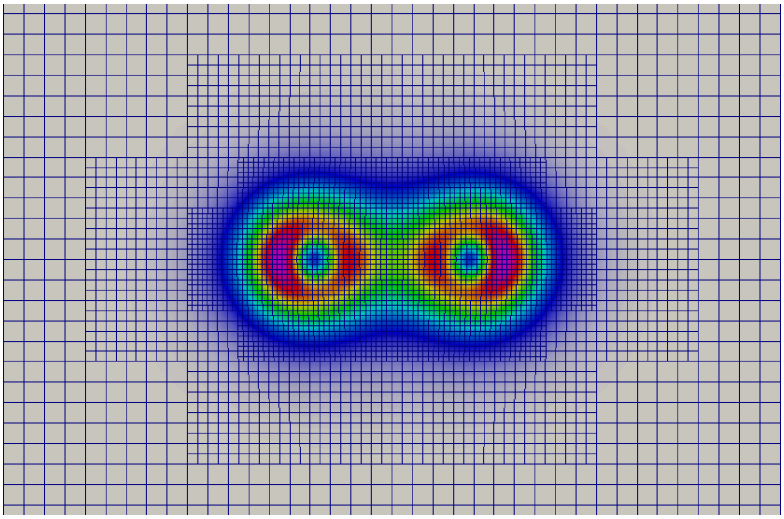
DFT-FE

Density Functional Theory calculations with Finite-Elements



User Manual

Version 0.5.0

(generated August 7, 2018)

Sambit Das
Vikram Gavini
Phani Motamarri

with contributions by:
Krishnendu Ghosh

website of dftfe

Copyright (c) 2017-2018 The Regents of the University of Michigan and DFT-FE authors.

Contents

1 Introduction

DFT-FE is a C++ code for materials modeling from first principles using Kohn-Sham density functional theory. It is based on adaptive finite-element discretization that handles all-electron and pseudopotential calculations in the same framework, and incorporates scalable and efficient solvers for the solution of the Kohn-Sham equations. Importantly, DFT-FE can handle general geometries and boundary conditions, including periodic, semi-periodic and non-periodic systems. 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 Developers with significant contributions to core functionalities and code architecture in the past who are no longer active principal developers, are listed under principal developers emeriti.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 to major parts of the DFT-FE code or sent important fixes and 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).

Principal 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).
- Shukan Parekh (University of Michigan, USA).

Mentors

- Vikram Gavini (University of Michigan, USA).

1.2 Acknowledgments

The development of DFT-FE open source code relating to pseudopotential calculations has been funded in part by the Department of Energy PRISMS Software Center at University of Michigan, and the Toyota Research Institute. The development of DFT-FE open source code relating to all-electron calculations has been funded by the Department of Energy Basic Energy Sciences. The methods and algorithms that have been implemented in DFT-FE are outputs from research activities over many years that have been supported by the Army Research Office, Air Force Office of Scientific Research, Department of Energy Basic Energy Sciences, National Science Foundation and Toyota Research Institute.

1.3 Referencing DFT-FE

Please refer to referencing DFT-FE to properly cite the use of DFT-FE in your scientific work.

2 Useful background information

We refer to the following articles for a background of the methods and algorithms implemented in DFT-FE.

1. P. Motamarri, M.R. Nowak, K. Leiter , J. Knap, V. Gavini, Higher-order adaptive finite-element methods for Kohn-Sham density functional theory, *J. Comput. Phys.* 253, 308-343 (2013).

2. P. Motamarri, V. Gavini, Subquadratic-scaling subspace projection method for large-scale Kohn-Sham DFT calculations using spectral finite-element discretization, *Phys. Rev. B* 90, 115127 (2014).

3. P. Motamarri, V. Gavini, V., Configurational forces in electronic structure calculations using Kohn-Sham density functional theory, *Phys. Rev. B* 97 165132 (2018).

In addition, below are some useful references on finite element method and some online resources that provide a background of finite elements and their application to the solution of partial differential equations.

1. T.J.R. Hughes, The finite element method: linear static and dynamic finite element analysis, Dover Publication, 2000.

2. K.-J. Bathe, Finite element procedures, Klaus-Jürgen Bathe, 2014.

3. The finite element method for problems in physics, online course by Krishna Garikipati. Link

4. Online lectures on “Finite element methods in scientific computing" by Wolfgang Bangerth. Link

3 Installation

All the underlying installation instructions assume a Linux operating system. We assume standard tools and libraries like CMake, compilers- (C, C++ and Fortran), and MPI libraries are 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 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 available on your high-performance computer. **Our experience shows that Intel compilers provide the best performance**. Furthermore, for 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 are set. 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) might already be 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 ALGLIB, Libxc, spglib, and Libxml2

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

   ```
   $ icpc -c -fPIC -O3 *.cpp
   $ icpc *.o -shared -o libAlglib.so
   ```

2. **Libxc**: Used by DFT-FE for exchange-correlation functionals. Download the current release from `http://www.tddft.org/programs/libxc/download/`, and do

   ```
   $ ./configure --prefix=libxc_install_dir_path
                 CC=c_compiler CXX=c++_compiler FC=fortran_compiler
         CFLAGS=-fPIC FCFLAGS=-fPIC

   $ make
   $ make install
   ```

   Do not forget to replace `libxc_install_dir_path` by some appropriate path on your file system and make sure that you have write permissions. Also replace `c_compiler, c++_compiler` and `fortran_compiler` with compilers on your system.

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 might be already installed in the high-performance computer you are working with. It is usually installed in the default locations like `/usr/lib64` (library path which contains `.so` file for Libxml2, something like `libxml2.so.2`) and `/usr/include/libxml2` (include path).

Libxml2 can also be installed by doing (Do not use these instructions if you have already have Libxml2 on your system)

```
$ git clone git://git.gnome.org/libxml2
$ ./autogen.sh --prefix=Libxml_install_dir_path
$ make
$ make install
```

There might be errors complaining that it can not create regular file libxml2.py in /usr/lib/python2.7/site-packages, but that should not matter.

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

1. **p4est**: This library is used by deal.II to create and distribute finite-element meshes across multiple processors. Download the current release tarball of p4est from http://www.p4est.org/. Also download the script from https://github.com/dftfeDevelopers/dftfe/raw/manual/p4est-setup.sh if using Intel compilers, or from https://dealii.org/developer/external-libs/p4est.html if using GCC compilers. Use the script to automatically compile and install a debug and optimized version of p4est by doing

   ```
   $ chmod u+x p4est-setup.sh
   $ ./p4est-setup.sh p4est-x-y-z.tar.gz p4est_install_dir_path
   ```

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 both the installation variants must have 64-bit indices and optimized mode enabled during the installation. To install PETSc, first download the current release tarball from https://www.mcs.anl.gov/petsc/download/index.html, unpack it, and follow the installation instructions in https://www.mcs.anl.gov/petsc/documentation/installation.html.

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

   ```
   $ ./configure --prefix=petscReal_install_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=prompted by PETSc
          PETSC_ARCH=prompted by PETSc

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

   For the complex installation variant, unpack a fresh PETSc directory from the tarball and repeat the above steps with the only changes being adding `--with-scalar-type=complex` and `--with-clanguage=cxx` to the configuration step (`./configure`) as well as providing a new installation path to `--prefix`.

   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. Also

make sure to follow compiling recommendations for the high-performance computer you are compiling on. 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=`". It can be 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"
```

3. **SLEPc**: The SLEPc library is built on top of PETSc, and it is used in DFT-FE for Gram-Schmidt Orthogonalization. To install SLEPc, first download the current 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`.

   For your reference you provide here an example installation of SLEPc for real scalar type

```
$ export PETSC_DIR=petscReal_install_dir_path
$ unset PETSC_ARCH
$ cd downloaded_slepc_dir
$ ./configure --prefix=slepcReal_install_dir_path
$ make
$ make install
```

4. **ScaLAPACK**: ScaLAPACK library is used by DFT-FE via deal.II for its parallel linear algebra routines involving dense matrices. ScaLAPACK is already installed in most high-performance computers. For example, in case of Intel MKL, linking to pre-installed ScaLAPACK 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. It is important to note that the second line above points to the BLACS library, which ScaLAPACK requires to be linked with, and the choice of the BLACS library depends upon the MPI library one is using. For instance, 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
```

   Installing ScaLAPACK from scratch
   Do not use these instructions if you already have pre-installed ScaLAPACK libraries on your high-performance computer. Download the current 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` for using the ScaLAPACK library. 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.

3.1.3 Instructions for deal.II

Assuming the above dependencies (p4est, PETSc, SLEPc, and ScaLAPACK) are installed, we now briefly discuss the steps to compile and install deal.II library linked with the above dependencies. You need to install two variants of the deal.II library– one variant linked with real scalar type PETSc and SLEPc installations, and the other variant linked with complex scalar type PETSc and SLEPc installations.

1. Obtain the development version of deal.II library via

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

2. In addition to requiring C, C++ and Fortran compilers, MPI library, and CMake, deal.II additionaly requires BOOST library. If not found externally, cmake will resort to the bundled BOOST that comes along with deal.II. Based on our experience, we recommend to use the bundled boost (enforced by unsetting/unloading external BOOST library environment paths) to avoid compilation issues.

3. ```
   $ mkdir buildReal
   $ cd buildReal
   $ cmake -DCMAKE_INSTALL_PREFIX=dealii_petscReal_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
   -DDEAL_II_WITH_P4EST=ON -DP4EST_DIR=p4est_install_dir_path
   -DDEAL_II_WITH_PETSC=ON -DPETSC_DIR=petscReal_install_dir_path
   -DDEAL_II_WITH_SLEPC=ON -DSLEPC_DIR=slepcReal_install_dir_path
   ```

   and LAPACK and ScaLAPACK link options (see below for example).

4. Repeat above step for installing deal.II linked with complex scalar type PETSc and SLEPc installations.

For more information about installing deal.II library refer to https://dealii.org/developer/readme.html. We also provide here an example of deal.II installation, which we did on a high-performance computer (STAMPEDE2) using Intel compilers and Intel MPI library (CXX flags used below are specific to the architecture)

```
$ 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
  -DDEAL_II_WITH_P4EST=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.

Using AVX, AVX-512 instructions in deal.II:
deal.II compilation will automatically try to pick the available vector instructions on the sytem like SSE2, AVX and AVX-512, and generate the following output message during compilation

```
-- Performing Test DEAL_II_HAVE_SSE2
-- Performing Test DEAL_II_HAVE_SSE2 - Success/Failed
-- Performing Test DEAL_II_HAVE_AVX
-- Performing Test DEAL_II_HAVE_AVX - Success/Failed
-- Performing Test DEAL_II_HAVE_AVX512
-- Performing Test DEAL_II_HAVE_AVX512 - Success/Failed
-- Performing Test DEAL_II_HAVE_OPENMP_SIMD
-- Performing Test DEAL_II_HAVE_OPENMP_SIMD - Success/Failed
```

"Success", means deal.II was able to use the corresponding vector instructions, and "Failed" would mean otherwise. If deal.II is not able to pick an available vector instruction on your high-performance computer, please contact the deal.II developers at deal.II mailing lists and/or contact your high-performance computer support for guidance on how to use the correct compiler flags for AVX or AVX-512.

Ensure that deal.II picks up AVX-512, which is strongly recommended for obtaining maximum performance on the new Intel Xeon Phi (KNL) and Skylake processors, both of which support Intel AVX-512 instructions.

3.1.4 Important generic instructions for deal.II and its dependencies

- If using Intel MKL for BLAS-LAPACK library, it is **very important** to use Intel MKL Link Line Advisor to set correctly link with BLAS-LAPACK library as needed in installations of PETSc, ScaLAPACK, and deal.II. To exploit performance benefit from threads, we recommend (strongly recommended for the new Intel Xeon Phi (KNL) and Skylane processors) linking to threaded versions of Intel MKL libraries by using the options "threading layer" and "OpenMP library" in Intel MKL Link Line Advisor.
- Use `-fPIC` compiler flag for compilation of deal.II and its dependencies, to prevent linking errors during DFT-FE compilation.
- **CAUTION! It is highly recommended to compile deal.II and its dependencies (p4est, PETSc, SLEPc, and ScaLAPACK), with the same compilers, same BLAS-LAPACK libraries, and same MPI libraries. This prevents deal.II compilation issues, occurence of run time crashes, and DFT-FE performance degradation.**

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. Obtain the source code of the current release of DFT-FE with all current patches using Git:

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

   Do `git pull` in the `dftfe` directory any time to obtain new patches that have been added since your `git clone` or last `git pull`. If you are not familiar with Git, you may download the current release tarball from the Downloads page in our website, but downloading via Git is recommended to avail new patches seamlessly.

2. 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 using cmake. For your reference, a few example `setup.sh` scripts are provided in the `/helpers` folder. If you are using GCC compilers, please add `-fpermissive` to the compiler flags. Also if you have installed deal.II by linking with Intel MKL library, set `withIntelMkl=ON` in setup.sh , otherwise set it to `OFF`.

3. To compile DFT-FE in release mode (the fast version), set `optimizedFlag=1` in `setup.sh` and do

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

   If compilation is successful, a `/build` directory will be created with the following executables:

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

4. To compile DFT-FE in debug mode (much slower but useful for debugging), set `optimizedFlag=0` in `setup.sh` and do

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

   which will create the following debug mode executables:

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

4 Running DFT-FE

After compiling DFT-FE as described in Section 3, we have now two executables — `/build/release/real/main` and `/build/release/complex/main`. The `/build/release/real/main` executable, which uses real data-structures is sufficient for fully non-periodic problems. The executable can also be used for periodic and semi-periodic problems involving a Gamma point calculation. On the other hand the `/build/release/complex/main` executable, which uses complex data-structures is required for periodic and semi-periodic problems with multiple k point sampling for Brillouin zone integration. These executables are to be used as follows– for a serial run use

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

or, for a parallel run use

```
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. As seen in Section A, there are two types of parameters: **Global parameters** and **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 **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 **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 **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 comments— 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
```

Also the indentation used in the above examples is only for readability.

4.2 Demo examples

Now we will walk you through a few demo examples in the `/dftfe/demo/` folder. The demo examples do not cover all the input parameter options. To get full list of input parameters see Section A. All input parameters are also conveniently indexed at the end of this manual in Section A.14.

4.2.1 Example 1

Let us consider the first example given in the folder `/dftfe/demo/ex1`, where we compute the ground state of the Nitrogen molecule using pseudopotential DFT calculations employing fully non-periodic boundary conditions. There are two input parameter files– parameterFile_a.prm and parameterFile_b.prm. parameterFile_a.prm is computing the ground-state and forces of the Nitrogen molecule while parameterFile_b.prm additionally does atomic relaxation.

Below, we provide a step by step procedure on setting up the above input parameter files, doing a total energy and force convergence study with respect to finite-element mesh discretization, and finally doing the atomic relaxation of Nitrogen molecule.

1. The geometry of the Nitrogen molecule (N_2) system is set using input parameters under `Geometry` subsection

   ```
   subsection Geometry
     set NATOMS=2
     set NATOM TYPES=1
     set ATOMIC COORDINATES FILE      = coordinates.inp
     set DOMAIN VECTORS FILE = domainVectors.inp
   end
   ```

   where

   - `NATOMS` is the total number of atoms, and `NATOM TYPES` is the total number of atom types.
   - "domainVectors.inp" (any other file name can be used), given as input to `DOMAIN VECTORS FILE`, is the external input file which lists the three domain vectors (in a.u) describing the 3D parallelepiped computational domain. For the current example we take a cubodial domain with 80 a.u as the edge length. Accordingly, the "domainVectors.inp" file is formatted as

     ```
     80.0 0.0 0.0
     0.0 80.0 0.0
     0.0 0.0 80.0
     ```

     wheres each row corresponds to a domain vector. It is a requirement that the above vectors must form a right-handed coordinate system i.e. $(v1 \times v2) \cdot v3 > 0$.
   - "coordinates.inp" (any other file name can be used), given as input to `ATOMIC COORDINATES FILE`, is the name of an external input file present in the same workspace which lists the Cartesian coordinates of the atoms (in a.u.) with respect to origin at the center of the domain. For this example, "coordinates.inp" is described as

     ```
     7    5   -1.30000000E+00   0.00000000E+00   0.00000000E+00
     7    5    1.30000000E+00   0.00000000E+00   0.00000000E+00
     ```

     where each line corresponds to "atomic-charge valence-charge x y z". Since this is a pseudopotential calculation, the valence-charge must correspond to the pseudopotential input, which we disuss in the later steps.

     We require Cartesian coordinates for fully non-periodic simulation domain like above while fractional coordinates are mandatory for periodic and semi-periodic simulation domain.

2. Set the fully non-periodic boundary conditions for the problem using the subsection `Boundary conditions`

   ```
   subsection Boundary conditions
     set PERIODIC1                       = false
     set PERIODIC2                       = false
     set PERIODIC3                       = false
   end
   ```

   where `PERIODIC1/2/3` sets the periodicity along the first, second, and third domain vectors. We note that DFT-FE allows for arbitrary boundary conditions.

3. Set the required DFT functional input parameters for pseudopotential calculation

```
subsection DFT functional parameters
  set EXCHANGE CORRELATION TYPE   = 4
  set PSEUDOPOTENTIAL CALCULATION = true
  set PSEUDOPOTENTIAL FILE NAMES LIST = pseudo.inp
end
```

where

- The choice of "4" for `EXCHANGE CORRELATION TYPE` corresponds to "GGA: Perdew-Burke-Ernzerhof functional [PRL. 77, 3865 (1996)]" functional.
- "pseudo.inp", given as input to `PSEUDOPOTENTIAL FILE NAMES LIST` is an external file (any other file name can be used) in the same workspace, which contains the list of pseudopotential file names in `UPF` format corresponding to the atom types involved in the calculations. The file is formatted as

  ```
  7 N_ONCV_PBE-1.0.upf
  ```

  where "7" is the atomic number of Nitrogen, and `N_ONCV_PBE-1.0.upf` is the Optimized Norm-Conserving Vanderbilt pseudopotential (ONCV) file obtained from http://www.quantum-simulation.org/potentials/sg15_oncv/upf/. **Presently, we only support Norm-Conserving pseudopotential (Troullier-Martins, ONCV) files in `UPF` format (version 2.0 or greater)**.

4. Set the input parameters for Self-Consistent field iterative procedure.

```
subsection SCF parameters
  set ANDERSON SCHEME MIXING HISTORY   = 70
  set ANDERSON SCHEME MIXING PARAMETER = 0.5
  set MAXIMUM ITERATIONS               = 40
  set TEMPERATURE                      = 500
  set TOLERANCE                        = 1e-5
  subsection Eigen-solver parameters
      set NUMBER OF KOHN-SHAM WAVEFUNCTIONS = 12
  end
end
```

where

- "70" set for `ANDERSON SCHEME MIXING HISTORY` is the number of SCF iteration history to be considered for mixing the electron- density using Anderson mixing sheme.
- "0.5" set for `ANDERSON SCHEME MIXING PARAMETER` is the mixing parameter to be used in Anderson scheme.
- "40" set for `MAXIMUM ITERATIONS` is the maximum number of iterations allowed in SCF iterative procedure.
- "500" set for `TEMPERATURE` is the Fermi-Dirac smearing temperature in Kelvin.
- "1e-5" set for `TOLERANCE` is the SCF stopping tolerance in terms of L2 norm of the electron-density difference between two successive iterations.
- "12" set for `NUMBER OF KOHN-SHAM WAVEFUNCTIONS` is the Number of Kohn-Sham wavefunctions to be computed in the Eigen solve (using Chebyshev subspace iteration solve) for every SCF iteration step. This parameter is set inside the subsection `Eigen-solver parameters`, which is nested within `SCF parameters`.

5. As we are also computing the force on the atoms in this example, update the `Geometry` subsection in the first step to

```
subsection Geometry
  set NATOMS=2
  set NATOM TYPES=1
  set ATOMIC COORDINATES FILE      = coordinates.inp
  set DOMAIN VECTORS FILE = domainVectors.inp
  subsection Optimization
    set ION FORCE = true
  end
end
```

   where the `ION FORCE` is set to true inside the nested subsection `Optimization`. This computes and prints the forces on the atoms at the end of the ground-state solve.

6. DFT-FE as mentioned before employs finite-element basis. These basis are piecewise polynomial functions. Hence DFT-FE allows for two approaches- (h and p refinement) for systematic converge of the ground-state energy and forces. h refinement is done primarily by tuning the input parameter– `MESH SIZE AROUND ATOM`, which controls the finite-element mesh size (grid spacing) around all atoms. p refinement is controlled by `POLYNOMIAL ORDER`, which is the degree of polynomial associated with the finite-element basis. In this example, we will first tune `MESH SIZE AROUND ATOM` for convergence while keeping `POLYNOMIAL ORDER` to 4 and then while keeping the `MESH SIZE AROUND ATOM` constant, we will increase the `POLYNOMIAL ORDER` to 5. `POLYNOMIAL ORDER` 4 or 5 is a good choice for most pseudopotential as well as all-electron problems. Let us take the following input parameters to start with

```
subsection Finite element mesh parameters
  set POLYNOMIAL ORDER=4
  subsection Auto mesh generation parameters
    set MESH SIZE AROUND ATOM  = 1.0
  end
end
```

   and now run the problem using the `/build/release/real/main` executable

```
mpirun -n 16 ../../build/release/real/main parameterFile_a.prm > outputMesh1 &
```

   From the "outputMesh1" file, you can obtain information on the number of degrees of freedom in the auto-generated finite-element mesh and the ground-state energy and forces.

   Repeat the above step twice, once with `MESH SIZE AROUND ATOM  = 0.5` and again with `MESH SIZE AROUND ATOM  = 0.25`, each time reducing the previous choice of `MESH SIZE AROUND ATOM` by half. Now run one more time with `POLYNOMIAL ORDER  = 5` while keeping `MESH SIZE AROUND ATOM  = 0.25`. We recommend to run this final simulation with around 64 MPI tasks for faster computational times. The ground-state energy per atom and force on the atomId 0 is tabulated in Table 1 for all the above cases. Upon comparing the errors in the energy and force with respect the most refined mesh (*Mesh No. 4*), we observe that for *Mesh No. 2* we obtain convergence in energy per atom to $\mathcal{O}(10^{-5})$ accuracy, and convergence in force to $\mathcal{O}(10^{-4})$ accuracy. For your reference, we have provided the output file for *Mesh No. 2* at `/demo/ex1/ex1_a.output`. DFT-FE also has the capability to write finite-element meshes with electron-density or wavefunction information to .vtu format which can be visualized using software like ParaView or Visit. As an example, Figure 1 shows the finite-element mesh and electron-density contours for *Mesh No. 2*, which are obtained via setting

```
set WRITE DENSITY=true
```

in the input parameter file, and visualizing the "densityOutput.vtu" file in ParaView.

Table 1: Nitrogen molecule ground-state energy and force convergence for demo example 1

Mesh No.	POLYNOMIAL ORDER	MESH SIZE AROUND ATOM	Degrees of freedom per atom	Energy per atom (Hartree)	Force on atomId 0 (Hartree/Bohr)
1	4	1.0	25332	-9.89828019	0.29152273
2	4	0.5	100984	-9.89917259	0.29401382
3	4	0.25	354624	-9.89921210	0.29417361
4	5	0.25	681717	-9.89921227	0.29417774

7. Finally we discuss how to set up the input parameter file for atomic relaxation. Use the same finite-element mesh input parameters as used for *Mesh No. 2*, and update the input parameters in `Geometry` subsection from the fifth step to

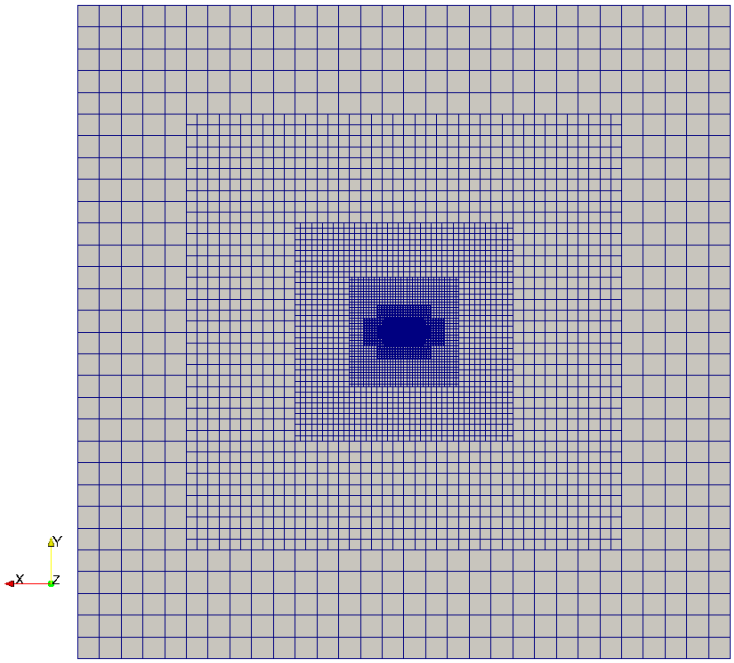
```
subsection Geometry
  set NATOMS=2
  set NATOM TYPES=1
  set ATOMIC COORDINATES FILE      = coordinates.inp
  set DOMAIN VECTORS FILE = domainVectors.inp
  subsection Optimization
    set ION OPT              = true
    set FORCE TOL            = 1e-4
    set ION RELAX FLAGS FILE = relaxationFlags.inp
  end
end
```

where

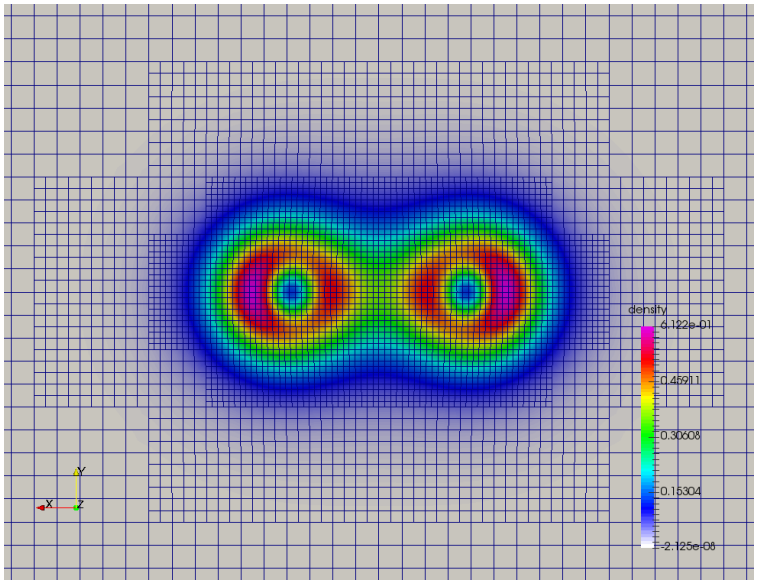
- `ION OPT` is set to true which enables atomic relaxation.
- "1e-4" for `FORCE TOL` sets the tolerance of the maximum force (in Hartree/Bohr) on an atom when atoms are considered to be relaxed.
- "relaxationFlags.inp", given as input to `ION RELAX FLAGS FILE` is an external file (any other file name can be used) in the same workspace, which specifies the permission flags (1– free to move, 0– fixed) for each coordinate axis and for all atoms. This file is described as

  ```
  1 0 0
  1 0 0
  ```

  which marks both the Nitrogen atoms to move freely along the x axis only. Now, run the atomic relaxation problem. From the output file, you should observe that the Nitrogen molecule geometry relaxed to an equilibrium bond length of 2.0715 Bohr after 14 geometry update steps. For your reference, we have provided an output file at `/demo/ex1/ex1_b.output`, which was run using 16 MPI tasks.



(a) Zoomed out



(b) Zoomed in with electron-density contours.

Figure 1: Finite-element mesh used in Nitrogen molecule pseudopotential DFT calculation (See 4.2.1).

4.2.2 Example 2

In the previous example, we discussed how to setup and run a fully non-periodic problem. Here we briefly discuss how to setup and run the fully periodic problem (FCC Aluminium unit cell) in the folder `/dftfe/demo/ex2`. There are two input parameter files– parameterFile_a.prm and parameterFile_b.prm. parameterFile_a.prm is for computing the ground-state and cell stress of the FCC Al unit cell, while parameterFile_b.prm additionally does cell stress relaxation. Important input parameters in the above parameter files, beyond what we discussed in the previous example are

1. "coordinates.inp" given as input to `ATOMIC COORDINATES FILE`, is the name of an external input file present in the same workspace which lists the fractional (reduced) coordinates of the atoms. For this example, "coordinates.inp" is described as

   ```
   13   3   0.00000000E+00   0.00000000E+00   0.00000000E+00
   13   3   0.00000000E+00   0.50000000E+00   0.50000000E+00
   13   3   0.50000000E+00   0.00000000E+00   0.50000000E+00
   13   3   0.50000000E+00   0.50000000E+00   0.00000000E+00
   ```

   where each line corresponds to "atomic-charge valence-charge fracx fracy fracz". **We require fractional coordinates for fully periodic or semi-periodic simulation domains while Cartesian coordinates are mandatory for fully non-periodic simulation domain.**

2. Set fully periodic boundary conditions

   ```
   subsection Boundary conditions
     set PERIODIC1                       = true
     set PERIODIC2                       = true
     set PERIODIC3                       = true
   end
   ```

3. Inside the `Optimization` subsection, nested within the `Geometry` subsection set

   ```
   set CELL STRESS = true
   ```

   for computing the ground state cell stress.

4. ```
   subsection Brillouin zone k point sampling options
     set USE TIME REVERSAL SYMMETRY = true
     subsection Monkhorst-Pack (MP) grid generation
       set SAMPLING POINTS 1 = 2
       set SAMPLING POINTS 2 = 2
       set SAMPLING POINTS 3 = 2
       set SAMPLING SHIFT 1  = 1
       set SAMPLING SHIFT 2  = 1
       set SAMPLING SHIFT 3  = 1
     end
   end
   ```

   where

   - `SAMPLING POINTS 1/2/3` sets the number of Monkhorst-Pack grid points to be used along reciprocal lattice vectors 1, 2, and 3.
   - Setting `SAMPLING SHIFT 1/2/3` to 1 enables fractional shifting to be used along reciprocal lattice vectors.
   ```

- Setting `USE TIME REVERSAL SYMMETRY` to true enables use of time reversal symmetry to reduce number of k points to be solved for. For this option to work `SAMPLING SHIFT 1/2/3` must be set to 1 as done above.

5. Set

```
subsection Parallelization
  set NPKPT=2
end
```

which parallelizes the work load of the irreducible k-points across two groups of MPI tasks.

6. The same strategy for convergence of the ground state energy and force discussed in the previous example is applied to the current example to get convergence in ground state energy and cell stress. The ground-state energy per atom and cell stress for finite-element meshes with increasing level of refinement is tabulated in Table 2. Upon comparing the errors in the energy and force with respect the most refined mesh (*Mesh No. 3*), we observe that for *Mesh No. 2* we have obtained convergence in energy per atom to $\mathcal{O}(10^{-5})$ accuracy, and convergence in cell stress to $\mathcal{O}(10^{-6})$ accuracy. The

Table 2: FCC Al ground-state energy and cell-stress convergence for demo example 2

Mesh No.	`POLYNOMIAL ORDER`	`MESH SIZE AROUND ATOM`	Degrees of freedom per atom	Energy per atom (Hartree)	Cell stress (Hartree/Bohr^3)
1	4	1.0	12084	-2.09082027	0.000030750
2	4	0.5	27172	-2.09323142	0.000034049
3	5	0.5	51239	-2.09323871	0.000037183

output file using the mesh parameters for *Mesh No.2* is provided at `/demo/ex2/ex2_a.output` (for parameterFile_a.prm).

7. For cell stress relaxation, use parameterFile_b.prm, where we set within the `Optimization` subsection nested under `Geometry`

```
set STRESS TOL            = 4e-6
set CELL OPT              = true
set CELL CONSTRAINT TYPE  = 1
```

where

- `CELL OPT` is set to true which enables cell stress relaxation.
- "4e-6" for `STRESS TOL` sets the tolerance of the cell stress (in a.u.) for cell stress relaxation.
- Choice of "1" for `CELL CONSTRAINT TYPE` enforces isotropic shape-fixed volume optimization constraint during cell stress relaxation.

For your reference, the output file for the cell stress relaxation is provided at `/demo/ex2/ex2_b.output`. From the output file, you should observe that you obtain a relaxed lattice constant of 7.5719 Bohr after three geometry updates.

5 Future extensions to DFT-FE

The future versions of DFT-FE will focus on the implementation of the following methodologies.

- *A-posteriori* mesh adaption techniques to construct optimal meshes with minimal user intervention.
- Improved electron-density mixing strategies to reduce the number of SCF iterations.
- Localization technique to reduce computational complexity of DFT-FE for both metallic and insulating systems in the same framework.
- Enrichment of the finite-element basis using single atom Kohn-Sham wavefunctions to reduce the computational cost of all-electron DFT calculations.
- GPU support for accelerating the numerical algorithms implemented in DFT-FE.

6 Finding answers to more questions

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

- For questions about DFT-FE, installation, bugs, etc., use the DFT-FE discussion forum.
- For latest news, updates, and release announcements about DFT-FE please send an email to dft-fe.admin@umich.edu, and we will add you to our announcement mailing list.
- 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/Advanced/Developer" label, which signifies whether an input parameter is a standard one, or an advanced level parameter, or a developer level one only meant for development purposes. The default values of the "Advanced" and "Developer" labelled parameters are good enough for almost all cases. However, in some cases user may need to use "Advanced" labelled parameters. 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 what is reproducible, 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. Ranges 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 DFT ground state 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. In the case of geometry optimization, the electron-density corresponding to the last ground-state solve is written. Default: false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `WRITE WFC`

  Default: false

  Description: [Standard] Writes DFT ground state 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. In the case of geometry optimization, the wavefunctions corresponding to the last ground-state solve are written. 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: [Advanced] The radius (in a.u) of the ball around an atom in which self-potential of the associated nuclear charge is solved. For the default value of 0.0, the radius value is automatically determined to accommodate 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 the use of 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 the use 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 rule to sample Brillouin zone. CAUTION: This option is recommended only 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 lattice 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 lattice 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 lattice 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 lattice 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 lattice 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 lattice 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 (do not create any checkpoint), 1 (create checkpoint for geometry optimization restart if either ION OPT or CELL OPT is set to true. Currently, checkpointing and restart framework does not work if both ION OPT and CELL OPT are set to true simultaneously- 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 CHK TYPE parameter before using this option. 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 2.0 or greater and norm-conserving pseudopotentials(ONCV and Troullier Martins) 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). Important Note: ONCV pseudopotentials data base in UPF format can be downloaded from http://www.quantum-simulation.org/potentials/sg15_oncv. Troullier-Martins pseudopotentials in UPF format can be downloaded from http://www.quantum-espresso.org/pseudopotentials/fhi-pp-from-abinit-web-site.

  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] Starting magnetization to be used for spin-polarized DFT calculations (must be between -0.5 and +0.5). Corresponding magnetization per simulation domain will be (2 x START MAGNETIZATION x Number of electrons) a.u.

  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: [Advanced] 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 support beyond 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 $0 \leq v \leq 20$

- *Parameter name:* `BASE MESH SIZE`

  Default: 0.0

  Description: [Advanced] 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: 0.8

  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. In most cases, MESH SIZE AROUND ATOM is the only parameter to be tuned to achieve the desired accuracy in energy and forces 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: [Advanced] 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 domain 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 are the vectors bounding the three 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 along domain vector component v1x), 4 (relax along domain vector component v2x), 5 (relax along domain vector component v3x), 6 (relax along domain vector components v2x and v3x), 7 (relax along domain vector components v1x and v3x), 8 (relax along domain vector components v1x and v2x), 9 (volume optimization- relax along domain vector components v1x, v2x and v3x), 10 (2D - relax along x and y components), 11(2D- relax only x and y components with inplane area fixed), 12(relax all domain vector 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 needs to be relaxed to achieve zero stress

  Possible values: A boolean value (true or false)

- *Parameter name:* `CELL STRESS`

  Default: false

  Description: [Standard] Boolean parameter specifying if cell stress needs 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 on the maximum force (in a.u.) on an atom when atoms 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 permission flags (1-free to move, 0-fixed) for the 3-coordinate directions and for all atoms. 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 specifying whether to include the force contributions arising out of non self-consistency in the Kohn-Sham ground-state calculation. Currently non self-consistent force computation is still in experimental 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.) during cell-relaxation.

  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 groups of MPI tasks 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 groups of MPI tasks 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: [Advanced] 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: [Advanced] 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 using Anderson mixing scheme. For metallic systems, 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: [Advanced] Boolean parameter specifying whether to compute the total energy at the end of every SCF. Setting it to false can lead to some computational time savings.

  Possible values: A boolean value (true or false)

- *Parameter name:* `MAXIMUM ITERATIONS`

  Default: 100

  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. Atom types 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 $1e-05 \leq v \leq$ MAX_DOUBLE

- *Parameter name:* `TOLERANCE`

  Default: 1e-06

  Description: [Standard] SCF iterations stopping tolerance in terms of L_2 norm of the electron-density difference between two successive iterations. CAUTION: A tolerance close to 1e-7 or lower can deteriorate the SCF convergence due to the round-off error accumulation.

  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: [Advanced] 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 only if intel mkl blas library is 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: [Advanced] Chebyshev filtering procedure involves the matrix-matrix multiplication where one matrix corresponds to the discretized Hamiltonian and the other matrix corresponds to the wavefunction matrix. The matrix-matrix multiplication is accomplished in a loop over the number of blocks of the wavefunction matrix to reduce the memory footprint of the code. This parameter specifies the block size of the wavefunction matrix to be used in the matrix-matrix multiplication. 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: [Advanced] Parameter specifying the accuracy of the occupied eigenvectors close to the Fermi-energy computed using Chebyshev filtering subspace iteration procedure. Default value is sufficient for most purposes

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

- *Parameter name:* `CHEBYSHEV POLYNOMIAL DEGREE`

  Default: 0

  Description: [Advanced] 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. See Phani Motamarri et.al., J. Comp. Phys. 253, 308-343 (2013).

  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 spin-polarized calculations, this parameter denotes the number of Kohn-Sham wavefunctions to be computed for each spin. A recommended value for this parameter is to set it to N/2+Nb where N is the number of electrons. Use Nb to be 10-20 percent of N/2 for insulators and for metals use Nb to be 20 percent of N/2. If 10-20 percent of N/2 is less than 10 wavefunctions, set Nb to be atleast 10.

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

- *Parameter name:* `ORTHOGONALIZATION TYPE`

  Default: Auto

  Description: [Advanced] Parameter specifying the type of orthogonalization to be used: GS(Gram-Schmidt Orthogonalization using SLEPc library), LW(Lowden Orthogonalization implemented using LAPACK/BLAS routines, 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.) Auto is the default option, which chooses GS for all-electron case and PGS for pseudopotential case.

  Possible values: Any one of GS, LW, PGS, Auto

- *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: [Advanced] 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). 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:* `SPECTRUM SPLIT CORE EIGENSTATES`

  Default: 0

  Description: [Advanced] Number of lowest Kohn-Sham eigenstates which should not be included in the Rayleigh-Ritz projection step. In other words, only the higher eigenstates (Number of Kohn-Sham wavefunctions minus the specified core eigenstates) are used to compute projected Hamiltonian and subsequently diagonalization is done on the projected Hamiltonian corresponding to the higher eigenstates. This value is usually chosen to be the sum of the number of core eigenstates for each atom type multiplied by number of atoms of that type. This setting is recommended for large systems (greater than 5000 electrons). Default value is 0 i.e., no core eigenstates are excluded from the Rayleigh-Ritz projection step.

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

- *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.