, while the direction sequence follows td_vext_dire.
  - True: Output electric field.
  - False: Do not output electric field.
- **Default**: False

out_vecpot

- **Type**: Boolean
- **Description**: Output vector potential or not (unit: a.u.).
  - True: Output vector potential into file At.dat.
  - False: Do not output vector potential.
- **Default**: False

back to top

15.1.27 Variables useful for debugging

nurse

- **Type**: Integer
- **Description**: Debugging flag for developers
- **Default**: 0

t_in_h

- **Type**: Boolean
- **Description**: Specify whether to include kinetic term in obtaining the Hamiltonian matrix.
  - 0: No.
  - 1: Yes.
- **Default**: 1

vl_in_h

- **Type**: Boolean
- **Description**: Specify whether to include local pseudopotential term in obtaining the Hamiltonian matrix.
  - 0: No.
  - 1: Yes.
- **Default**: 1

vnl_in_h

- **Type**: Boolean
- **Description**: Specify whether to include non-local pseudopotential term in obtaining the Hamiltonian matrix.
  - 0: No.
  - 1: Yes.
- **Default**: 1

vh_in_h

- **Type**: Boolean
- **Description**: Specify whether to include Hartree potential term in obtaining the Hamiltonian matrix.
  - 0: No.
  - 1: Yes.
- **Default**: 1

vion_in_h

- **Type**: Boolean
- **Description**: Specify whether to include local ionic potential term in obtaining the Hamiltonian matrix.
  - 0: No.
  - 1: Yes.
- **Default**: 1

test_force

- **Type**: Boolean
- **Description**: Specify whether to output the detailed components in forces.
  - 0: No.
  - 1: Yes.
- **Default**: 0

test_stress

- **Type**: Boolean
- **Description**: Specify whether to output the detailed components in stress.
  - 0: No.
  - 1: Yes.
- **Default**: 0

test_skip_ewald

- **Type**: Boolean
- **Description**: Specify whether to skip the calculation of the ewald energy.
  - 0: No.
  - 1: Yes.
- **Default**: 0

back to top

15.1.28 Electronic conductivities

cal_cond

- **Type**: Boolean
- **Availability**: *basis_type = pw*
- **Description**: Whether to calculate electronic conductivities.
- **Default**: False

cond_che_thr

- **Type**: Real
- **Availability**: *esolver_type = sdft*
- **Description**: Control the error of Chebyshev expansions for conductivities.
- **Default**: 1e-8

cond_dw

- **Type**: Real
- **Availability**: *basis_type = pw*
- **Description**: Frequency interval () for frequency-dependent conductivities.
- **Default**: 0.1
- **Unit**: eV

cond_wcut

- **Type**: Real
- **Availability**: *basis_type = pw*
- **Description**: Cutoff frequency for frequency-dependent conductivities.
- **Default**: 10.0
- **Unit**: eV

cond_dt

- **Type**: Real
- **Availability**: *basis_type = pw*
- **Description**: Time interval () to integrate Onsager coefficients.
- **Default**: 0.02
- **Unit**: a.u.

cond_dtbatch

- **Type**: Integer
- **Availability**: *esolver_type = sdft*
- **Description**: exp(iH\dt\cond_dtbatch) is expanded with Chebyshev expansion to calculate conductivities. It is faster but costs more memory.
  - If cond_dtbatch = 0: Autoset this parameter to make expansion orders larger than 100.
- **Default**: 0

cond_smear

- **Type**: Integer
- **Description**: Smearing method for conductivities
  - 1: Gaussian smearing
  - 2: Lorentzian smearing
- **Default**: 1

cond_fwhm

- **Type**: Real
- **Availability**: *basis_type = pw*
- **Description**: FWHM for conductivities. For Gaussian smearing, ; for Lorentzian smearing, .
- **Default**: 0.4
- **Unit**: eV

cond_nonlocal

- **Type**: Boolean
- **Availability**: *basis_type = pw*
- **Description**: Whether to consider nonlocal potential correction when calculating velocity matrix .
  - True: .
  - False: .
- **Default**: True

back to top

15.1.29 Implicit solvation model

imp_sol

- **Type**: Boolean
- **Description**: Calculate implicit solvation correction
- **Default**: False

eb_k

- **Type**: Real
- **Availability**: *imp_sol is true.*
- **Description**: The relative permittivity of the bulk solvent, 80 for water
- **Default**: 80

tau

- **Type**: Real
- **Description**: The effective surface tension parameter that describes the cavitation, the dispersion, and the repulsion interaction between the solute and the solvent which are not captured by the electrostatic terms
- **Default**: 1.0798e-05

sigma_k

- **Type**: Real
- **Description**: The width of the diffuse cavity that is implicitly determined by the electronic structure of the solute
- **Default**: 0.6

nc_k

- **Type**: Real
- **Description**: The value of the electron density at which the dielectric cavity forms
- **Default**: 0.00037

back to top

15.1.30 Quasiatomic Orbital (QO) analysis

qo_switch

- **Type**: Boolean
- **Description**: Whether to let ABACUS output QO analysis required files
- **Default**: False

qo_basis

- **Type**: String
- **Description**: Type of QO basis function:
  - hydrogen: hydrogen-like basis
  - pswfc: read basis from pseudopotential
  - szv: single-zeta valence basis
- **Default**: szv

qo_strategy

- **Type**: Vector of String (1 or n values where n is the number of atomic types)
- **Description**: Strategy to generate radial orbitals for QO analysis. For hydrogen: energy-valence, for pswfc and szv: all
- **Default**: for hydrogen: energy-valence, for pswfc and szv: all

qo_screening_coeff

- **Type**: Vector of Real (n values where n is the number of atomic types; 1 value allowed for qo_basis=pswfc)
- **Description**: The screening coefficient for each atom type to rescale the shape of radial orbitals
- **Default**: 0.1
- **Unit**: Bohr^-1

qo_thr

- **Type**: Real
- **Description**: The convergence threshold determining the cutoff of generated orbital. Lower threshold will yield orbital with larger cutoff radius.
- **Default**: 1.0e-6

back to top

15.1.31 PEXSI

pexsi_npole

- **Type**: Integer
- **Description**: The number of poles used in the pole expansion method, should be a even number.
- **Default**: 40

pexsi_inertia

- **Type**: Boolean
- **Description**: Whether inertia counting is used at the very beginning.
- **Default**: True

pexsi_nmax

- **Type**: Integer
- **Description**: Maximum number of PEXSI iterations after each inertia counting procedure.
- **Default**: 80

pexsi_comm

- **Type**: Boolean
- **Description**: Whether to construct PSelInv communication pattern.
- **Default**: True

pexsi_storage

- **Type**: Boolean
- **Description**: Whether to use symmetric storage space used by the Selected Inversion algorithm for symmetric matrices.
- **Default**: True

pexsi_ordering

- **Type**: Integer
- **Description**: Ordering strategy for factorization and selected inversion. 0: Parallel ordering using ParMETIS, 1: Sequential ordering using METIS, 2: Multiple minimum degree ordering
- **Default**: 0

pexsi_row_ordering

- **Type**: Integer
- **Description**: Row permutation strategy for factorization and selected inversion, 0: No row permutation, 1: Make the diagonal entry of the matrix larger than the off-diagonal entries.
- **Default**: 1

pexsi_nproc

- **Type**: Integer
- **Description**: Number of processors for PARMETIS. Only used if pexsi_ordering == 0.
- **Default**: 1

pexsi_symm

- **Type**: Boolean
- **Description**: Whether the matrix is symmetric.
- **Default**: True

pexsi_trans

- **Type**: Boolean
- **Description**: Whether to factorize the transpose of the matrix.
- **Default**: False

pexsi_method

- **Type**: Integer
- **Description**: The pole expansion method to be used. 1 for Cauchy Contour Integral method, 2 for Moussa optimized method.
- **Default**: 1

pexsi_nproc_pole

- **Type**: Integer
- **Description**: The point parallelizaion of PEXSI. Recommend two points parallelization.
- **Default**: 1

pexsi_temp

- **Type**: Real
- **Description**: Temperature in Fermi-Dirac distribution, in Ry, should have the same effect as the smearing sigma when smearing method is set to Fermi-Dirac.
- **Default**: 0.015

pexsi_gap

- **Type**: Real
- **Description**: Spectral gap, this can be set to be 0 in most cases.
- **Default**: 0

pexsi_delta_e

- **Type**: Real
- **Description**: Upper bound for the spectral radius of S^{-1}H.
- **Default**: 20

pexsi_mu_lower

- **Type**: Real
- **Description**: Initial guess of lower bound for mu.
- **Default**: -10

pexsi_mu_upper

- **Type**: Real
- **Description**: Initial guess of upper bound for mu.
- **Default**: 10

pexsi_mu

- **Type**: Real
- **Description**: Initial guess for mu (for the solver).
- **Default**: 0

pexsi_mu_thr

- **Type**: Real
- **Description**: Stopping criterion in terms of the chemical potential for the inertia counting procedure.
- **Default**: 0.05

pexsi_mu_expand

- **Type**: Real
- **Description**: If the chemical potential is not in the initial interval, the interval is expanded by this value.
- **Default**: 0.3

pexsi_mu_guard

- **Type**: Real
- **Description**: Safe guard criterion in terms of the chemical potential to reinvoke the inertia counting procedure.
- **Default**: 0.2

pexsi_elec_thr

- **Type**: Real
- **Description**: Stopping criterion of the PEXSI iteration in terms of the number of electrons compared to numElectronExact.
- **Default**: 0.001

pexsi_zero_thr

- **Type**: Real
- **Description**: if the absolute value of CCS matrix element is less than this value, it will be considered as zero.
- **Default**: 1e-10

back to top

15.1.32 Linear Response TDDFT

ri_hartree_benchmark

- **Type**: String
- **Description**: Whether to use the RI approximation for the Hartree term in LR-TDDFT for benchmark (with FHI-aims/ABACUS read-in style)
- **Default**: none

aims_nbasis

- **Type**: A number(ntype) of Integers
- **Availability**: *ri_hartree_benchmark = aims*
- **Description**: Atomic basis set size for each atom type (with the same order as in STRU) in FHI-aims.
- **Default**: {} (empty list, where ABACUS use its own basis set size)

back to top

15.1.33 Linear Response TDDFT (Under Development Feature)

xc_kernel

- **Type**: String
- **Description**: The exchange-correlation kernel used in the calculation. Currently supported: RPA, LDA, PBE, HSE, HF.
- **Default**: LDA

lr_init_xc_kernel

- **Type**: Vector of String (>=1 values)
- **Description**: The method to initalize the xc kernel.
  - "default": Calculate xc kernel from the ground-state charge density.
  - "file": Read the xc kernel on grid from the provided files. The following words should be the paths of ".cube" files, where the first 1 (nspin==1) or 3 (nspin==2, namely spin-aa, spin-ab and spin-bb) will be read in. The parameter xc_kernel will be invalid. Now only LDA-type kernel is supported as the potential will be calculated by directly multiplying the transition density.
  - "from_charge_file": Calculate fxc from the charge density read from the provided files. The following words should be the paths of ".cube" files, where the first nspin files will be read in.
- **Default**: "default"

lr_solver

- **Type**: String
- **Description**: The method to solve the Casida equation in LR-TDDFT under Tamm-Dancoff approximation (TDA).
  - dav/dav_subspace/cg: Construct and diagonalize the Hamiltonian matrix iteratively with Davidson/Non-ortho-Davidson/CG algorithm.

  - lapack: Construct the full matrix and directly diagonalize with LAPACK.
  - spectrum: Calculate absorption spectrum only without solving Casida equation.
- **Default**: dav

lr_thr

- **Type**: Real
- **Description**: The convergence threshold of iterative diagonalization solver for LR-TDDFT. It is a pure-math number with the same meaning as pw_diag_thr, but since the Casida equation is a one-shot eigenvalue problem, it is also the convergence threshold of LR-TDDFT.
- **Default**: 1e-2

nocc

- **Type**: Integer
- **Description**: The number of occupied orbitals (up to HOMO) used in the LR-TDDFT calculation.
  - Note: If the value is illegal (> nelec/2 or <= 0), it will be autoset to nelec/2.
- **Default**: nband

nvirt

- **Type**: Integer
- **Description**: The number of virtual orbitals (starting from LUMO) used in the LR-TDDFT calculation.
- **Default**: 1

lr_nstates

- **Type**: Integer
- **Description**: The number of 2-particle states to be solved.
- **Default**: 0

lr_unrestricted

- **Type**: Boolean
- **Description**: Whether to use unrestricted construction for LR-TDDFT (the matrix size will be doubled).
  - True: Always use unrestricted LR-TDDFT.
  - False: Use unrestricted LR-TDDFT only when the system is open-shell.
- **Default**: False

abs_wavelen_range

- **Type**: Real Real
- **Description**: The range of the wavelength for the absorption spectrum calculation.
- **Default**: 0.0 0.0
- **Unit**: nm

out_wfc_lr

- **Type**: Boolean
- **Description**: Whether to output the eigenstates (excitation energy) and eigenvectors (excitation amplitude) of the LR-TDDFT calculation. The output files are OUT.{suffix}/Excitation_Amplitude_${processor_rank}.dat.
- **Default**: False

abs_gauge

- **Type**: String
- **Description**: Whether to use length or velocity gauge to calculate the absorption spectrum in LR-TDDFT.
- **Default**: length

abs_broadening

- **Type**: Real
- **Description**: The broadening factor for the absorption spectrum calculation.
- **Default**: 0.01

back to top

15.1.34 Reduced Density Matrix Functional Theory

rdmft

- **Type**: Boolean
- **Description**: Whether to perform rdmft calculation (reduced density matrix funcional theory)
- **Default**: false

rdmft_power_alpha

- **Type**: Real
- **Description**: The alpha parameter of power-functional(or other exx-type/hybrid functionals) which used in RDMFT, g(occ_number) = occ_number^alpha
- **Default**: 0.656

back to top

15.2 The STRU file

- *LATTICE_VECTORS*
- *LATTICE_PARAMETERS*
- *ATOMIC_POSITIONS*
- *More Key Words*

15.2.1 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; pseudo_type

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
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)
```

(continues on next page)

(continued from previous page)

```
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.

15.2.2 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.

When *esolver_type* is set to `lj` or `dp`, the keyword `pseudo_file` and `pseudo_type` is needless.

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.
5. For additional pseudopotential options and to view the basic benchmark test results of these pseudopotentials in ABACUS, please refer to the Benchmarks website

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. Recommendation for Pseudopotential and Orbital Sets For general usage requirements, the APNSv1.0 pseudopotential and orbital set is recommended. You can access it via AIS square 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.

More Key Words

Several other parameters could be defined after the atom position using key words :

- `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, used only for restarting MD 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 overrided.
  - `angle1`: in non-colinear case, specify the angle between z-axis and real spin, in angle measure instead of radian measure
  - `angle2`: in non-colinear case, specify angle between x-axis and real spin in projection in xy-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
```

  - Default: If users do not specalize a finite magnetic moment for all atoms in a magnetic calculations (`nspin==2 || nspin == 4`), e.g.:

```
Fe
0.0
2
0.0 0.0 0.0 m 0 0 0
0.5 0.5 0.5 m 1 1 1

O
0.0
2
0.0 0.0 0.0 m 0 0 0
0.5 0.5 0.5 m 1 1 1
```

    For `nspin==2`, we will autoset atomic magmon is `1.0`:

```
Fe
1.0
2
0.0 0.0 0.0 m 0 0 0
0.5 0.5 0.5 m 1 1 1

Fe
1.0
```

(continues on next page)

(continued from previous page)

```
2
0.0 0.0 0.0 m 0 0 0
0.5 0.5 0.5 m 1 1 1
```

For `nspin==4`, we will autoset atomic magmon as follow:

```
Fe
0.0
2
0.0 0.0 0.0 m 0 0 0 mag 1 1 1
0.5 0.5 0.5 m 1 1 1 mag 1 1 1

O
0.0
2
0.0 0.0 0.0 m 0 0 0 mag 1 1 1
0.5 0.5 0.5 m 1 1 1 mag 1 1 1
```

However, this autoset will not be vaild once `STRU` specalize a finite magnetic for any single atom.

- `lambda`: Lagrange multiplier vector for spin constraint method. Can specify one value (z-component) or three values for x, y, z components (e.g., `lambda 0.5` or `lambda 0.1 0.2 0.3`). Values are in eV and will be converted to Rydberg internally. Used with spin-constrained DFT (enable with `sc_mag_switch` in INPUT file).
- `sc`: set the spin constraint target magnetization for each atom. Can specify one value (z-component) or three values for x, y, z components (e.g., `sc 1.0` or `sc 0.5 0.5 1.0`). Used with spin-constrained DFT (enable with `sc_mag_switch` in INPUT file).

Important Notes for ATOMIC_POSITIONS

1. **Coordinate System Selection**: Choose the appropriate coordinate system based on your needs:
   - Use `Direct` for fractional coordinates (most common for periodic systems)
   - Use `Cartesian_angstrom` when working with molecular structures or experimental data
   - Use centered coordinate systems (`Cartesian_angstrom_center_xy/xz/yz/xyz`) for surface or slab calculations where you want to center the structure
2. **Magnetization Settings**:
   - For collinear calculations (`nspin=2`), only specify one magnetization value per atom
   - For non-collinear calculations (`nspin=4`), you can specify:
     - Three components directly: `mag 1.0 0.0 0.0` (mx, my, mz)
     - Magnitude with angles: `mag 1.0 angle1 90 angle2 0` (magnitude, polar angle, azimuthal angle)
   - If no magnetization is specified for any atom, ABACUS will automatically set default values (1.0 for nspin=2, or (1,1,1) for nspin=4)
3. **Movement Constraints**:
   - Use `m 1 1 1` to allow the atom to move freely in all directions during relaxation
   - Use `m 0 0 0` to fix the atom completely
   - Use `m 1 0 1` to allow movement only in x and z directions (useful for constraining surface atoms)

4. **Keyword Order**: The optional keywords (m, v, mag, angle1, angle2, lambda, sc) can appear in any order after the atomic coordinates, but each keyword should only appear once per atom.
5. **Common Mistakes to Avoid**:
    - Don't mix Direct and Cartesian coordinates in the same STRU file
    - Ensure the number of atoms specified matches the actual number of coordinate lines provided
    - When using vector magnetization (`mag x y z`), don't also specify angles for the same atom
    - Remember that angles are in degrees, not radians

15.3 The KPT file

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`.

15.3.1 Gamma-only Calculations

In ABACUS, we offer the 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.

15.3.2 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 reciprocal 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

15.3.3 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
```

K-point Weights and Symmetry

When explicitly setting k-points, you can specify custom weights for each k-point. These weights determine the contribution of each k-point to the total energy and density calculations.

Important notes about k-point weights:

1. **Custom weights are preserved**: When using explicit k-point lists (non-Monkhorst-Pack), ABACUS preserves the custom weights you specify, even when symmetry operations are applied to reduce the k-points to the irreducible Brillouin zone (IBZ).
2. **Symmetry reduction**: When *symmetry* is set to 1, ABACUS will analyze the crystal symmetry and reduce the k-point set to the irreducible Brillouin zone. During this reduction:
   - For **Monkhorst-Pack grids** (automatically generated): All k-points have uniform weights (1/N where N is the total number of k-points)
   - For **explicit k-point lists**: Custom weights are preserved and properly combined when symmetry-equivalent k-points are merged
3. **Weight normalization**: After symmetry reduction, k-point weights are normalized so that their sum equals `degspin` (2 for non-spin-polarized calculations, 1 for spin-polarized calculations).

Example with custom weights:

```
K_POINTS
5
Direct
0.0 0.0 0.0   0.1   // Gamma point with weight 0.1
0.5 0.0 0.0   0.2   // X point with weight 0.2
0.0 0.5 0.0   0.3   // Y point with weight 0.3
0.5 0.5 0.0   0.2   // M point with weight 0.2
0.0 0.0 0.5   0.2   // Z point with weight 0.2
```

In this example, different k-points have different weights, which might be useful for:

- Special sampling schemes
- Convergence testing with specific k-point importance
- Custom integration methods

Note: When using custom weights with symmetry, ensure that your weight distribution is consistent with the crystal symmetry. ABACUS will preserve your weights during IBZ reduction, but inconsistent weights may lead to unexpected results.

back to top

15.3.4 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

SIXTEEN

WINDOWS ONE-CLICK INSTALLER (WSL2 + CONDA-FORGE)

A lightweight installer that brings ABACUS to Windows via WSL2 and conda-forge. No C++ toolchain, no MPI build, no manual dependency juggling — run `install-abacus.bat` once and type `abacus` from any Windows terminal.

The scripts live in the repository under `tools/windows/`. This page is the user-facing documentation for the same scripts.

16.1 How it works

ABACUS depends on a heavy Linux-native scientific stack (OpenMPI, ScaLAPACK, ELPA, FFTW, libxc, OpenBLAS, …) that is painful to build natively on Windows. Instead of porting, this installer provisions a standard Linux environment inside WSL2 and exposes it through thin Windows launchers.

The pipeline, end to end:

1. **`install-abacus.bat`** (runs on Windows, requires admin)
   - Checks the Windows build (≥ 19041) and whether WSL is installed; if not, runs `wsl --install --no-launch` and asks the user to reboot once.
   - Optionally enables TUNA (Tsinghua) mirrors for users in Mainland China.
   - Prompts for an ABACUS version (blank = latest on conda-forge; an exact version like `3.7.4` is pinned; a match-spec like `>=3.7,<3.8` is passed through to conda).
   - Detects the target distribution (`Ubuntu-22.04`) by querying the WSL registry key `HKCU\Software\Microsoft\Windows\CurrentVersion\Lxss`. This is immune to UTF-16 parsing pitfalls and to Store appx leftovers that can make `wsl -d <name> -- true` falsely report success.
   - Calls `wsl --install -d Ubuntu-22.04 --no-launch` if the distro is missing, then verifies the registry entry appeared.
   - Invokes `provision.sh` inside the distribution, stripping any `\r` bytes on the fly (`sed 's/\r$//' script | bash`) so Windows line endings don't break shell parsing.
   - Writes two small launcher `.cmd` files and adds them to the user PATH via PowerShell (avoiding `setx`'s 1024-character truncation).
2. **`provision.sh`** (runs as root inside the WSL distribution)
   - Optionally rewrites `/etc/apt/sources.list` to TUNA.
   - `apt-get install`s a minimal set of prerequisites (curl, ca-certificates, bzip2).
   - Downloads the Miniforge installer (from GitHub or TUNA's GitHub-releases mirror) and installs it to `/opt/abacus-miniforge`.

- `conda create -n abacus_env -c conda-forge abacus` (or the TUNA conda-forge channel) — a single package pulls in the entire scientific runtime. conda-forge ships `abacus` for `linux-64` and `linux-aarch64`, which is exactly what WSL2 provides.
- Writes two system-wide launchers, `/usr/local/bin/abacus` and `/usr/local/bin/abacus-mpi`, that activate the env and exec the real binary. Both set `OMP_NUM_THREADS=1` by default to avoid thread oversubscription. `abacus-mpi` additionally sets OpenMPI 4/5 + PRRTE "allow run as root" environment variables and passes `--allow-run-as-root` to `mpirun`, so the default WSL root user can launch parallel jobs without creating a non-root user.

3. **Windows launchers** (`abacus.cmd`, `abacus-mpi.cmd`)
   - Added to `%LOCALAPPDATA%\ABACUS\bin` and the user PATH.
   - Each launcher sets `WSLENV=OMP_NUM_THREADS:MKL_NUM_THREADS:OPENBLAS_NUM_THREADS:...` so thread-count overrides set on the Windows side are visible inside WSL.
   - Body is just:

     ```
     wsl -d Ubuntu-22.04 --cd "%CD%" -- abacus %*
     ```

     `--cd "%CD%"` maps the current Windows directory (`C:\...\case`) to its WSL path (`/mnt/c/.../case`), so users can `cd` into a case directory in `cmd`/PowerShell/Terminal and just type `abacus`.
4. **`uninstall-abacus.bat`**
   - Reads `install-state.txt` (written by the installer) to learn whether the Ubuntu-22.04 distribution was pre-existing or added by us.
   - If it was added by us, prompts whether to `wsl --unregister` the entire distribution, or to only wipe `/opt/abacus-miniforge` and the launchers.
   - If it was pre-existing, only the ABACUS files inside are removed.
   - Cleans Windows-side launchers and removes the bin directory from the user PATH.
   - Does **not** touch WSL itself (runtime, Windows optional features, or other distributions). See *Uninstallation* below for how to fully remove WSL if you want to.

16.2 Requirements

- Windows 10 build 19041 (2004) or newer, or any Windows 11.
- Administrator privileges for the first run (to enable WSL features).
- Virtualization enabled in BIOS/UEFI.
- ~2 GB free disk space (Ubuntu + conda env).
- Network access to GitHub and conda-forge, or to TUNA if you choose the China mirror option.

16.3 Installation

1. Clone or download this repository.
2. In `tools/windows/`, right-click `install-abacus.bat` → **Run as administrator**.
3. Answer the China-mirror prompt (`y` recommended inside Mainland China). Then pick an ABACUS version when prompted (leave blank for the latest on conda-forge; for a pinned install, type an exact version such as `3.7.4`).
4. If this is the first time WSL is installed on the machine, the script will ask you to reboot and run it again.

5. Wait for `[*] Provisioning ABACUS ...` to finish (5–15 minutes on first run; most of it is the conda-forge download).
6. When you see `Installation complete!`, **open a new terminal window** (so the updated PATH takes effect) and verify:

```
abacus --version
```

16.4 Usage

Serial run in any case directory:

```
cd path\to\my_case
abacus
```

Parallel run with 4 MPI ranks:

```
abacus-mpi -n 4
```

Hybrid MPI + OpenMP (e.g. 4 MPI ranks × 2 threads each):

```
set OMP_NUM_THREADS=2
abacus-mpi -n 4
```

Set `OMP_NUM_THREADS` (and/or `MKL_NUM_THREADS`, `OPENBLAS_NUM_THREADS`) in your Windows shell and the launcher will forward the value into WSL through `WSLENV`. Unset, it defaults to 1 — a safe choice when running pure MPI.

Interactive Linux shell (for advanced debugging, manually running `mpirun`, inspecting logs, etc.):

```
wsl -d Ubuntu-22.04
```

Inside the shell you can `conda activate abacus_env` to get access to `mpirun`, `mpiexec`, and other tools from the conda environment.

16.5 Uninstallation

16.5.1 Standard: remove ABACUS only

Run `uninstall-abacus.bat`. This handles the common case:

- Removes `/opt/abacus-miniforge` and the `abacus` / `abacus-mpi` launchers inside WSL.
- If the installer added the `Ubuntu-22.04` distribution, asks whether you also want to `wsl --unregister` it (pick `y` to reclaim the disk space, `n` to keep the Linux environment for other uses).
- Deletes `%LOCALAPPDATA%\ABACUS\` and removes its `bin\` directory from your user PATH.

This is enough for almost every user. WSL itself and any *other* WSL distributions you have stay untouched — important because WSL is commonly shared with Docker Desktop, VS Code Remote, and other toolchains.

16.5.2 Nuclear: remove WSL itself

Only do this if you truly have no other use for WSL on this machine. Removing WSL will break Docker Desktop, VS Code Remote-WSL, any other Linux distros you have, and so on. Run the following in an elevated PowerShell:

```
# 1. Unregister every WSL distribution (this wipes all their files).
wsl --list --quiet | ForEach-Object { wsl --unregister $_.Trim() }

# 2. Uninstall the WSL runtime itself (wsl.exe + the Linux kernel package).
#    This is Microsoft's official command; it does not disable the Windows
#    optional features and does not touch any distro appx packages.
wsl --uninstall

# 3. Optionally remove leftover distribution appx packages from the Store
#    (e.g. "Ubuntu 22.04 LTS"). `wsl --unregister` deletes the data only;
#    the Store app that installs the distro is separate.
Get-AppxPackage *Ubuntu* | Remove-AppxPackage

# 4. Optional: disable the Windows optional features (requires a reboot).
#    Skip this step if anything else on the machine still uses Hyper-V
#    virtualization (Docker Desktop, Windows Sandbox, Hyper-V VMs, ...).
dism.exe /online /disable-feature /featurename:Microsoft-Windows-Subsystem-Linux /
↪norestart
dism.exe /online /disable-feature /featurename:VirtualMachinePlatform /norestart

# 5. Remove user config if present.
Remove-Item "$env:UserProfile\.wslconfig" -ErrorAction SilentlyContinue

# 6. Reboot to finalize.
Restart-Computer
```

After the reboot `wsl.exe` no longer exists. If you also ran step 4, the Hyper-V virtualization layer used by WSL2 is disabled.

> On older Windows builds where `wsl --uninstall` is not available (WSL shipped via the in-box `wsl.exe` rather than the Store package), use `Get-AppxPackage *WindowsSubsystemForLinux* | Remove-AppxPackage` as a fallback for step 2.

16.6 Performance notes

- Files under `/mnt/c/...` are served through the 9P protocol and are noticeably slower than native ext4. For heavy I/O (large SCF, MD trajectories), run the case from inside the WSL filesystem:

```
wsl -d Ubuntu-22.04
cp -r /mnt/c/path/to/case ~/case
cd ~/case
abacus
```

- The first `wsl` invocation after a boot triggers a 10–30 s VM cold start.
- OpenMPI runs all ranks inside a single WSL2 VM, so there is no network overhead between ranks — you get near-native parallel performance.

16.7 File layout

```
tools/windows/
├── install-abacus.bat     # Windows entry point (admin, interactive)
```

(continues on next page)

(continued from previous page)

```
├── uninstall-abacus.bat   # Clean removal, optionally including the distro
├── provision.sh           # Linux-side installer (runs as root in WSL)
├── .gitattributes         # Pin *.sh to LF, *.bat/*.cmd to CRLF
└── README.md              # Mirror of this page, shipped with the scripts
```

Artifacts created at install time:

```
%LOCALAPPDATA%\ABACUS\
├── bin\
│   ├── abacus.cmd         # Windows launcher (serial)
│   └── abacus-mpi.cmd     # Windows launcher (MPI)
└── install-state.txt      # Records whether we created the WSL distro

Inside WSL (Ubuntu-22.04):
/opt/abacus-miniforge/                      # Private Miniforge install
/opt/abacus-miniforge/envs/abacus_env/      # conda env holding abacus
/usr/local/bin/abacus, /usr/local/bin/abacus-mpi   # Linux-side launchers
```

16.8 Design choices and trade-offs

- **Why WSL2 + conda-forge instead of a native Windows build?** ABACUS's MPI + ScaLAPACK + ELPA stack has no reliable native Windows build. Going through WSL2 lets us reuse the Linux binaries conda-forge already ships, turning a multi-week porting problem into a 200-line shell script.
- **Why a dedicated `Ubuntu-22.04` distribution?** conda-forge ABACUS is built against glibc from 22.04-era Ubuntu. Using `Ubuntu` (rolling) risks mismatches; pinning the version keeps the install reproducible.
- **Why put conda under `/opt/abacus-miniforge` rather than `/root`?** Clean uninstall path, clear ownership, and doesn't interfere with a user's personal conda install if they later add one inside the same distribution.
- **Why not ship a pre-built WSL rootfs?** Would cut first-run time from ~10 min to ~1 min, but balloons the installer from a few KB of scripts to 300–500 MB, requires CI infrastructure, and needs code signing to avoid SmartScreen warnings. A scripted online installer is the lowest-maintenance starting point; the pre-built rootfs path remains open for a future v1.

CHAPTER

SEVENTEEN

HOW TO CITE

The following references are required to be cited when using ABACUS. Specifically:

- **For general purpose:**

  For LCAO basis:

  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.

  Peize Lin, Xinguo Ren, Xiaohui Liu, Lixin He. Ab initio electronic structure calculations based on numerical atomic orbitals: Basic fomalisms and recent progresses. Wiley Interdisciplinary Reviews: Computational Molecular Science, 2024, 14(1): e1687.

  For LCAO and PW basis:

  Weiqing Zhou, Daye Zheng, Qianrui Liu, et al. ABACUS: An Electronic Structure Analysis Package for the AI Era. arXiv preprint arXiv:2501.08697, 2025.

- **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." J. Phys. Chem. A 126.49 (2022): 9154-9164.

- **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.

- **If ML-KEDF is used:**

  Sun, Liang, and Mohan Chen. "Machine learning based nonlocal kinetic energy density functional for simple metals and alloys." Physical Review B 109.11 (2024): 115135.

  Sun, Liang, and Mohan Chen. "Multi-channel machine learning based nonlocal kinetic energy density functional for semiconductors." Electronic Structure 6.4 (2024): 045006.

- **If DFT-D4 dispersion correction is used:**

  Eike Caldeweyher, Sebastian Ehlert, Andreas Hansen, Hagen Neugebauer, Sebastian Spicher, Christoph Bannwarth, and Stefan Grimme. "A generally applicable atomic-charge dependent London dispersion correction." The Journal of Chemical Physics 150, 154122 (2019). DOI: 10.1063/1.5090222.

  Eike Caldeweyher, Jan-Michael Mewes, Sebastian Ehlert, and Stefan Grimme. "Extension and evaluation of the D4 London-dispersion model for periodic systems." Physical Chemistry Chemical Physics 22, 8499-8512 (2020). DOI: 10.1039/D0CP00502A.

  Nikolay V. Tkachenko, Linus B. Dittmer, Rebecca Tomann, and Martin Head-Gordon. "Smooth D4S model." The Journal of Physical Chemistry Letters 15, 10629-10637 (2024). DOI: 10.1021/acs.jpclett.4c02653.

CHAPTER

EIGHTEEN

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 (Dr. Daye Zheng)
- Institute of Artificial Intelligence, Hefei Comprehensive National Science Center (Dr. Lixin He).

CHAPTER

NINETEEN

DEVELOPERS GUIDE

This section provides guidelines and resources for developers working on the ABACUS codebase.

19.1 Basic Tool Classes Design Guide for ABACUS

19.1.1 Overview

This document provides guidelines for designing and implementing basic tool classes in the ABACUS codebase, focusing on best practices for memory management, code style, and testing. These guidelines apply to all basic mathematical and utility classes, including but not limited to:

- vector3.h
- matrix.h
- timer.h
- ndarray.h
- realarray.h
- complexarray.h
- complexmatrix.h
- matrix3.h
- intarray.h
- formatter.h
- math_chebyshev.h

While this guide uses `IntArray` as an example for illustration purposes, the principles and practices described here are applicable to all basic tool classes in ABACUS.

19.1.2 Memory Management

1. Exception Handling for Memory Allocation

Always use try-catch blocks when allocating memory to handle `std::bad_alloc` exceptions gracefully:

2. Two-Stage Memory Allocation

When reallocating memory (e.g., in `create` methods), use a two-stage approach to ensure that the original object remains valid if memory allocation fails.

3. Null Pointer Checks

Always check for null pointers before accessing memory, especially in methods that might be called on objects with failed memory allocation.

19.1.3 Class Design

1. Copy Constructor

Implement a copy constructor to avoid shallow copy issues.

2. Move Semantics

Implement move constructor and move assignment operator to improve performance.

3. Boundary Checks

Add boundary checks to prevent out-of-bounds access.

19.1.4 Code Style

1. Brace Style

Use separate lines for braces, and always use braces for "if" and "for" statements, even if they contain one line of code

2. Indentation

Use spaces instead of tabs for indentation (4 spaces per indent level).

3. Comments

Use English for comments and document important functionality. Follow Doxygen-style documentation for classes and methods.

19.1.5 Code Quality

1. Named Constants

Avoid using magic numbers. Instead, define named constants for numerical values:

2. Header Includes

Ensure all necessary header files are included, especially for functions like `assert`:

```
#include <cassert>
```

19.1.6 Testing

1. Unit Tests

Write comprehensive unit tests for all classes, including:

- Constructor tests
- Method tests
- Exception handling tests
- Edge case tests

2. Test Class Initialization

Use constructor initialization lists for test classes to improve compatibility:

```
class IntArrayTest : public testing::Test
{
protected:
    ModuleBase::IntArray a2, a3, a4, a5, a6;
    int aa;
    int bb;
    int count0;
    int count1;
    const int zero;

    IntArrayTest() : aa(11), bb(1), zero(0)
    {
    }
};
```

19.1.7 Best Practices

1. **Single Responsibility Principle**: Each class should have a single, well-defined responsibility.
2. **Encapsulation**: Hide implementation details and expose only necessary interfaces.
3. **Error Handling**: Handle errors gracefully, especially memory allocation failures.
4. **Performance**: Use move semantics and other performance optimizations where appropriate.
5. **Testing**: Write comprehensive tests for all functionality.
6. **Code Style**: Follow consistent code style guidelines, including:
    - Always use braces for if and for statements
    - Use separate lines for braces
    - Use spaces instead of tabs for indentation
    - Use English for comments
7. **Code Quality**: Maintain high code quality by:
    - Using named constants instead of magic numbers
    - Ensuring all necessary header files are included
    - Adding boundary checks to prevent out-of-bounds access
8. **Documentation**: Document classes and methods to improve maintainability.
9. **Compatibility**: Ensure code is compatible with C++11 standard.
10. **Portability**: Write code that works across different platforms.
11. **Reusability**: Design classes to be reusable in different contexts.

19.1.8 Application to Other Basic Tool Classes

While this guide uses `IntArray` as an example, these principles apply to all basic tool classes in ABACUS. For example:

- **vector3.h**: Apply the same memory management and error handling principles, with additional focus on vector operations and operator overloading.

- **matrix.h**: Extend the memory management practices to 2D arrays, with additional considerations for matrix operations.
- **timer.h**: Focus on static member management and time measurement accuracy.
- **ndarray.h**: Apply the same principles to multi-dimensional arrays, with additional considerations for shape manipulation.
- **formatter.h**: Focus on string manipulation and formatting, with attention to performance and usability.
- **math_chebyshev.h**: Apply the principles to template classes, with additional focus on mathematical algorithm implementation.

By following these guidelines, you can ensure that all basic tool classes in ABACUS are well-designed, robust, and maintainable.

CHAPTER

TWENTY

ABACUS CONTRIBUTION GUIDE

20.1 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

TWENTYONE

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*

21.1 Table of Contents

21.2 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.

21.3 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.

For those who are interested in the source code, the following figure shows the structure of the source code.

```
|-- source_base                 A basic module including
|   |                           (1) Mathematical library interface functions: BLAS,␣
↪LAPACK, Scalapack;
|   |                           (2) Custom data classes: matrix, vector definitions␣
↪and related functions;
|   |                           (3) Parallelization functions: MPI, OpenMP;
|   |                           (4) Utility functions: timer, random number generator,
↪ etc.
|   |                           (5) Global parameters: input parameters, element␣
↪names, mathematical and physical constants.
|   |-- module_container        The container module for storing data and performing␣
↪operations on them and on different architectures.
|-- source_basis                Basis means the basis set to expand the wave function.
|   |-- module_ao               Atomic orbital basis set to be refactored.
|   |-- module_nao              New numerical atomic orbital basis set for two-center␣
↪integrals in LCAO calculations
|   `-- module_pw               Data structures and relevant methods for planewave␣
↪involved calculations
|-- source_cell                 The module for defining the unit cell and its␣
↪operations, and reading pseudopotentials.
|   |-- module_neighbor         The module for finding the neighbors of each atom in␣
↪the unit cell.
|   |-- module_symmetry         The module for finding the symmetry operations of the␣
↪unit cell.
|-- source_estate               The module for defining the electronic state and its␣
↪operations.
|   |-- module_charge           The module for calculating the charge density, charge␣
↪mixing
|   |-- potentials              The module for calculating the potentials, including␣
↪Hartree, exchange-correlation, local pseudopotential, etc.
|-- source_esolver              The module defining task-specific driver of␣
↪corresponding workflow for evaluating energies, forces, etc., including lj, dp, ks,␣
↪sdft, ofdft, etc.
|   |                           TDDFT, Orbital-free DFT, etc.
|-- source_hamilt               The module for defining general Hamiltonian that can␣
↪be used both in PW and LCAO calculations.
|   |-- module_ewald            The module for calculating the Ewald summation.
|   |-- module_surchem          The module for calculating the surface charge␣
↪correction.
|   |-- module_vdw              The module for calculating the van der Waals␣
↪correction.
|   |-- module_xc               The module for calculating the exchange-correlation␣
↪energy and potential.
|-- source_lcao                 The module for defining the Hamiltonian in LCAO␣
```

(continues on next page)

(continued from previous page)

```
↪calculations.
|   |-- hamilt_lcaodft          The module for defining the Hamiltonian in LCAO-DFT␣
↪calculations.
|   |   |-- operator_lcao       The module for defining the operators in LCAO-DFT␣
↪calculations.
|   |-- module_deepks           The module for defining the Hamiltonian in DeepKS␣
↪calculations.
|   |-- module_dftu             The module for defining the Hamiltonian in DFT+U␣
↪calculations.
|   |-- module_gint             The module for performing grid integral in LCAO␣
↪calculations.
|   |-- module_hcontainer       The module for storing the Hamiltonian matrix in LCAO␣
↪calculations.
|   |-- module_rt               The module for defining the Hamiltonian in TDDFT␣
↪calculations.
|   `-- module_ri               The module for performing RI calculations.
|-- source_pw                   The module for defining the Hamiltonian in PW␣
↪calculations.
|   |-- module_ofdft            The module for defining the Hamiltonian in OFDFT␣
↪calculations.
|   |-- module_pwdft            The module for defining the Hamiltonian in PW-DFT␣
↪calculations.
|   |   |-- operator_pw         The module for defining the operators in PW-DFT␣
↪calculations.
|   `-- module_stodft           The module for defining the Hamiltonian in STODFT␣
↪calculations.
|-- source_hsolver              The module for solving the Hamiltonian with different␣
↪diagonalization methods, including CG, Davidson in PW
|   |                           calculations, and scalapack and genelpa in LCAO␣
↪calculations.
|-- source_io                   The module for reading of INPUT files and output␣
↪properties including band structure, density of states, charge density, etc.
|-- source_md                   The module for performing molecular dynamics.
|-- source_psi                  The module for defining the wave function and its␣
↪operations.
`-- source_relax                The module for performing structural optimization,␣
↪optimized for cell and ion simultaneously.
```

21.4 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.

21.5 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}
```

21.6 Documenting INPUT Parameters

ABACUS includes a built-in help system that allows users to query INPUT parameters directly from the command line (e.g., `abacus -h ecutwfc`). Parameter metadata is defined inline in the C++ source files (`source/source_io/module_parameter/read_input_item_*.cpp`) using `Input_Item` registrations.

A checked-in file `docs/parameters.yaml` contains a YAML dump of all parameter metadata, generated from the binary itself. This file is used by Sphinx to produce the online documentation page `input-main.md`.

21.6.1 When to Update `docs/parameters.yaml`

You **must** regenerate `docs/parameters.yaml` whenever you:

- Add a new INPUT parameter
- Remove an existing INPUT parameter
- Change a parameter's description, type, default value, unit, category, or availability

21.6.2 How to Regenerate

After building and installing ABACUS, run:

```
abacus --generate-parameters-yaml > docs/parameters.yaml
```

Then verify the YAML is valid:

```
python3 -c "import yaml; d=yaml.safe_load(open('docs/parameters.yaml')); print(len(d[
↪'parameters']), 'parameters')"
```

You can also regenerate the markdown documentation locally:

```
python3 docs/generate_input_main.py docs/parameters.yaml --output docs/advanced/input_
↪files/input-main.md
```

Important: Include the updated `docs/parameters.yaml` and `input-main.md` in your commit when submitting a PR that modifies INPUT parameters. Reviewers should verify the YAML changes match the C++ source changes and the `input-main.md` is updated.

21.6.3 Parameter Documentation Format

When adding or modifying INPUT parameters in C++ source, set the following fields on the `Input_Item`:

```
{
    Input_Item item("my_parameter");
    item.category = "System variables";
    item.type = "Integer";
    item.description = "Description of what this parameter does.";
    item.default_value = "0";
    item.unit = "Ry";          // Optional, empty string if no unit
    item.availability = "";    // Optional, empty string if always available
    // ... read_value, reset_value, check_value functions ...
    this->add_item(item);
}
```

Supported types: `Integer`, `Real`, `String`, `Boolean`

21.6.4 Format Validation

After regenerating the YAML, you can spot-check a specific parameter:

```
./build/abacus -h my_parameter
```

This uses the same runtime registry that generates the YAML, so if the help output looks correct, the YAML will be correct too.

21.7 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.

We use https://pre-commit.ci/ to format the code. It is performed after pushing new commits to a PR. You might need to pull the changes before adding new commits.

To use pre-commit locally (**generally not required**): Please install the pre-commit tool by running the following command:

```
pip install pre-commit
pip install clang-tidy clang-format # if you haven't installed them
```

Then, run the following command to install the pre-commit hooks:

```
pre-commit install
```

21.8 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. `source_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.
```

(continues on next page)

(continued from previous page)

```
  LIBS math_libs # `math_libs` includes all math libraries in ABACUS.
)
```

- Build with `-D BUILD_TESTING=1` flag, `cmake` will look for `GoogleTest` in the default path (usually `/usr/local`); if not found, you can specify the path with `-D GTEST_DIR`. 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`.
- Considering `-D BUILD_TESTING=1`, the compilation will be slower compared with the case `-D BUILD_TESTING=0`.

21.9 Running unit tests

1. Compiling ABACUS with unit tests.

   In order to run unit tests, ABACUS needs to be configured with `-D BUILD_TESTING=ON` flag. For example:

   ```
   cmake -B build -DBUILD_TESTING=ON
   ```

   then build ABACUS and unit testing with

   ```
   cmake --build build -j${number of processors}
   ```

   It is import to run the folloing command before running unit tests:

   ```
   cmake --install build
   ```

   to install mandatory supporting input files for unit tests. If you modified the unit tests to add new tests or learn how to write unit tests, it is convenient to run

   ```
   cmake --build build -j${number of processors} --target ${unit test name}
   ```

   to build a specific unit test. And please remember to run `cmake --install build` after building the unit test if the unit test requires supporting input files.

2. Running unit tests

   The test cases are located in `build/source/${module_name}/test` directory. Note that there are other directory names for unit tests, for example, `test_parallel` for running parallel unit tests, `test_pw` for running unit tests only used in plane wave basis calculation.

   You can run a single test in the specific directory. For example, run

   ```
   ./cell_unitcell_test
   ```

   under the directory of `build/source/source_cell/test` to run the test `cell_unitcell_test`. However, it is more convenient to run unit tests with `ctest` command under the `build` directory. You can check all unit tests by

   ```
   ctest -N
   ```

   The results will be shown as

```
Test project /root/abacus/build
Test   #1: integrated_test
Test   #2: Container_UTs
Test   #3: base_blas_connector
Test   #4: base_blacs_connector
Test   #5: base_timer
...
```

Note that the first one is integrated test, which is not a unit test. It is the test suite for testing the whole ABACUS package. The examples are located in the `tests/integrate` directory.

To run a subset of tests, run the following command

```
ctest -R <test-match-pattern> -V
```

For example, `ctest -R cell` will perform tests with name matched by `cell`. You can also run a single test with

```
ctest -R <test-name>
```

For example, `ctest -R cell_unitcell_test_readpp` will perform test `cell_unitcell_test_readpp`. To run all the unit tests, together with the integrated test, run

```
cmake --build build --target test ARGS="-V --timeout 21600"
```

in the `abacus-develop` directory.

21.10 Adding an integrate test

The integrate test is a test suite for testing the whole ABACUS package. The examples are located in the `tests/integrate` directory. Before adding a new test, please firstly read `README.md` in `tests/integrate` to understand the structure of the integrate test. To add an integrate test:

1. Add a new directory under `tests/integrate` for the new test.
2. Prepare the input files for the new test.
    - The input files should be placed in the new directory. Pseudopotential files and orbital files should be placed in `tests/PP_ORB`. You should define the correct `pseudo_dir` and `orb_dir`(if need orbital files) in INPUT with the relative path to the `tests/PP_ORB` directory, and be sure the new test can be run successfully.
    - The running time of the new test should not exceed 20 seconds. You can try to reduce the time by below methods (on the premise of ensuring the effectiveness of the test):
        - Reduce the number of atoms in the unit cell (1~2 atoms).
        - Reduce the number of k-points (`1 1 1` or `2 2 2`).
        - Reduce ecutwfc (20~50 Ry).
        - Reduce the number of steps for relax or md job (2~3 steps).
        - Reduce the basis set for LCAO calculations (DZP orbital and 6 a.u. cutoff).
    - For PW calculations, should set `pw_seed 1` in INPUT file to ensure the reproducibility of the test.
3. Generate the reference results for the new test.
    - Run the new test with GNU compiler and 4 MPI processes 2 OpenMP threads. The command is `OMP_NUM_THREADS=2 mpirun -np 4 abacus > log.txt`.

- Execute tests/integrate/tools/catch_properties.sh script to generate the reference results. At the new test directory, run `bash ../tools/catch_properties.sh result.ref`. A `result.ref` file may be like:

```
etotref -3439.007931317310
etotperatomref -3439.0079313173
totaltimeref 2.78
```

- If you want to test the correctness of some output files, you need to do extra below steps:

  1. add the corresponding comparison method in `catch_properties.sh`. For example, to verify whether the output of the BANDS_1.dat file is correct, you need to add the following code in `catch_properties.sh`:

```
has_band=$(awk '$1=="out_band" {a=$2} END{print a}' INPUT)   # check if␣
↪the BAND is outputed
if ! test -z "$has_band"  && [  $has_band == 1 ]; then       # if band is␣
↪outputed, then check if the band is correct
    bandref=refBANDS_1.dat                                   # this file␣
↪should be prepared in new test directory
    bandcal=OUT.autotest/BANDS_1.dat                         # this file␣
↪is generated by each run of test
    python3 ../tools/CompareFile.py $bandref $bandcal 8      # compare the␣
↪new band file with the reference file
    echo "CompareBand_pass $?" >>$1                          # record the␣
↪comparison result, $? is the return value of last command
fi
```

     `CompareFile.py` is used to determine if two files are identical. It accepts three arguments: the first two are the files to be compared, and the third specifies the precision for comparing numerical values. The comparison fails if the difference between any two corresponding numerical values exceeds 1e-{precision} (such as: 1e-8 in previous case). If the files are identical, the script returns 0; otherwise, it returns 1.

  2. Add the reference file (such as: `refBANDS_1.dat` in previous case) to the new test directory.

  3. Add the reference comparison result to the `result.ref` file. For example, `CompareBand_pass 0` means the comparison of the band file is passed. (This statement should be added before the `totaltimeref` line)

4. Add a `jd` file in the new test directory, which is one setence to describe the new test.

5. Add the new test to `tests/integrate/CASES_CPU.txt` file (or `tests/integrate/CASES_GPU.txt` file if it is for GPU verion).

6. Enter directory tests/integrate and run `bash Autotest.sh -r <the-new-test-name>` to check if the new test can be run successfully.

21.11 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.

```
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.

21.12 Adding a new building component

ABACUS uses CMake as its default building system. To add a new building component:

1. Add an `OPTION` to toggle the component to the `CMakeLists.txt` file under root directory. For example:

```
OPTION(ENABLE_NEW_COMPONENT "Enable new component" OFF)
```

2. Add the new component. For example:

```
IF (ENABLE_NEW_COMPONENT)
  add_subdirectory(module_my_new_feature) # if the feature is implemented in a␣
↪subdirectory
  find_package(NewComponent REQUIRED) # if third-party libs are required
  target_link_libraries(${ABACUS_BIN_NAME} PRIVATE NewComponent) # if the␣
↪component is linked
  include_directories(${NewComponent_INCLUDE_DIRS}) # if the component is included
endif()
```

3. Add the required third-party libraries to Dockerfiles.
4. After the changes above are merged, submit another PR to build and test the new component in the CI pipeline.
   - For integration test and unit test: add `-DENABLE_NEW_COMPONENT=ON` to the building step at `.github/workflows/test.yml`.
   - For building test: add `-DENABLE_NEW_COMPONENT=ON` as a new configuration at `.github/workflows/build_test_cmake.yml`.

21.13 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"
```

If configuration fails unfortunately, you can find required files (including three *.cmake and llvm-cov-wrapper), and copy these four files into `/abacus-develop/cmake`. Alternatively, you can define the path with option `-D CMAKE_CURRENT_SOURCE_DIR`.

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.

21.14 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. If more changes are needed, you can add more commits to your branch and push them to GitHub. Your PR will be updated automatically.
8. 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
```

21.15 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

TWENTYTWO

LISTS OF CONTINUOUS INTEGRATION (CI) ACTIONS

The directory `.github/workflows` contains the continuous integration (CI) actions for the project. The actions are written in YAML format and are executed by GitHub Actions. Check the Actions page of the repo for the status of the actions.

22.1 On Pull Request (PR)

The following CI actions are triggered on pull request (PR) creation or update (the user pushes a new commit to the incoming branch):

- Integration test and unit tests (`test.yml`): This action builds ABACUS with all available features, runs integration tests, and runs unit tests. It also performs *static analysis* with <pre-commit.ci>.
- Building tests with CMake (`build_test_cmake.yml`) and Makefile (`build_test_makefile.yml`): This action builds ABACUS with each feature separately, ensuring: (i) there are no conflicts between features, (ii) the compilation is successful whether the feature is enabled, and (iii) it works well on multiple platforms, i.e. with GNU+OpenBLAS toolchain and Intel+MKL toolchain.
- Rerender the docs site: This action rerenders the documentation site on Read the Docs. It is automatically triggered when the documentation is updated.
- Testing GPU features (`cuda.yml`): This action builds ABACUS with GPU support and runs several tests on the GPU.

  Some tests are executed on self-hosted runners, which are maintained by the deepmodeling community. To save resources, the actions triggered by new commits may cancel the previous actions in the same PR.

22.2 On PR Merge

After the PR merges into the main branch, the following actions are triggered:

- Building Docker images (`devcontainer.yml`): This action builds the Docker images with latest codes and executables. The images are tagged as `abacus-gnu:latest`, `abacus-intel:latest`, and `abacus-cuda:latest`, and then pushed to the GitHub Container Registry (`ghcr.io/deepmodeling`) and AliCloud Container Registry (`registry.dp.tech/deepmodeling`, recommended for one having issue connecting to GitHub). For example: `docker pull ghcr.io/deepmodeling/abacus-intel:latest`.
- Generate doxygen site (`doxygen.yml`): This action generates the Doxygen site for the project. The site is published on GitHub Pages.
- Mirror the repo to Gitee (`mirror.yml`): This action mirrors the repo to Gitee.

22.3 On Routine

- Dynamic analysis (`dynamic_analysis.yml`): This action runs integration tests with AddressSanitizer to detect memory errors. The action is scheduled to run **every Sunday**. The results are published to the dashboard branch.

22.4 On Release

- Coverage test (`coverage.yml`): This action builds ABACUS with all available features, runs integration tests, and runs unit tests. It also measures the code coverage of the tests. The results are published at codecov.io.
- Building tagged Docker images (`devcontainer.yml`): Same as that action above; in addition the built image is tagged in the pattern of `abacus-intel:3.6.0`. For example: `docker pull ghcr.io/deepmodeling/abacus-intel:3.6.0`.

CHAPTER

TWENTYTHREE

FREQUENTLY ASKED QUESTIONS

23.1 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.

2. How to contact the ABACUS community?

Users can contact the ABACUS community by join ABACUS WeChat or QQ group. One can quickly join the QQ group via group ID: 759914681

If users have any questions or suggestions, please feel free to contact us in group or by raising issue in GitHub repo.

23.2 Installation

23.3 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?

To perform SOC calculations in ABACUS, follow these steps:

1. **Set `lspinorb=1` in the INPUT file**: This enables spin-orbit coupling effects
2. **Use full-relativistic pseudopotentials**: SOC calculations require pseudopotentials with `has_so=true` in the UPF header
    - Download full-relativistic versions of SG15_ONCV pseudopotentials from quantum-simulation.org
    - Check the UPF file header for `relativistic="full"` and `has_so="T"`
3. **Verify automatic settings**: When `lspinorb=1` is set, `nspin` is automatically set to 4 and symmetry is automatically disabled

Basis set support: Both `basis_type=pw` (plane wave) and `basis_type=lcao` (numerical atomic orbitals) support SOC calculations for both SCF and NSCF.

Force and stress calculations: Atomic forces and cell stresses can be calculated with SOC (supported since ABACUS v3.9.0).

Numerical atomic orbitals: The numerical orbital files generated from scalar-relativistic pseudopotentials (available at abacus.ustc.edu.cn) can be used with full-relativistic pseudopotentials, as orbitals are spin-independent.

Common error: If you see "no soc upf used for lspinorb calculation", ensure you are using full-relativistic pseudopotentials with `has_so=true`.

For detailed information, examples, and troubleshooting, see *Spin-polarization and SOC*.

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.

9. How to estimate the max memory consumption?

Run `/usr/bin/time -v mpirun -n 4 abacus`, and locate "Maximum resident set size" in the output log at the end. Please note that this value is the peak memory size of the main MPI process.

10. Why there are two sigma (smearing_sigma and dos_sigma) in some examples for dos calculation?

The tag `smearing_sigma` is used for SCF calculation, and does not affect NSCF calculation. The tag `dos_smearing` is only used for plotting density of states, which does affect SCF or NSCF results. So `smearing_sigma` should not be set in dos calculation.

11. How to set `nbands` and `ncpus`?

For both pw and LCAO calculations, the default value for `nbands` can be found here. Note that the number of CPUs called for a parallel job (i.e., the number specified after the command `mpirun -n`) should be smaller than `nbands`,

otherwise the job will fail with an error message `nbands < ncpus`. Note also that for LCAO calculations, `nbands` should always be smaller than `nlocal`, i.e., the number of the atomic orbital basis of the system.

back to top

23.4 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 "1.0".

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.

4. Error "Read -1" when using mpirun in docker environment This is a known issue of OpenMPI running in a docker container. In this case, please set environment variable `OMPI_MCA_btl_vader_single_copy_mechanism=none`.

23.5 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:

```
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'}
```

(continues on next page)

(continued from previous page)

```
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.

4. Why EDIFF is much slower than DRHO?

For metaGGA calculations, it is normal because in addition to charge density, kinetic density also needs to be considered in metaGGA calculations. In this case, you can try set `mixing_tau = true`. If you find EDIFF is much slower than DRHO for non-metaGGA calculations, please start a new issue to us.

back to top