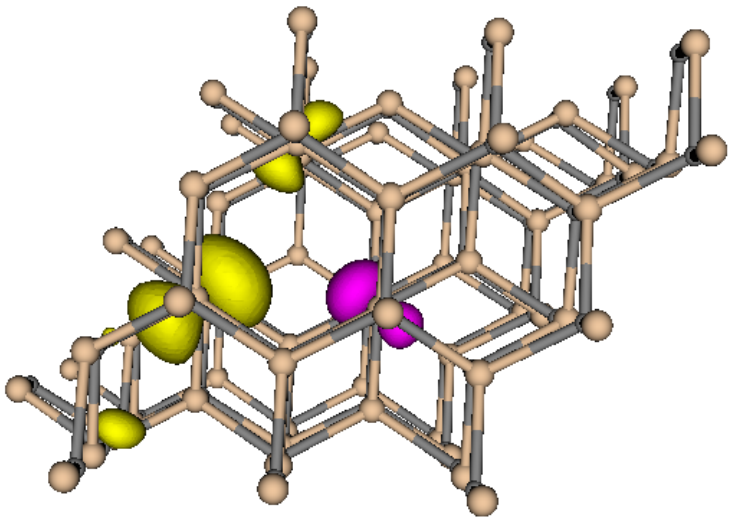
DFT-FE

Density Functional Theory with Finite-Elements



User Manual

Version 0.5.0-pre

(generated July 15, 2018)

Sambit Das
Vikram Gavini
Phani Motamarri

with contributions by:
Krishnendu Ghosh

Website of DFT-FE

Contents

1 Introduction

DFT-FE is a C++ code for material modeling from first principles using Kohn-Sham density functional theory. It is based on adaptive finite-element based methodologies and handles all-electron and pseudopotential calculations in the same framework while accommodating arbitrary boundary conditions. dft-fe code builds on top of the deal.II library for everything that has to do with finite elements, geometries, meshes, etc., and, through deal.II on p4est for parallel adaptive mesh handling.

1.1 Authors

DFT-FE is hosted by the Computational Materials Physics group at University of Michigan, with Vikram Gavini, Associate Professor of Mechanical Engineering and Materials Science and Engineering, as the principal investigator broadly overseeing this effort. The code is maintained by a group of principal developers, who manage the architecture of the code and the core functionalities. Developers with significant contributions to core functionalities and code architecture in the past who are no longer active principal developers, are listed under principal developers emeriti. A subset of the principal developers and mentors are administrators. Finally, all contributors who have contributed major parts of the DFT-FE code or sent fixes and small enhancements are listed under contributors. All the underlying lists are in alphabetical order.

Principal developers

- Sambit Das (University of Michigan, USA).
- Phani Motamarri (University of Michigan, USA).

Prinicipal developers emeriti

- Krishnendu Ghosh (University of Michigan, USA).
- Shiva Rudraraju (University of Wisconsin Madison, USA).

Contributors

- Sambit Das (University of Michigan, USA).
- Denis Davydov (University of Erlangen-Nuremberg, Germany).
- Krishnendu Ghosh (University of Michigan, USA).
- Phani Motamarri (University of Michigan, USA).
- Shiva Rudraraju (University of Wisconsin Madison, USA).
- Sukhan Parekh (University of Michigan, USA).

Mentors

- Vikram Gavini (University of Michigan, USA).

1.2 Acknowledgments

The development of DFT-FE has been funded ...

1.3 Referencing DFT-FE

2 Overview of the formulation

Formulation...

3 Installation

All the underlying installation instructions assume a Linux operating system. Also standard tools and libraries like CMake, compilers- (C, C++ and Fortran), and MPI libraries are assumed to be pre-installed. Most high-performance computers would have the latest version of these libraries in the default environment. However, in many cases you would have to use Environment Modules to set the correct environment variables for compilers-(C, C++ and Fortran), MPI libraries, and compilation tools like CMake. For example, on one of the high-performance computers we use to develop and test the DFT-FE code, we use the following commands to set the desired environment variables

```
$ module load cmake
$ module load intel/18.0.1
$ module load openmpi/3.0.0/intel/18.0.1
```

We strongly recommend using the latest stable version of compilers-(C, C++ and Fortran), and MPI libraries (only for DFT-FE and external libraries which require MPI) available on your high-performance computer. Furthermore, the installations which use CMake, version 2.8.12 or later is required.

3.1 Compiling and installing external libraries

DFT-FE is primarily based on the open source finite element library deal.II, through which external dependencies on p4est, PETSc, SLEPc, and ScaLAPACK. ScaLAPACK is an optional requirement, but strongly recommended for large problem sizes with 5000 electrons or more. The other required external libraries, which are not interfaced via deal.II are ALGLIB, Libxc, spglib, and Libxml2. Some of the above libraries (PETSc, SLEPc, ScaLAPACK, and Libxml2) are already installed on most high-performance computers. Below, we give brief installation and/or linking instructions for each of the above libraries.

3.1.1 Instructions for deal.II's dependencies p4est, PETSc, SLEPc, and ScaLAPACK

First, the installation instructions for the dependencies of deal.II:

1. **p4est**: This library is used by deal.II to distribute create and distribute finite-element meshes across multiple processors. Download the latest release tarball of p4est from `http://www.p4est.org/`, and follow the installation instructions in `https://www.dealii.org/9.0.0/external-libs/p4est.html`.

2. **PETSc**: PETSc is a parallel linear algebra library. DFT-FE needs two variants of the PETSc installation- one with real scalar type and the another with complex scalar type. Also 64-bit indices must be enabled in both the installation variants. To install PETSc, first download the latest release tarball from `https://www.mcs.anl.gov/petsc/download/index.html`. Next unpack using

   ```
   $ tar -zxvf tarball.tar.gz
   ```

   , enter into the directory and follow the installation instructions in `http://slepc.upv.es/documentation/instal.htm`. You have to install two variants of PETSc, a real scalar type with 64b-bit indices installation configured (do `./configure --help` for more information about configuration flags) using

   ```
   $ ./configure --with-debugging=no --with-64-bit-indices=true
                 [other flags- see below for example]
   ```

and a complex scalar type with 64-bit indices installation configured using

```
$ ./configure --with-debugging=no --with-64-bit-indices=true
              --with-scalar-type=complex
              [other flags- see below for example]
```

After the configuration step, follow the prompts from the terminal output to compile and install PETSc (see example below).

Below, we show an example installation for the real scalar type variant. This example should be used only as a reference.

```
$ ./configure --prefix=petsc_installation_dir_path --with-debugging=no
              --with-64-bit-indices=true --with-cc=c_compiler
              --with-cxx=c++_compiler --with-fc=fortran_compiler
              --with-blas-lapack-lib=(optimized BLAS-LAPACK library path)
              CFLAGS=c_compilter_flags CXXFLAGS=c++_compiler_flags
              FFLAGS=fortran_compiler_flags

$ make PETSC_DIR=whatever prompted by PETSc
       PETSC_ARCH=whatever prompted by PETSc

$ make PETSC_DIR=whatever prompted by PETSc
       PETSC_ARCH=whatever prompted by PETSc
       install
```

Please notice that we have used place holders for values of some of the above configuration flags. You have to use the correct values specific to the compilers and MPI libraries you are working with, and based on compiling recommendations for the high-performance computer you are working with. For example, if using Intel compilers and Intel MKL for blas-lapack, it is **very important** to use Intel MKL Link Line Advisor to set the appropriate path for `--with-blas-lapack-lib`, something like

```
$ --with-blas-lapack-lib="-Wl,--start-group
  ${MKLROOT}/lib/intel64/libmkl_intel_lp64.a
  ${MKLROOT}/lib/intel64/libmkl_intel_thread.a
  ${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group
  -liomp5 -lpthread -lm -ldl"
```

To exploit performance benifit from threads, we recommend (strongly recommended for the new Intel Xeon Phi processors codenamed Knights Landing) linking to threaded versions of Intel MKL libraries by using the options "threading layer" and "OpenMP library" in Intel MKL Link Line Advisor. The above recommendations for Intel MKL also hold true for ScaLAPACK and deal.II installations below. Finally, for more detalied installation instructions for PETSc see `http://slepc.upv.es/documentation/instal.htm`.

3. **SLEPc**: The SLEPc library is built on top of PETSc, and it is used for solution of large scale sparse eigenvalue problems on parallel computers. To install SLEPc, first download the latest release tarball from `http://slepc.upv.es/download/`, and then follow the installation procedure described in `http://slepc.upv.es/documentation/instal.htm`. **Important:** SLEPc installation requires PETSc to be installed first. You also need to create two separate SLEPc installations- one for PETSc installed with `--with-scalar-type=real`, and the second for PETSc installed with `--with-scalar-type=complex`.

4. **ScaLAPACK**: ScaLAPACK library is used by DFT-FE via deal.II for its parallel linear algebra routines for solving dense linear systems. ScaLAPACK already installed in most high-performance computers. For example, in case of Intel MKL the paths for ScaLAPACK and BLACS (ScaLAPACK requires this) libraries would be something like (obtained via Intel MKL Link Line Advisor)

   ```
   ${MKLROOT}/lib/intel64/libmkl_scalapack_lp64.so
   ${MKLROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.so
   ```

   where `$MKLROOT` points to the directory path for Intel MKL library. The above example is shown for Intel MPI library. For Open MPI library, the BLACS path would become something like

   ```
   ${MKLROOT}/lib/intel64/libmkl_blacs_openmpi_lp64.so
   ```

   If you need to install ScaLAPACK library, download the latest release version from http://www.netlib.org/scalapack/#_software, and build a shared library (use `BUILD_SHARED_LIBS=ON` and `BUILD_STATIC_LIBS=OFF` during the `cmake` configuration) installation of ScaLAPACK using `cmake`. BLACS library, which is required for linking to Intel MKL ScaLAPACK, is not required to be installed separately as it is compiled along with the ScaLAPACK library. Hence you just have to link to `/your_scalapack_installation_dir/lib/libscalapack.so` while compiling deal.II library. **Important:** for best performance, ScaLAPACK must be linked to optimized BLAS-LAPACK libraries by using `USE_OPTIMIZED_LAPACK_BLAS=ON`, and providing external paths to BLAS-LAPACK during the `cmake` configuration.

3.1.2 Instructions for deal.II

Assuming the above dependencies are installed, we now briefly discuss the steps to compile and install the deal.II library linked with the above dependencies:

1. Obtain the latest release version of deal.II by downloading and unpacking the .tar.gz file from https://www.dealii.org/download.html. Alternatively (recommended as certain performance enhancements specific to DFT-FE are missing in the latest release version), you may obtain the forked development version of deal.II library via

   ```
   $ git clone -b fixIntelCompilation https://github.com/dftfeDevelopers/dealii.git
   ```

   The above forked version has a minor fix which resolves compilation issues when using intel compilers. Once this fix is available in the deal.II development repository, users can obtain the development version of deal.II library via

   ```
   $ git clone https://github.com/dealii/dealii.git
   ```

2. ```
   $ mkdir build
   $ cd build
   $ cmake -DCMAKE_INSTALL_PREFIX=dealii_install_dir_path otherCmakeOptions ../deal.II
   $ make install
   ```

   "otherCmakeOptions" **must include** the following options

   ```
   -DDEAL_II_WITH_MPI=ON -DDEAL_II_WITH_64BIT_INDICES=ON
   -DP4EST_DIR=p4est_install_dir_path
   -DDEAL_II_WITH_PETSC=ON -DDEAL_II_WITH_SLEPC=ON
   ```

   and LAPACK and ScaLAPACK link options.
   ```

For your reference, below we provide an example of deal.II installation, which we did on a high-performance computer (STAMPEDE2) using Intel compilers and Intel MPI library

```
$ mkdir build
$ cd build
$ cmake -DCMAKE_C_COMPILER=mpicc -DCMAKE_CXX_COMPILER=mpicxx
  -DCMAKE_Fortran_COMPILER=mpif90 -DDEAL_II_CXX_FLAGS_RELEASE=-xMIC-AVX512
  -DDEAL_II_CXX_FLAGS_DEBUG=-xMIC-AVX512  -DDEAL_II_COMPONENT_EXAMPLES=OFF
  -DDEAL_II_WITH_MPI=ON -DDEAL_II_WITH_64BIT_INDICES=ON
  -DP4EST_DIR=p4est_install_dir_path
  -DDEAL_II_WITH_PETSC=ON
  -DPETSC_DIR=petsc_install_dir_path
  -DDEAL_II_WITH_SLEPC=ON
  -DSLEPC_DIR=petsc_install_dir_path
  -DDEAL_II_WITH_LAPACK=ON
  -DLAPACK_DIR="${MKLROOT}/lib/intel64" -DLAPACK_FOUND=true
  -DLAPACK_LIBRARIES="${MKLROOT}/lib/intel64/libmkl_intel_lp64.so;
  ${MKLROOT}/lib/intel64/libmkl_core.so;${MKLROOT}/lib/intel64/libmkl_intel_thread.so"
  -DLAPACK_LINKER_FLAGS="-liomp5 -lpthread -lm -ldl"
  -DSCALAPACK_DIR="${MKLROOT}/lib/intel64"
  -DSCALAPACK_LIBRARIES="${MKLROOT}/lib/intel64/libmkl_scalapack_lp64.so;
  ${MKLROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.so"
  -DCMAKE_INSTALL_PREFIX=dealii_install_dir_path
  ../dealii
$ make -j 8
$ make install
```

The values for `-DLAPACK_DIR`,`-DLAPACK_LIBRARIES`, `-DLAPACK_LINKER_FLAGS`,`-DSCALAPACK_DIR`, and `-DSCALAPACK_LIBRARIES` were obtained with the help of Intel MKL Link Line Advisor.

For more information about installing deal.II library refer to `https://dealii.org/developer/readme.html`. If you are installing the development version of the deal.II library, sometimes you may run into cross platform compilation issues. If you face such an issue, and/or if you have any questions about the deal.II installation, please contact the deal.II developers at deal.II mailing lists.

It is very important to ensure that deal.II and its dependencies (p4est, PETSc, SLEPc, and ScaLAPACK), are compiled with the same compilers, BLAS-LAPACK libraries, and MPI libraries to prevent deal.II compilation issues, occurence of runtime crashes, and DFT-FE performance degradation.

3.1.3 Instructions for ALGLIB, Libxc, spglib, and Libxml2

1. **ALGLIB**: Used by DFT-FE for spline fitting. Download the latest release of the Alglib free C++ edition from `http://www.alglib.net/download.ph`. After downloading and unpacking, go to `cpp/src`, and create a shared library using a C++ compiler. For example, using g++ compiler do

   ```
   $ g++ -c -fPIC *.cpp
   $ g++ *.o -shared -o libAlglib.so
   ```

2. **Libxc**: Used by DFT-FE for exchange-correlation functionals. Download the latest release from `http://www.tddft.org/programs/libxc/download/`, and follow the recommended installation procedure (using `./configure`) described in `http://www.tddft.org/programs/libxc/installation/`.

3. **spglib**: Used by DFT-FE to find crystal symmetries. To install spglib, first obtain the development version of spglib from their github repository by

```
$ git clone https://github.com/atztogo/spglib.git
```

and next follow the "Compiling using cmake" installation procedure described in https://atztogo.github.io/spglib/install.html.

4. **Libxml2**: Libxml2 is used by DFT-FE to read `.xml` files. Most likely, Libxml2 is already installed in the high-performance computer you are working with. It is usually installed in the default locations like `/usr/lib64` (library path) and `/usr/include/libxml2` (include path). Libxml2 can also be installed by doing

   ```
   $ git clone git://git.gnome.org/libxml2
   ```

   and following the installation instructions in the README.

3.2 Obtaining and Compiling DFT-FE

Assuming that you have already installed the above external dependencies, next follow the steps below to obtain and compile DFT-FE.

1. Download the source code of the latest release of DFT-FE from here. After downloading, unpack the file using the command

   ```
   $ tar -zxvf dftfe-0.5.0.tar.gz
   ```

   For obtaining the development version, clone the development branch of our github repository directly on the command line via

   ```
   $ git clone -b publicGitHubDevelop https://github.com/dftfeDevelopers/dftfe
   ```

2. `$ cd dftfe`

3. Set paths to external libraries (deal.II, ALGLIB, Libxc, spglib, and Libxml2), compiler options, and compiler flags in `setup.sh`, which is a script to compile DFT-FE. For your reference, a few example `setup.sh` scripts are provided in the `/helpers` folder.

   Please make sure to set `withIntelMkl=ON` in setup.sh if you have installed deal.II by linking with Intel MKL library, otherwise set it to `OFF`.

4. Set `optimizedFlag=1` in `setup.sh` and do

   ```
   $ ./setup.sh
   ```

   which compiles DFT-FE in the release mode (the fast version). If compilation is successful, a `/build` directory will be created with the following executables:

   ```
   /build/release/real/main
   /build/release/complex/main
   ```

   One could also set `optimizedFlag=0` in `setup.sh` and do

   ```
   $ ./setup.sh
   ```

   which will create debug mode executables (useful for debugging purposes but much slower than the release mode executable) in the folders

   ```
   /build/debug/real/main
   /build/debug/complex/main
   ```

4 Running DFT-FE

After compiling DFT-FE as described above, we have now the `real/main` executable, which uses real data-structures for the Kohn-Sham DFT eigen solve. This is sufficient for fully non-periodic problems, and periodic and semi-periodic problems with only one Brillouin zone sampling point at the origin. The other executable is `complex/main`, which uses complex data-structrues for the Kohn-Sham DFT eigen solve. This is required for periodic and semi-periodic problems with multiple Brillouin zone sampling points. These executables are to be used as follows:

```
./main parameterFile.prm
```

or, for a parallel program:

```
mpirun -n N ./main parameterFile.prm
```

to run with N processors.

4.1 Structuring the input file

In the above, an input file with `.prm` extension is used. This file contains input parameters as described in Section A, which can be of multiple types (`string, double, integer, bool etc.`). All input parameters are also conveniently indexed at the end of this manual in Section A.14. There are two types of parameters: "*Global parameters*" and "*Parameters in section A/B/..*". This can be seen directly in Section A, where each parameter belongs to a group of parameters under the headings: "*Global parameters*" or "*Parameters in section A/B/..*". In "*Parameters in section A/B/..*", *A* refers to the primary subsection name, *B* if present refers to a subsection inside *A*, and so on.

First, lets consider how to use a parameter named `PARAMETER xyz` under the heading "*Global parameters*". To set it to a value, say `value` in the `.prm` file, directly use

```
set PARAMETER xyz=value
```

Next consider a parameter named `PARAMETER xyzA` under the heading "*Parameters in section A*". To set it to a value, say `value` in the `.prm` file, use

```
subsection A
  set PARAMETER xyzA=value
end
```

Finally, consider a nested parameter named `PARAMETER xyzAB` under the heading "*Parameters in section A/B*". To set it to a value, say `value` in the `.prm` file, use

```
subsection A
  subsection B
    set PARAMETER xyzAB=value
  end
end
```

Couple of final notes- more than one parameter could be used inside the same `subsection`. For example

```
subsection A
  set PARAMETER SUBSECTION xyzA1=value1
  set PARAMETER SUBSECTION xyzA2=value2
  subsection B
    set PARAMETER SUBSUBSECTION xyzAB1=value1
    set PARAMETER SUBSUBSECTION xyzAB2=value2
  end
end
```

, and indentation used in the above examples is only for readability.

4.2 Demo examples walkthrough

4.2.1 Example 1

Please follow the examples in the `/dftfe/demo/` folder.

4.2.2 Example 2

To be written

5 Contributing to DFT-FE's development

Discussion on how to contribute to DFT-FE and the expected contributing standards go here. For obtaining the development version, clone the development branch of our github repository directly on the command line via

```
git clone -b publicGitHubDevelop https://github.com/dftfeDevelopers/dftfe
```

6 Future plans for DFT-FE

Some bullet points. To be expanded upon

- Adaptive meshing
- Implement improved mixing strategies to decrease the number of SCF iterations.
- Implement localization strategies to reduce computational cost for for large domain sizes (>1000 atoms).
- Enriched FEM to reduce computational cost of all-electron calculations.
- Post-processing tools.

7 Finding answers to more questions

If you have questions that go beyond this manual, there are a number of resources:

- For questions about the source code of DFT-FE, portability, installation, etc., use the DFT-FE mailing list at `dft-fe.users@umich.edu`.
- DFT-FE is primarily based on the deal.II library. If you have particular questions about deal.II, contact the mailing lists described at `https://www.dealii.org/mail.html`.
- If you have specific questions about DFT-FE that are not suitable for public and archived mailing lists, you can contact the primary developers and mentors:
  - Phani Motamarri: `phanim@umich.edu`.
  - Sambit Das: `dsambit@umich.edu`.
  - Vikram Gavini: `vikramg@umich.edu` (Mentor).

A Run-time input parameters

The underlying description of the input parameters also includes a "Standard/Developer" label, which signifies whether an input parameter is a standard one, or is an expert level one for development purposes. The default values of the development level parameters are good enough for almost all cases. For user convenience, all input parameters are also indexed at the end of this manual in Section A.14.

A.1 Global parameters

- *Parameter name:* `REPRODUCIBLE OUTPUT`

  Default: false

  Description: [Developer] Limit output to that which is reprodicible, i.e. don't print timing or absolute paths. This parameter is only used for testing purposes.

  Possible values: A boolean value (true or false)

- *Parameter name:* `VERBOSITY`

  Default: 1

  Description: [Standard] Parameter to control verbosity of terminal output. Ranging from 1 for low, 2 for medium (prints eigenvalues and fractional occupancies at the end of each ground-state solve), 3 for high (prints eigenvalues and fractional occupancies at the end of each self-consistent field iteration), and 4 for very high, which is only meant for code development purposes. VERBOSITY=0 is only used for unit testing and shouldn't be used by standard users.

  Possible values: An integer n such that $0 \leq n \leq 4$

- *Parameter name:* `WRITE DENSITY`

  Default: false

  Description: [Standard] Writes KSDFT ground state (last ground state solve in case of geometry optimization) electron-density solution fields (FEM mesh nodal values) to densityOutput.vtu file for visualization purposes. The electron-density solution field in densityOutput.vtu is named density. In case of spin-polarized calculation, two additional solution fields- density_0 and density_1 are also written where 0 and 1 denote the spin indices. Default: false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `WRITE WFC`

  Default: false

  Description: [Standard] Writes KSDFT ground state (last ground state solve in case of geometry optimization) wavefunction solution fields (FEM mesh nodal values) to wfcOutput.vtu file for visualization purposes. The wavefunction solution fields in wfcOutput.vtu are named wfc_s_k_i in case of spin-polarized calculations and wfc_k_i otherwise, where s denotes the spin index (0 or 1), k denotes the k point index starting from 0, and i denotes the Kohn-Sham wavefunction index starting from 0. Default: false.

  Possible values: A boolean value (true or false)

A.2 Parameters in section `Boundary conditions`

- *Parameter name:* `PERIODIC1`

  Default: false

  Description: [Standard] Periodicity along the first domain bounding vector.

  Possible values: A boolean value (true or false)

- *Parameter name:* `PERIODIC2`

  Default: false

  Description: [Standard] Periodicity along the second domain bounding vector.

  Possible values: A boolean value (true or false)

- *Parameter name:* `PERIODIC3`

  Default: false

  Description: [Standard] Periodicity along the third domain bounding vector.

  Possible values: A boolean value (true or false)

- *Parameter name:* `SELF POTENTIAL RADIUS`

  Default: 0.0

  Description: [Developer] The radius (in a.u) of the ball around an atom on which self-potential of the associated nuclear charge is solved. For the default value of 0.0, the radius value is automatically determined to accomodate the largest radius possible for the given finite element mesh. The default approach works for most problems.

  Possible values: A floating point number v such that $0 \leq v \leq 10$

A.3 Parameters in section `Brillouin zone k point sampling options`

- *Parameter name:* `USE GROUP SYMMETRY`

  Default: false

  Description: [Standard] Flag to control whether to use point group symmetries. Currently this feature cannot be used if ION FORCE or CELL STRESS input parameters are set to true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE TIME REVERSAL SYMMETRY`

  Default: false

  Description: [Standard] Flag to control usage of time reversal symmetry.

  Possible values: A boolean value (true or false)

- *Parameter name:* `kPOINT RULE FILE`

  Default:

  Description: [Developer] File specifying the k-Point quadrature rule to sample Brillouin zone. CAUTION: This option is only used for postprocessing, for example band structure calculation. To set k point rule for DFT solve use the Monkhorst-Pack (MP) grid generation.

  Possible values: Any string

A.4 Parameters in section `Brillouin zone k point sampling options/Monkhorst-Pack (MP) grid generation`

- *Parameter name:* `SAMPLING POINTS 1`

  Default: 1

  Description: [Standard] Number of Monkhorst-Pack grid points to be used along reciprocal latttice vector 1.

  Possible values: An integer n such that $1 \leq n \leq 1000$

- *Parameter name:* `SAMPLING POINTS 2`

  Default: 1

  Description: [Standard] Number of Monkhorst-Pack grid points to be used along reciprocal latttice vector 2.

  Possible values: An integer n such that $1 \leq n \leq 1000$

- *Parameter name:* `SAMPLING POINTS 3`

  Default: 1

  Description: [Standard] Number of Monkhorst-Pack grid points to be used along reciprocal latttice vector 3.

  Possible values: An integer n such that $1 \leq n \leq 1000$

- *Parameter name:* `SAMPLING SHIFT 1`

  Default: 0

  Description: [Standard] If fractional shifting to be used (0 for no shift, 1 for shift) along reciprocal latttice vector 1.

  Possible values: An integer n such that $0 \leq n \leq 1$

- *Parameter name:* `SAMPLING SHIFT 2`

  Default: 0

  Description: [Standard] If fractional shifting to be used (0 for no shift, 1 for shift) along reciprocal latttice vector 2.

  Possible values: An integer n such that $0 \leq n \leq 1$

- *Parameter name:* `SAMPLING SHIFT 3`

  Default: 0

  Description: [Standard] If fractional shifting to be used (0 for no shift, 1 for shift) along reciprocal latttice vector 3.

  Possible values: An integer n such that $0 \leq n \leq 1$

A.5 Parameters in section `Checkpointing and Restart`

- *Parameter name:* `CHK TYPE`

  Default: 0

  Description: [Standard] Checkpoint type, 0(dont create any checkpoint), 1(create checkpoint for geometry optimization restart if ION OPT or CELL OPT is set to true. Currently, checkpointing and restart framework doesn't work if both ION OPT and CELL OPT are set to true- the code will throw an error if attempted.), 2(create checkpoint for scf restart. Currently, this option cannot be used if geometry optimization is being performed. The code will throw an error if this option is used in conjunction with geometry optimization.)

  Possible values: An integer n such that $0 \leq n \leq 2$

- *Parameter name:* `RESTART FROM CHK`

  Default: false

  Description: [Standard] Boolean parameter specifying if the current job reads from a checkpoint. The nature of the restart corresponds to the CHK TYPE parameter. Hence, the checkpoint being read must have been created using the same value of the CHK TYPE parameter. RESTART FROM CHK is always false for CHK TYPE 0.

  Possible values: A boolean value (true or false)

A.6 Parameters in section `DFT functional parameters`

- *Parameter name:* `EXCHANGE CORRELATION TYPE`

  Default: 1

  Description: [Standard] Parameter specifying the type of exchange-correlation to be used: 1(LDA: Perdew Zunger Ceperley Alder correlation with Slater Exchange[PRB. 23, 5048 (1981)]), 2(LDA: Perdew-Wang 92 functional with Slater Exchange [PRB. 45, 13244 (1992)]), 3(LDA: Vosko, Wilk & Nusair with Slater Exchange[Can. J. Phys. 58, 1200 (1980)]), 4(GGA: Perdew-Burke-Ernzerhof functional [PRL. 77, 3865 (1996)]).

  Possible values: An integer n such that $1 \leq n \leq 4$

- *Parameter name:* `PSEUDOPOTENTIAL CALCULATION`

  Default: true

  Description: [Standard] Boolean Parameter specifying whether pseudopotential DFT calculation needs to be performed. For all-electron DFT calculation set to false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `PSEUDOPOTENTIAL FILE NAMES LIST`

  Default:

  Description: [Standard] Pseudopotential file. This file contains the list of pseudopotential file names in UPF format corresponding to the atoms involved in the calculations. UPF version greater than 2.0 and norm-conserving pseudopotentials in UPF format are only accepted. File format (example for two atoms Mg(z=12), Al(z=13)): 12 filename1.upf(row1), 13 filename2.upf (row2)

  Possible values: Any string

- *Parameter name:* `PSEUDO TESTS FLAG`

  Default: false

  Description: [Developer] Boolean parameter specifying the explicit path of pseudopotential upf format files used for ctests

  Possible values: A boolean value (true or false)

- *Parameter name:* `SPIN POLARIZATION`

  Default: 0

  Description: [Standard] Spin polarization: 0 for no spin polarization and 1 for collinear spin polarization calculation. Default option is 0.

  Possible values: An integer n such that $0 \leq n \leq 1$

- *Parameter name:* `START MAGNETIZATION`

  Default: 0.0

  Description: [Standard] Magnetization to start with (must be between -0.5 and +0.5). Corresponding magnetization per unit cell will be (2 x START MAGNETIZATION x Ne) a.u. , where Ne is the number of electrons in the unit cell

  Possible values: A floating point number v such that $-0.5 \leq v \leq 0.5$

A.7 Parameters in section `Finite element mesh parameters`

- *Parameter name:* `MESH FILE`

  Default:

  Description: [Developer] External mesh file path. If nothing is given auto mesh generation is performed. The option is only for testing purposes.

  Possible values: Any string

- *Parameter name:* `POLYNOMIAL ORDER`

  Default: 4

  Description: [Standard] The degree of the finite-element interpolating polynomial. Default value is 4. POLYNOMIAL ORDER= 4 or 5 is usually a good choice for most pseudopotential as well as all-electron problems.

  Possible values: An integer n such that $1 \leq n \leq 12$

A.8 Parameters in section `Finite element mesh parameters/Auto mesh generation parameters`

- *Parameter name:* `ATOM BALL RADIUS`

  Default: 2.0

  Description: [Developer] Radius of ball enclosing every atom inside which the mesh size is set close to MESH SIZE AROUND ATOM. The default value of 2.0 is good enough for most cases. On rare cases, where the nonlocal pseudopotential projectors have a compact supportbeyond 2.0, a slightly larger ATOM BALL RADIUS between 2.0 to 2.5 may be required. Standard users do not need to tune this parameter. Units: a.u.

  Possible values: A floating point number v such that $0 \leq v \leq 3$

- *Parameter name:* `BASE MESH SIZE`

  Default: 0.0

  Description: [Developer] Mesh size of the base mesh on which refinement is performed. For the default value of 0.0, a heuristically determined base mesh size is used, which is good enough for most cases. Standard users do not need to tune this parameter. Units: a.u.

  Possible values: A floating point number v such that $0 \leq v \leq 20$

- *Parameter name:* `MESH SIZE AROUND ATOM`

  Default: 1.0

  Description: [Standard] Mesh size in a ball of radius ATOM BALL RADIUS around every atom. For pseudopotential calculations, a value between 0.5 to 1.0 is usually a good choice. For all-electron calculations, a value between 0.1 to 0.3 would be a good starting choice. MESH SIZE AROUND ATOM is the only parameter standard users need to tune to achieve the desired accuracy in their results with respect to the mesh refinement. Units: a.u.

  Possible values: A floating point number v such that $0.0001 \leq v \leq 10$

- *Parameter name:* `MESH SIZE AT ATOM`

  Default: 0.0

  Description: [Developer] Mesh size of the finite elements in the immediate vicinity of the atom. For the default value of 0.0, a heuristically determined MESH SIZE AT ATOM is used, which is good enough for most cases. Standard users do not need to tune this parameter. Units: a.u.

  Possible values: A floating point number v such that $0 \leq v \leq 10$

A.9 Parameters in section `Geometry`

- *Parameter name:* `ATOMIC COORDINATES FILE`

  Default:

  Description: [Standard] Atomic-coordinates input file name. For fully non-periodic domain give cartesian coordinates of the atoms (in a.u) with respect to origin at the center of the domain. For periodic and semi-periodic give fractional coordinates of atoms. File format (example for two atoms): Atom1-atomic-charge Atom1-valence-charge x1 y1 z1 (row1), Atom2-atomic-charge Atom2-valence-charge x2 y2 z2 (row2). The number of rows must be equal to NATOMS, and number of unique atoms must be equal to NATOM TYPES.

  Possible values: Any string

- *Parameter name:* `DOMAIN VECTORS FILE`

  Default:

  Description: [Standard] Domain vectors input file name. Domain vectors describe the edges of the 3D parallelepiped computational domain. File format: v1x v1y v1z (row1), v2x v2y v2z (row2), v3x v3y v3z (row3). Units: a.u. CAUTION: please ensure that the domain vectors form a right-handed coordinate system i.e. dotProduct(crossProduct(v1,v2),v3)>0. Domain vectors are the typical lattice vectors in a fully periodic calculation.

  Possible values: Any string

- *Parameter name:* `NATOMS`

  Default: 0

  Description: [Standard] Total number of atoms. This parameter requires a mandatory non-zero input which is equal to the number of rows in the file passed to ATOMIC COORDINATES FILE.

  Possible values: An integer n such that $0 \leq n \leq 2147483647$

- *Parameter name:* `NATOM TYPES`

  Default: 0

  Description: [Standard] Total number of atom types. This parameter requires a mandatory non-zero input which is equal to the number of unique atom types in the file passed to ATOMIC COORDINATES FILE.

  Possible values: An integer n such that $0 \leq n \leq 2147483647$

A.10 Parameters in section `Geometry/Optimization`

- *Parameter name:* `CELL CONSTRAINT TYPE`

  Default: 12

  Description: [Standard] Cell relaxation constraint type, 1(isotropic shape-fixed volume optimization), 2(volume-fixed shape optimization), 3(relax only cell component v1x), 4(relax only cell component v2x), 5(relax only cell component v3x), 6(relax only cell components v2x and v3x), 7(relax only cell components v1x and v3x), 8(relax only cell components v1x and v2x), 9(volume optimization- relax only v1x, v2x and v3x), 10(2D- relax only x and y components relaxed), 11(2D- relax only x and y shape components- inplane area fixed), 12(relax all cell components), 13 automatically decides the constraints based on boundary conditions. CAUTION: A majority of these options only make sense in an orthorhombic cell geometry.

  Possible values: An integer n such that $1 \leq n \leq 13$

- *Parameter name:* `CELL OPT`

  Default: false

  Description: [Standard] Boolean parameter specifying if cell stress is to be relaxed

  Possible values: A boolean value (true or false)

- *Parameter name:* `CELL STRESS`

  Default: false

  Description: [Standard] Boolean parameter specifying if cell stress is to be computed. Automatically set to true if CELL OPT is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `FORCE TOL`

  Default: 1e-4

  Description: [Standard] Sets the tolerance of the maximum force (in a.u.) on an ion when forces are considered to be relaxed.

  Possible values: A floating point number v such that $0 \leq v \leq 1$

- *Parameter name:* `ION FORCE`

  Default: false

  Description: [Standard] Boolean parameter specifying if atomic forces are to be computed. Automatically set to true if ION OPT is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `ION OPT`

  Default: false

  Description: [Standard] Boolean parameter specifying if atomic forces are to be relaxed.

  Possible values: A boolean value (true or false)

- *Parameter name:* `ION RELAX FLAGS FILE`

  Default:

  Description: [Standard] File specifying the atomic position update permission flags. 1- update 0- no update. File format (example for two atoms with atom 1 fixed and atom 2 free): 0 0 0 (row1), 1 1 1 (row2).

  Possible values: Any string

- *Parameter name:* `NON SELF CONSISTENT FORCE`

  Default: false

  Description: [Developer] Boolean parameter specfiying whether to add the force terms arising due to the non self-consistency error. Currently non self-consistent force computation is still in developmental phase. The default option is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `STRESS TOL`

  Default: 1e-6

  Description: [Standard] Sets the tolerance of the cell stress (in a.u.) when cell stress is considered to be relaxed.

  Possible values: A floating point number v such that $0 \leq v \leq 1$

A.11 Parameters in section `Parallelization`

- *Parameter name:* `NPBAND`

  Default: 1

  Description: [Standard] Number of pools of MPI processors across which the work load of the bands is parallelised. NPKPT times NPBAND must be a divisor of total number of MPI tasks. Further, NPBAND must be less than or equal to NUMBER OF KOHN-SHAM WAVEFUNCTIONS.

  Possible values: An integer n such that $1 \leq n \leq 2147483647$

- *Parameter name:* `NPKPT`

  Default: 1

  Description: [Standard] Number of pools of MPI processors across which the work load of the irreducible k-points is parallelised. NPKPT times NPBAND must be a divisor of total number of MPI tasks. Further, NPKPT must be less than or equal to the number of irreducible k-points.

  Possible values: An integer n such that $1 \leq n \leq 2147483647$

A.12 Parameters in section `Poisson problem parameters`

- *Parameter name:* `MAXIMUM ITERATIONS`

  Default: 5000

  Description: [Developer] Maximum number of iterations to be allowed for Poisson problem convergence.

  Possible values: An integer n such that $0 \leq n \leq 20000$

- *Parameter name:* `TOLERANCE`

  Default: 1e-12

  Description: [Developer] Relative tolerance as stopping criterion for Poisson problem convergence.

  Possible values: A floating point number v such that $0 \leq v \leq 1$

A.13 Parameters in section `SCF parameters`

- *Parameter name:* `ANDERSON SCHEME MIXING HISTORY`

  Default: 10

  Description: [Standard] Number of SCF iteration history to be considered for mixing the electron-density. For metallic systems, typically a mixing history larger than the default value provides better scf convergence.

  Possible values: An integer n such that $1 \leq n \leq 1000$

- *Parameter name:* `ANDERSON SCHEME MIXING PARAMETER`

  Default: 0.5

  Description: [Standard] Mixing parameter to be used in Anderson scheme.

  Possible values: A floating point number v such that $0 \leq v \leq 1$

- *Parameter name:* `COMPUTE ENERGY EACH ITER`

  Default: true

  Description: [Developer] Boolean parameter specifying whether to compute the total energy at the end of every scf. Setting it to false can lead to some time savings.

  Possible values: A boolean value (true or false)

- *Parameter name:* `MAXIMUM ITERATIONS`

  Default: 50

  Description: [Standard] Maximum number of iterations to be allowed for SCF convergence

  Possible values: An integer n such that $1 \leq n \leq 1000$

- *Parameter name:* `STARTING WFC`

  Default: RANDOM

  Description: [Standard] Sets the type of the starting Kohn-Sham wavefunctions guess: Atomic(Superposition of single atom atomic orbitals. Wavefunctions for which atomic orbitals are not available, random wavefunctions are taken. Currently, atomic orbitals data is not available for all atoms.), Random(The starting guess for all wavefunctions are taken to be random). Default: RANDOM.

  Possible values: Any one of ATOMIC, RANDOM

- *Parameter name:* `TEMPERATURE`

  Default: 500.0

  Description: [Standard] Fermi-Dirac smearing temperature (in Kelvin).

  Possible values: A floating point number v such that $0 \leq v \leq$ MAX_DOUBLE

- *Parameter name:* `TOLERANCE`

  Default: 1e-06

  Description: [Standard] SCF iterations stopping tolerance in terms of L2 norm of the electron-density difference between two successive iterations. CAUTION: A tolerance close to 1e-7 or lower can detoriate the SCF convergence due to the round-off errors.

  Possible values: A floating point number v such that $1e-12 \leq v \leq 1$

A.14 Parameters in section `SCF parameters/Eigen-solver parameters`

- *Parameter name:* `BATCH GEMM`

  Default: true

  Description: [Developer] Boolean parameter specifying whether to use gemm batch blas routines to perform matrix-matrix multiplication operations with groups of matrices, processing a number of groups at once using threads instead of the standard serial route. CAUTION: gemm batch blas routines will only be activated if the CHEBYSHEV FILTER BLOCK SIZE is less than 1000, and intel mkl blas library linked with the dealii installation. Default option is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `CHEBYSHEV FILTER BLOCK SIZE`

  Default: 400

  Description: [Developer] The maximum number of wavefunctions which are handled by one call to the Chebyshev filter. This is useful for optimization purposes. The optimum value is dependent on the computing architecture.

  Possible values: An integer n such that $1 \leq n \leq 2147483647$

- *Parameter name:* `CHEBYSHEV FILTER TOLERANCE`

  Default: 1e-02

  Description: [Developer] Parameter specifying the tolerance to which eigenvectors need to computed using chebyshev filtering approach.

  Possible values: A floating point number v such that $1e-10 \leq v \leq$ MAX_DOUBLE

- *Parameter name:* `CHEBYSHEV POLYNOMIAL DEGREE`

  Default: 0

  Description: [Developer] Chebyshev polynomial degree to be employed for the Chebyshev filtering subspace iteration procedure to dampen the unwanted spectrum of the Kohn-Sham Hamiltonian. If set to 0, a default value depending on the upper bound of the eigen-spectrum is used.

  Possible values: An integer n such that $0 \leq n \leq 2000$

- *Parameter name:* `ENABLE SWITCH TO GS`

  Default: true

  Description: [Developer] Controls automatic switching to Gram-Schimdt orthogonalization if Lowden Orthogonalization or Pseudo-Gram-Schimdt orthogonalization are unstable. Default option is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `LOWER BOUND UNWANTED FRAC UPPER`

  Default: 0

  Description: [Developer] The value of the fraction of the upper bound of the unwanted spectrum, the lower bound of the unwanted spectrum will be set. Default value is 0.

  Possible values: A floating point number v such that $0 \leq v \leq 1$

- *Parameter name:* `LOWER BOUND WANTED SPECTRUM`

  Default: -10.0

  Description: [Developer] The lower bound of the wanted eigen spectrum. It is only used for the first iteration of the Chebyshev filtered subspace iteration procedure. A rough estimate based on single atom eigen values can be used here. Default value is good enough for most problems.

  Possible values: A floating point number v such that $-\text{MAX\_DOUBLE} \leq v \leq \text{MAX\_DOUBLE}$

- *Parameter name:* `NUMBER OF KOHN-SHAM WAVEFUNCTIONS`

  Default: 10

  Description: [Standard] Number of Kohn-Sham wavefunctions to be computed. For insulators use N/2+(10-20) and for metals use 20 percent more than N/2 (atleast 10 more). N is the total number of electrons. For spin-polarized calculations this parameter denotes the number of Kohn-Sham wavefunctions to be computed for each spin.

  Possible values: An integer n such that $0 \leq n \leq 2147483647$

- *Parameter name:* `ORTHOGONALIZATION TYPE`

  Default: PGS

  Description: [Standard] Parameter specifying the type of orthogonalization to be used: GS(Gram-Schmidt Orthogonalization using SLEPc library), LW(Lowden Orthogonalization implemented using LAPACK library, extension to use ScaLAPACK library not implemented yet), PGS(Pseudo-Gram-Schmidt Orthogonalization: if dealii library is compiled with ScaLAPACK and if you are using the real executable, parallel ScaLAPACK functions are used, otherwise serial LAPACK functions are used.) PGS is the default option.

  Possible values: Any one of GS, LW, PGS

- *Parameter name:* `ORTHO RR WFC BLOCK SIZE`

  Default: 200

Description: [Developer] This block size is used for memory optimization purposes in the orthogonalization and Rayleigh-Ritz steps. This optimization is only activated if dealii library is compiled with ScaLAPACK. Default value is 200.

Possible values: An integer n such that $1 \leq n \leq 2147483647$

- *Parameter name:* `SCALAPACKPROCS`

  Default: 0

  Description: [Developer] Uses a processor grid of SCALAPACKPROCS times SCALAPACKPROCS for parallel distribution of the subspace projected matrix in the Rayleigh-Ritz step and the overlap matrix in the Pseudo-Gram-Schmidt step. Default value is 0 for which a thumb rule is used (see http://netlib.org/scalapack/slug/node106.html#SECTION045110000000000000000). This parameter is only used if dealii library is compiled with ScaLAPACK.

  Possible values: An integer n such that $0 \leq n \leq 300$

- *Parameter name:* `SUBSPACE ROT DOFS BLOCK SIZE`

  Default: 2000

  Description: [Developer] This block size is used for memory optimization purposes in subspace rotation step in Pseudo-Gram-Schmidt orthogonalization and Rayleigh-Ritz steps. This optimization is only activated if dealii library is compiled with ScaLAPACK. Default value is 2000.

  Possible values: An integer n such that $1 \leq n \leq 2147483647$

Index of run-time parameters with section names

The following is a listing of all run-time parameters, sorted by the section in which they appear.