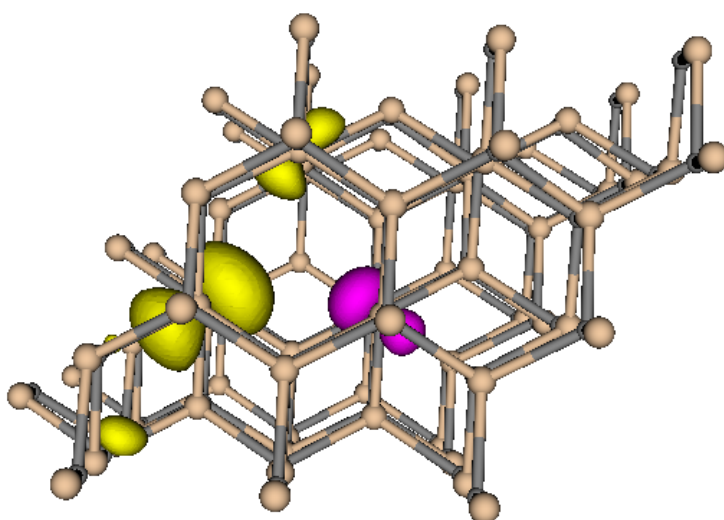


DFT-FE

Density Functional Theory calculations with Finite-Elements



User Manual
Version 0.5.0-pre
(generated August 1, 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	3
1.1	Authors	3
1.2	Acknowledgments	4
1.3	Referencing DFT-FE	4
2	Useful background information	4
3	Installation	4
3.1	Compiling and installing external libraries	5
3.1.1	Instructions for ALGLIB, Libxc, spglib, and Libxml2	5
3.1.2	Instructions for deal.II's dependencies– p4est, PETSc, SLEPc, and ScaLAPACK	6
3.1.3	Instructions for deal.II	8
3.1.4	Important generic instructions for deal.II and its dependencies	9
3.2	Obtaining and Compiling DFT-FE	10
4	Running DFT-FE	11
4.1	Structuring the input file	11
4.2	Demo examples	12
4.2.1	Example 1	12
5	Future extensions to DFT-FE	12
6	Finding answers to more questions	12
A	Run-time input parameters	13
A.1	Global parameters	13
A.2	Parameters in section Boundary conditions	13
A.3	Parameters in section Brillouin zone k point sampling options	14
A.4	Parameters in section Brillouin zone k point sampling options/Monkhorst-Pack (MP) grid generation	14
A.5	Parameters in section Checkpointing and Restart	15
A.6	Parameters in section DFT functional parameters	16
A.7	Parameters in section Finite element mesh parameters	17
A.8	Parameters in section Finite element mesh parameters/Auto mesh generation parameters	17
A.9	Parameters in section Geometry	18
A.10	Parameters in section Geometry/Optimization	18
A.11	Parameters in section Parallelization	20
A.12	Parameters in section Poisson problem parameters	20
A.13	Parameters in section SCF parameters	20
A.14	Parameters in section SCF parameters/Eigen-solver parameters	21
	Index of run-time parameters with section names	24

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_compiler_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
${MKLR00T}/lib/intel64/libmkl_intel_lp64.a
${MKLR00T}/lib/intel64/libmkl_intel_thread.a
${MKLR00T}/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](#))

```
${MKLR00T}/lib/intel64/libmkl_scalapack_lp64.so
${MKLR00T}/lib/intel64/libmkl_blacs_intelmpi_lp64.so
```

where \$MKLR00T 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

```
${MKLR00T}/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 additionally 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
-DMAKE_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="{MKLRROOT}/lib/intel64" -DLAPACK_FOUND=true
-DLAPACK_LIBRARIES="{MKLRROOT}/lib/intel64/libmkl_intel_lp64.so;
{MKLRROOT}/lib/intel64/libmkl_core.so;{MKLRROOT}/lib/intel64/libmkl_intel_thread.so"
```



```

-DLAPACK_LINKER_FLAGS="-liomp5 -lpthread -lm -ldl"
-DSCALAPACK_DIR="${MKLRROOT}/lib/intel64"
-DSCALAPACK_LIBRARIES="${MKLRROOT}/lib/intel64/libmkl_scalapack_lp64.so;
${MKLRROOT}/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](#). Note that in the above procedure one is installing the development version of deal.II library and this version is continuously updated by deal.II developers, which can sometimes lead to installation issues on certain compilers. If you face any issues during the installation procedure of deal.II development version as explained above, you may alternatively obtain the current release version of deal.II by downloading and unpacking the .tar.gz file from <https://www.dealii.org/download.html> and following the same procedure as above. If you still face installation issues, and/or if you have any questions about the deal.II installation, please contact the deal.II developers at [deal.II mailing lists](#).

Using AVX, AVX-512 instructions in deal.II:

deal.II compilation will automatically try to pick the available vector instructions on the system 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.

Ensuring that deal.II picks up AVX-512 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 Skylake 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 very important to ensure that deal.II and its dependencies (p4est, PETSc, SLEPc, and ScaLAPACK), are compiled with the same compilers, BLAS-LAPACK libraries (except for p4est), and MPI libraries to prevent deal.II compilation issues, occurrence 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 from the current release [tarball](#), but downloading via Git is recommended to avail new patches seamlessly.

Obtaining previous releases: (Skip this part if you are using the current release version)

```
(a) $ git clone https://github.com/dftfeDevelopers/dftfe.git
    $ cd dftfe
```

- (b) To get the list of all release tags (current and previous releases) do

```
$ git tag -l
```

- (c) Choose the desired release tag name and do

```
$ git checkout tags/<tag_name>
```

Alternatively, you could download the appropriate tarball from [github-releases](#).

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 above in Section 3, 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-structures 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, let's 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

Now we will walk you through a few demo examples in the `/dftfe/demo/` folder.

4.2.1 Example 1

Lets consider the first example given in the folder `/dftfe/demo/ex1`, where we compute the pseudopotential KSDFT groundstate of Nitrogen molecule. There are two input parameter files– `parameterFile_a.prm` and `parameterFile_b.prm`. `parameterFile_a.prm` is for computing the KSDFT ground-state of the Nitrogen molecule without atomic relaxation while `parameterFile_b.prm` involves atomic relaxation. Below we provide a walkthrough on setting up `parameterFile_a.prm` and `parameterFile_b.prm`

```
7 5 -1.50000000E+00 0.00000000E+00 0.00000000E+00
7 5 1.50000000E+00 0.00000000E+00 0.00000000E+00
```

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

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

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/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 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. 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- densityI_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 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(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 $(2 \times \text{START MAGNETIZATION} \times N_e)$ a.u., where N_e 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 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 $0 \leq v \leq 9$

- *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: 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: [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. $\text{dotProduct}(\text{crossProduct}(\mathbf{v1}, \mathbf{v2}), \mathbf{v3}) > 0$. 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 specifying 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 $1e-05 \leq v \leq \text{MAX_DOUBLE}$

- *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 deteriorate 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 be computed using chebyshev filtering approach.

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

- *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-Schmidt orthogonalization if Lowden Orthogonalization or Pseudo-Gram-Schmidt 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#SECTION04511000000000000000>). 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: [Standard] Number of lowest Kohn-Sham eigenstates which should not be included in the Rayleigh-Ritz projection step. In other words, only the higher eigenstates excluding the specified core eigenstates are used to compute projected Hamiltonian and subsequently diagonalization is done on this reduced Hamiltonian. 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. Recommended for large systems with greater than 5000 electrons. Default value is 0- 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.

Boundary conditions

PERIODIC1, [13](#)
PERIODIC2, [14](#)
PERIODIC3, [14](#)
SELF POTENTIAL RADIUS, [14](#)

Brillouin zone k point sampling options

kPOINT RULE FILE, [14](#)
Monkhorst-Pack (MP) grid generation
 SAMPLING POINTS 1, [14](#)
 SAMPLING POINTS 2, [15](#)
 SAMPLING POINTS 3, [15](#)
 SAMPLING SHIFT 1, [15](#)
 SAMPLING SHIFT 2, [15](#)
 SAMPLING SHIFT 3, [15](#)
USE GROUP SYMMETRY, [14](#)
USE TIME REVERSAL SYMMETRY, [14](#)

Checkpointing and Restart

CHK TYPE, [15](#)
RESTART FROM CHK, [15](#)

DFT functional parameters

EXCHANGE CORRELATION TYPE, [16](#)
PSEUDO TESTS FLAG, [16](#)
PSEUDOPOTENTIAL CALCULATION, [16](#)
PSEUDOPOTENTIAL FILE NAMES LIST,
 [16](#)
SPIN POLARIZATION, [16](#)
START MAGNETIZATION, [16](#)

Finite element mesh parameters

Auto mesh generation parameters
 ATOM BALL RADIUS, [17](#)
 BASE MESH SIZE, [17](#)
 MESH SIZE AROUND ATOM, [17](#)
 MESH SIZE AT ATOM, [17](#)
MESH FILE, [17](#)
POLYNOMIAL ORDER, [17](#)

Geometry

ATOMIC COORDINATES FILE, [18](#)
DOMAIN VECTORS FILE, [18](#)
NATOM TYPES, [18](#)
NATOMS, [18](#)

Optimization

CELL CONSTRAINT TYPE, [18](#)
CELL OPT, [19](#)
CELL STRESS, [19](#)

FORCE TOL, [19](#)
ION FORCE, [19](#)
ION OPT, [19](#)
ION RELAX FLAGS FILE, [19](#)
NON SELF CONSISTENT FORCE, [19](#)
STRESS TOL, [19](#)

Parallelization

NPBAND, [20](#)
NPKPT, [20](#)

Poisson problem parameters

MAXIMUM ITERATIONS, [20](#)
TOLERANCE, [20](#)

REPRODUCIBLE OUTPUT, [13](#)

SCF parameters

ANDERSON SCHEME MIXING
 HISTORY, [20](#)
ANDERSON SCHEME MIXING
 PARAMETER, [20](#)
COMPUTE ENERGY EACH ITER, [20](#)

Eigen-solver parameters

BATCH GEMM, [21](#)
CHEBYSHEV FILTER BLOCK SIZE, [21](#)
CHEBYSHEV FILTER TOLERANCE, [21](#)
CHEBYSHEV POLYNOMIAL DEGREE,
 [22](#)
ENABLE SWITCH TO GS, [22](#)
LOWER BOUND UNWANTED FRAC
 UPPER, [22](#)
LOWER BOUND WANTED
 SPECTRUM, [22](#)
NUMBER OF KOHN-SHAM
 WAVEFUNCTIONS, [22](#)
ORTHO RR WFC BLOCK SIZE, [22](#)
ORTHOGONALIZATION TYPE, [22](#)
SCALAPACKPROCS, [23](#)
SPECTRUM SPLIT CORE
 EIGENSTATES, [23](#)
SUBSPACE ROT DOFS BLOCK SIZE,
 [23](#)

MAXIMUM ITERATIONS, [21](#)
STARTING WFC, [21](#)
TEMPERATURE, [21](#)
TOLERANCE, [21](#)

VERBOSITY, [13](#)

WRITE DENSITY, [13](#)

WRITE WFC, [13](#)