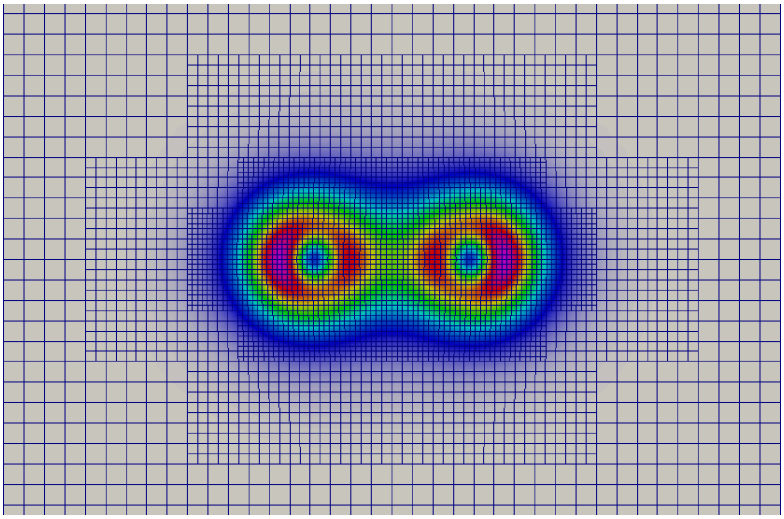
DFT-FE

Density Functional Theory calculations with Finite-Elements



User Manual

Version 1.0.0

(generated May 7, 2022)

Sambit Das
Vikram Gavini
Phani Motamarri

with contributions by:
Krishnendu Ghosh

website of dftfe

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 developed and maintained by the Computational Materials Physics group at University of Michigan, and the MATRIX lab, Indian Institute of Science. 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 Developers with significant contributions to core functionalities and code architecture in the past who are no longer active principal developers, are listed under principal developers emeriti.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 (Indian Institute of Science, Bangalore, India).

Principal developers emeriti

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

Contributors

- Sambit Das (University of Michigan, USA).
- Vishal Subramanian (University of Michigan, USA).
- Bikash Kanungo (University of Michigan, USA).
- Denis Davydov (University of Erlangen-Nuremberg, Germany).
- Krishnendu Ghosh (Intel Corp., USA).
- Phani Motamarri (Indian Institute of Science, Bangalore, India).
- Shiva Rudraraju (University of Wisconsin Madison, USA).
- Shukan Parekh (University of Michigan, USA).
- David Rogers (Oak Ridge National Laboratory, 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, S. Das, S. Rudraraju, K. Ghosh, D. Davydov, V. Gavini, DFT-FE–A massively parallel adaptive finite-element code for large-scale density functional theory calculations, *Comput. Phys. Commun.* 246, 106853 (2020).

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).

3. P. Motamarri, V. Gavini, 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), CUDA (in case of GPU architectures), MPI, and math(BLAS-LAPACK) 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 the above and compilation tools like CMake. For example, on one of the high-performance computers (UMICH Greatlakes) we develop and test the DFT-FE code, we can use the following commands to set the desired environment variables

```
$ module load cmake/3.18.2
$ module load gcc/8.2.0
$ module load openmpi/3.1.4
$ module load mkl/2018.0.4
$ module load cuda/11.0.2 (if installing for GPUs)
```

Note the above are shown only as an example. We strongly recommend using the latest stable version of compilers-(C, C++ and Fortran), CUDA, MPI and math libraries available on your high-performance computer. DFT-FE's installation requires also the following minimum versions of the above compilers and libraries:

- CMake 3.17.0
- GCC 8.2.0
- CUDA 11.0.2 (if installing for GPUs)

We currently do not support Intel compilers due to a compilation issue of the deal.II library. Please use GNU compilers only. Further, if version of CMake greater than 3.17.0 is not available on your machine please install latest version from here CMake or use pre-installed binaries most appropriate for your machine.

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, ScaLAPACK and BLAS-LAPACK are set. The other required external libraries, which are not interfaced via deal.II are ALGLIB, Libxc, spglib, Libxml2 and ELPA. DFT-FE also optionally links to PETSc, SLEPc (via deali.II) and to nccl (for GPU compilation). The optional dependencies of PETSc and SLEPc are only required for all-electron calculations using DFT-FE, which uses the more stable Gram-Schmidt orthogonalization routine instead of the default Cholesky-Gram-Schmidt orthognalization. For pseudopotential calculations, PETSc and SLEPc dependencies are not required as the default Cholesky-Gram-Schmidt orthogonalization is very robust. Below, we give brief installation instructions for each of the above libraries.

3.1.1 Instructions for dependencies: ALGLIB, Libxc, spglib, Libxml2, ScaLAPACK, ELPA, p4est and nccl (nccl is optional)

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 GCC compiler do

   ```
   $ g++ -c -fPIC -O2 *.cpp
   $ g++ *.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="-O2 -fPIC" FCFLAGS="-O2 -fPIC" CXXFLAGS="-O2 -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. We recommend using the ccmake gui interface for the installation and also use appropriate compiler for `CMAKE_C_COMPILER`.

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 https://gitlab.gnome.org/GNOME/libxml2.git
   $ ./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.

5. **ScaLAPACK**: ScaLAPACK library is used by DFT-FE via deal.II for its parallel linear algebra routines involving dense matrices, as well being a dependency for ELPA. **If Intel MKL math library is available, please skip this step, as the ScaLAPACK libraries therein can be used directly.** If Intel MKL math library is not available, Netlib ScaLAPACK http://www.netlib.org/scalapack/ needs to be installed using the instructions below. Download the current release version (2.1.0) from http://www.netlib.org/scalapack/#_software, and build a shared library (use `BUILD_SHARED_LIBS=ON`, `BUILD_STATIC_LIBS=OFF` and `BUILD_TESTING=OFF`) installation of ScaLAPACK using cmake. We recommend using the ccmake gui interface for the installation. Further, use the appropriate compilers for `CMAKE_C_COMPILER` and `CMAKE_FORTRAN_COMPILER`, and also use `-fPIC` flag for `CMAKE_C_FLAGS` and `CMAKE_Fortran_FLAGS`. 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 libraries (MKL, OpenBlas, ESSL etc.) during the cmake configuration.

6. **ELPA**: ELPA library is used by DFT-FE for its parallel linear algebra routines involving dense matrices. ELPA requires the ScaLAPACK library (see above) as a dependency. Download version elpa-2020.05.002 (this version has been thoroughly tested) from https://elpa.mpcdf.mpg.de/software/ and follow the installation instructions in there. Example of ELPA installation on UMICH Greatlakes supercomputer with GNU compiler, Open MPI library, and Intel MKL math library:

   ```
   $ cd elpaDir
   $ mkdir build
   $ cd build
   $ ../configure --enable-openmp FC=mpif90 CC=mpicc CXX=mpicxx
   FCFLAGS="-fopenmp -O2 -march=native" CFLAGS="-fopenmp -O2 -march=native"
   ```

```
CXXFLAGS="-fopenmp -O2 -march=native" --prefix=elpa_install_path
SCALAPACK_LDFLAGS=" -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64
-Wl,--no-as-needed -lmkl_intel_lp64
-lmkl_gnu_thread -lmkl_core -lmkl_blacs_openmpi_lp64 -lgomp -lpthread -lm -ldl"
SCALAPACK_FCFLAGS="-L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64
-Wl,--no-as-needed -lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_core
-lmkl_blacs_openmpi_lp64 -lgomp -lpthread -lm -ldl"
$ make -j 4
$ make install
```

The MKL paths and linker flags were obtained with the help of Intel MKL Link Line Advisor for GNU Fortran compiler (Note the usage of `-lmkl_gf_lp64` flag above).

If the machine of interest has NVIDIA GPUs and CUDA library, ELPA can take advantage of GPUs. For example on OLCF Summit GPU nodes, we use the following configure line

```
../configure FC=mpif90 CC=mpicc FCFLAGS="-O2
-fPIC -mcpu=power9 -mvsx -maltivec" CFLAGS="-O2
-fPIC -mcpu=power9 -mvsx -maltivec"
--enable-nvidia-gpu
--with-NVIDIA-GPU-compute-capability="sm_70"
--with-cuda-path="$OLCF_CUDA_ROOT"
--with-cuda-sdk-path="$OLCF_CUDA_ROOT"
--prefix=elpa_install_path
LDFLAGS="-L$netlib_scalapack_installation_path/lib -lscalapack
-L/$OLCF_ESSL_ROOT/lib64 -lessl
-L/$OLCF_NETLIB_LAPACK_ROOT/lib64 -llapack
-L/$OLCF_OPENBLAS_ROOT/lib -lopenblas"
--disable-sse --disable-sse-assembly
--disable-avx --disable-avx2 --disable-avx512
--enable-c-tests=no
```

Note the use of LDFLAGS instead of SCALAPACK_LDFLAGS and SCALAPACK_FCFLAGS, since we are using netlib ScaLAPACK instead of Intel MKL ScaLAPACK in the above. Also note use of

```
--disable-sse --disable-sse-assembly --disable-avx --disable-avx2 --disable-avx512
```

above. **Some or all of these options may be required for systems without Intel CPUs such as IBM Power and AMD Epyc processors depending on what they support.**

7. **p4est**: This library is used by deal.II to create and distribute finite-element meshes across multiple processors. Download the v2.2 tarball of p4est from http://www.p4est.org/. Next download the `p4est-setup.sh` script from https://raw.githubusercontent.com/dftfeDevelopers/dftfe/manual/p4est-setup.sh. 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
```

8. **nccl (optional)**: nccl is an optional dependency for DFT-FE for optimal GPU Direct MPI collective communication calls. This library is recommended for running very large system sizes (greater than 20,000 electrons) on GPUs using DFT-FE. Caution: nccl library requires appropriate hardware support for GPU Direct MPI communication and CUDA Aware MPI library. To install nccl, clone development version https://github.com/NVIDIA/nccl and follow installation instructions therein.

3.1.2 Instructions for deal.II

Assuming the above dependencies (p4est 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 customized version of deal.II library via

   ```
   $ git clone -b dealiiCustomizedCUDARelease 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 deal.II's bundled boost (enforced by unsetting/unloading external BOOST library environment paths) to avoid compilation issues. Further if installing on NVIDIA GPUs, deal.II requires CUDA (minimum version 11.0.2).

3. ```
   $ mkdir build
   $ cd build
   $ cmake -DCMAKE_INSTALL_PREFIX=dealii_install_dir_path
           otherCmakeOptions ../deal.II
   $ make -j 8
   $ make install
   ```

   "otherCmakeOptions" include the following options for CPU installation:

   ```
   -DCMAKE_C_COMPILER=c_compiler
   -DCMAKE_CXX_COMPILER=cxx_compiler
   -DCMAKE_Fortran_COMPILER=fortran_compiler
   -DMPI_C_COMPILER=mpi_c_compiler_wrapper
   -DMPI_CXX_COMPILER=mpi_cxx_compiler_wrapper
   -DMPI_Fortran_COMPILER=mpi_fortran_compiler_wrapper
   -DCMAKE_CXX_FLAGS=cxx_flags
   -DCMAKE_C_FLAGS=c_flags
   -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_LAPACK=ON
   -DLAPACK_DIR=lapack_dir_paths (both BLAS and LAPACK directory paths)
   -DLAPACK_FOUND=true
   -DLAPACK_LIBRARIES=lapack_lib_paths (both BLAS and LAPACK library paths)
   -DDEAL_II_WITH_SCALAPACK=ON
   -DSCALAPACK_DIR=scalapack_dir_path (only required if linking to netlib ScaLAPACK)
   -DSCALAPACK_LIBRARIES=scalapack_lib_path
   -DDEAL_II_WITH_TBB=OFF
   -DDEAL_II_WITH_TASKFLOW=OFF
   -DDEAL_II_COMPONENT_EXAMPLES=OFF
   ```

   and additional cmake options for GPU installation:

   ```
   -DDEAL_II_CUDA_FLAGS=gpu_arch_flag (eg. "-arch=sm_70")
   -DDEAL_II_WITH_CUDA=ON
   -DDEAL_II_MPI_WITH_CUDA_SUPPORT=ON
   ```
   ```

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 UMICH Greatlakes supercomputer with GNU compiler, Open MPI library, and Intel MKL math library

```
$ mkdir build
$ cd build
$ cmake -DCMAKE_C_COMPILER=gcc
-DCMAKE_CXX_COMPILER=g++
-DCMAKE_Fortran_COMPILER=gfortran
-DMPI_C_COMPILER=mpicc
-DMPI_CXX_COMPILER=mpicxx
-DMPI_Fortran_COMPILER=mpif90
-DCMAKE_CXX_FLAGS="-march=native"
-DCMAKE_C_FLAGS="-march=native"
-DDEAL_II_CXX_FLAGS_RELEASE="-O2"
-DDEAL_II_COMPONENT_EXAMPLES=OFF
-DDEAL_II_WITH_MPI=ON
-DDEAL_II_WITH_64BIT_INDICES=ON
-DDEAL_II_WITH_TBB=OFF
-DDEAL_II_WITH_TASKFLOW=OFF
-DDEAL_II_WITH_P4EST=ON
-DP4EST_DIR=p4est_install_path
-DDEAL_II_WITH_LAPACK=ON -DLAPACK_DIR="${MKLROOT}/lib/intel64"
-DLAPACK_FOUND=true
-DLAPACK_LIBRARIES="-L${MKLROOT}/lib/intel64
-Wl,--no-as-needed -lmkl_intel_lp64
-lmkl_gnu_thread -lmkl_core -lgomp -lpthread -lm
-ldl" -DLAPACK_INCLUDE_DIRS="-I${MKLROOT}/include"
-DDEAL_II_WITH_SCALAPACK=ON
-DSCALAPACK_LIBRARIES="-L${MKLROOT}/lib/intel64
-lmkl_scalapack_lp64 -Wl,--no-as-needed
-lmkl_intel_lp64 -lmkl_gnu_thread -lmkl_core
-lmkl_blacs_openmpi_lp64 -lgomp -lpthread -lm
-ldl"
-DCMAKE_INSTALL_PREFIX=dealii_install_path ../dealii
$ make -j 8
$ make install
```

The values for `-DLAPACK_DIR`,`-DLAPACK_LIBRARIES` and `-DLAPACK_LINKER_FLAGS` were obtained with the help of Intel MKL Link Line Advisor for GNU C++ compiler (cf. Fig. 1).

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
```

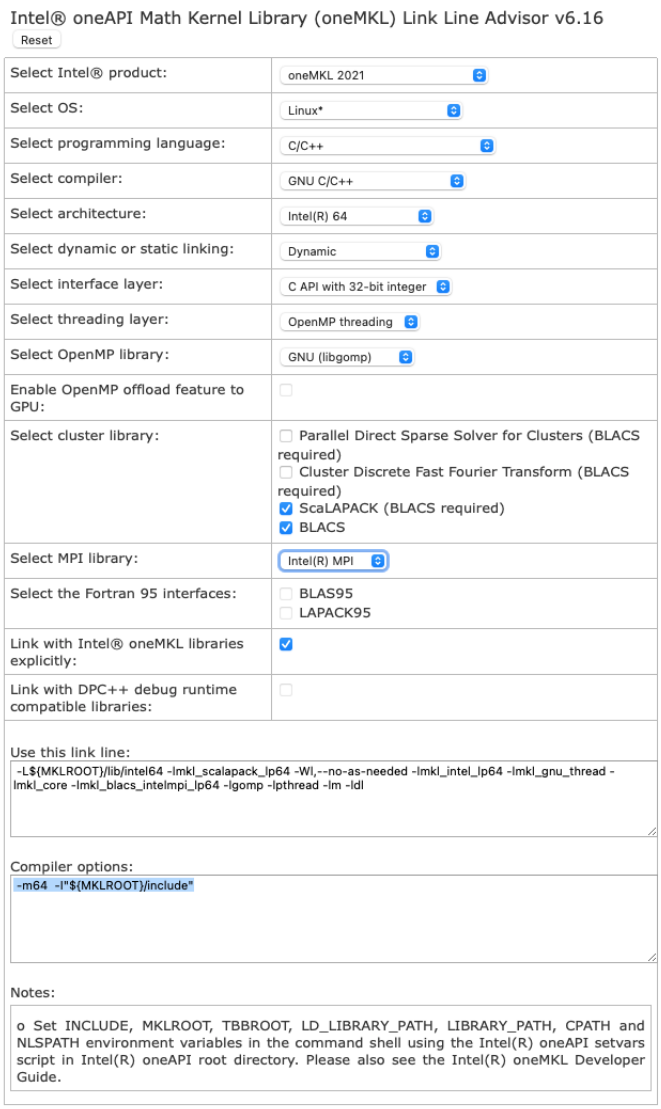


Figure 1: Example usage of Intel MKL line advisor. Use the options in the "link line" generated by the line advisor tool. Please note to change "intelmpi" in "-lmkl_blacs_intelmpi_lp64" to "openmpi" if using openmpi MPI library.

"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.

3.2 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, Libxml2, and ELPA), C++ compiler, and C++ compiler flags in `setupUser.sh`, which is a script to compile DFT-FE using cmake. nccl library can also be optionally provided in case of GPU compilation. If compiling only for CPUs, set the following to OFF

   ```
   withGPU=OFF
   withNCCL=OFF
   ```

   For GPU compilation, `withGPU` must be set to ON.

3. To compile DFT-FE, first create a build directory anywhere on your machine. Next from inside the build directory do

   ```
   $ bash $dftfe_source/setupUser.sh
   ```

   Please use the full directory path for `$dftfe_source` above. Also note that sometimes compilation on login node can crash due to insufficient memory. In those cases, we recommend using an interactive job if available on your computing cluster.

4. If compilation is successful, the following executables will be created:

   ```
   $dftfe_build_dir/release/real/dftfe
   $dftfe_build_dir/release/complex/dftfe
   ```

5. To compile DFT-FE in debug mode (much slower but useful for debugging), set `build_type=Debug` in `setupUser.sh` and do:

   ```
   $ bash $dftfe_source/setupUser.h
   ```

   which will create the following debug mode executables:

   ```
   $dftfe_build_dir/debug/real/dftfe
   $dftfe_build_dir/debug/complex/dftfe
   ```

3.3 Optional PETSc, SLEPc, deal.II and DFT-FE installation instructions for all-electron calculations using DFT-FE

Users can ignore this section if only interested in pseudopotential calculations. The optional dependencies of PETSc and SLEPc are only required for all-electron calculations using DFT-FE, which uses the more stable Gram-Schmidt orthogonalization routine instead of the default Cholesky-Gram-Schmidt orthognalization.

1. **PETSc**: PETSc is a parallel linear algebra library. DFT-FE with PETSc and SLEPc dependencies 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 (3.15.0 or later) 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 (required) from the tarball and repeat the above steps with the only changes being adding `--with-scalar-type=complex` and `--with-fortran-kernels=true` to the configuration step (`./configure`) as well as providing a new installation path to `--prefix`. Below we provide example configure lines for real and complex versions on UMICH Greatlakes supercomputer with GNU compiler, Open MPI library, and Intel MKL math library:

   ```
   ./configure --prefix=petsc_real_install_path --with-debugging=no
   --with-64-bit-indices=true --with-cc=mpicc --with-cxx=mpicxx
   --with-fc=mpif90 --with-blas-lapack-lib="-Wl,--start-group
   ${MKLROOT}/lib/intel64/libmkl_intel_lp64.a
   ${MKLROOT}/lib/intel64/libmkl_gnu_thread.a
   ${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -lgomp -lpthread -lm -ldl"
   CFLAGS="-O2" CXXFLAGS="-O2" FFLAGS="-O2"

   ./configure --prefix=petsc_complex_install_path --with-debugging=no
   --with-64-bit-indices=true --with-cc=mpicc --with-cxx=mpicxx
   --with-fc=mpif90 --with-fortran-kernels=true --with-scalar-type=complex
   --with-blas-lapack-lib="-Wl,--start-group
   ${MKLROOT}/lib/intel64/libmkl_intel_lp64.a ${MKLROOT}/lib/intel64/libmkl_gnu_thread.a
   ${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -lgomp -lpthread -lm -ldl"
   CFLAGS="-O2" CXXFLAGS="-O2" FFLAGS="-O2"
   ```

2. **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 (3.15.0 or later) 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
```

3. **deal.II**: Assuming PETSc and SLEPc 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.

```
$ mkdir buildReal
$ cd buildReal
$ cmake -DCMAKE_INSTALL_PREFIX=dealii_petscReal_install_dir_path
        otherCmakeOptions ../deal.II
$ make install
```

   "otherCmakeOptions" include the following options

```
-DCMAKE_C_COMPILER=c_compiler
-DCMAKE_CXX_COMPILER=cxx_compiler
-DCMAKE_Fortran_COMPILER=fortran_compiler
-DMPI_C_COMPILER=mpi_c_compiler_wrapper
-DMPI_CXX_COMPILER=mpi_cxx_compiler_wrapper
-DMPI_Fortran_COMPILER=mpi_fortran_compiler_wrapper
-DCMAKE_CXX_FLAGS=cxx_flags
-DCMAKE_C_FLAGS=c_flags
-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
-DDEAL_II_WITH_LAPACK=ON
-DLAPACK_DIR=lapack_dir_path
-DLAPACK_FOUND=true
-DLAPACK_LIBRARIES=lapack_lib_path
-DSCALAPACK_DIR=scalapack_dir_path (only required if linking to Netlib ScaLAPACK)
-DSCALAPACK_LIBRARIES=scalapack_lib_path
-DDEAL_II_WITH_TBB=OFF
-DDEAL_II_WITH_TASKFLOW=OFF
-DDEAL_II_COMPONENT_EXAMPLES=OFF
```

4. **DFT-FE**: Follow the same instructions as in Sec. 3.2, except to modify the `setupUserPetsc.sh` script instead of the `setupUser.sh`. In `setupUserPetsc.sh`, in addition to updating the paths and flags as discussed in Sec 3.2, update the dealii installation paths from the previous step as follows:

```
dealiiPetscRealDir=dealii_petscReal_install_dir_path
dealiiPetscComplexDir=dealii_petscComplex_install_dir_path
```

3.4 Important generic instructions

- We strongly recommend to link to optimized BLAS-LAPACK library. If using Intel MKL for BLAS-LAPACK library, it is **very important** to use Intel MKL Link Line Advisor to correctly link with Intel MKL for installations of PETSc, ScaLAPACK, ELPA, deal.II, and PETSc. 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 DFT-FE and its dependencies, to prevent linking errors during DFT-FE compilation.
- **CAUTION! It is highly recommended to compile deal.II, p4est, ScaLAPACK, ELPA, DFT-FE, PETSc and SLEPc with the same compilers, same BLAS-LAPACK libraries (if applicable), and same MPI libraries. This prevents deal.II compilation issues, occurrence of run time crashes, and DFT-FE performance degradation.**

4 Running DFT-FE

After compiling DFT-FE as described in Section 3, we have now two executables — `$dftfe_build_dir/release/real/dftfe` and `$dftfe_build_dir/release/complex/dftfe`. The `$dftfe_build_dir/release/real/dftfe` executable, which uses real data-structures is sufficient for fully non-periodic problems. The executable can also be used for periodic and semi-periodic problems involving a Gamma point calculation. On the other hand the `$dftfe_build_dir/release/complex/dftfe` 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 for a parallel run as follows:

```
mpirun -n N ./dftfe 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.18. 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 `/demo/` folder. We refer to

`https://github.com/dftfeDevelopers/dftfe-benchmarks.git`

repository for a comphrehensive list of accuracy and performance examples, which are designed to cover most of the capabilities of the DFT-FE code. Further, we note that the demo examples in the `/demo/` folder 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.18.

4.2.1 Example 1

Let us consider the first example given in the folder `/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 40 a.u as the edge length. Accordingly, the "domainVectors.inp" file is formatted as

  ```
  40.0 0.0 0.0
  0.0 40.0 0.0
  0.0 0.0 40.0
  ```

  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.upf
  ```

  where "7" is the atomic number of Nitrogen, and `N.upf` is the Optimized Norm-Conserving Vanderbilt pseudopotential (ONCV) file obtained from http://www.pseudo-dojo.org/. **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 MIXING PARAMETER = 0.5
     set MAXIMUM ITERATIONS               = 40
     set TEMPERATURE                      = 500
     set TOLERANCE                        = 5e-5
     subsection Eigen-solver parameters
         set NUMBER OF KOHN-SHAM WAVEFUNCTIONS = 12
     end
   end
   ```

   where

   - "0.5" set for `MIXING PARAMETER` is the mixing parameter to be used in the mixing 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.
   - "5e-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. The three important FE discretization related parameters in DFT-FE, which the user needs to set are the POLYNOMIAL ORDER, MESH SIZE AROUND ATOM, and ATOM BALL RADIUS.

   First, the POLYNOMIAL ORDER sets the order of the piecewise continuous FE interpolating polynomial, with higher values affording faster convergence rates with respect to discretization. Default value is set to 6. Based on our numerical investigations, we recommend POLYNOMIAL ORDER=7 for soft pseudopotentials (< 20 Ha plane-wave cutoff), POLYNOMIAL ORDER=6 for medium hard/very hard pseudpotentials, and POLYNOMIAL ORDER=5 for all-electron calculations to be most computationally efficient choices on both CPUs and GPUs.

   Second, the MESH SIZE AROUND ATOM input parameter sets the size (in Bohr units) of the FE mesh element around the atoms. Mesh sizes away from the atoms and the intervening mesh adaptivity is heuristically set inside the code depending on the domain boundary conditions. The MESH SIZE AROUND ATOM is the most important parameter for the user to control in pseudopotential DFT calculations. This parameter is inversely related to the plane-wave cutoff in plane-wave based DFT codes. Smaller mesh sizes lead to lower discretization related errors, but at the cost of more degrees of freedom. Based on our numerical validation studies as will be demonstrated in many of the examples, we obtain chemical accuracy (10^{-4} Ha/atom in energy, 10^{-4} Ha/Bohr in ionic forces and 5×10^{-6} Ha/Bohr^3 in cell stresses) using the following choice of discretization parameters: (POLYNOMIAL ORDER=7, MESH SIZE AROUND ATOM=1.5 — 2.5) for soft pseudopotentials, and (POLYNOMIAL ORDER=6, MESH SIZE AROUND ATOM=0.5 — 1.5) for medium hard/very hard pseudopotentials. We advice the users to use the above recommendations as starting choices only and always check the convergence of their quantity of interest with respect to MESH SIZE AROUND ATOM as described below for the current demo example. For all-electron calculations, a value of around 0.5 is a good starting choice.

   Third, the ATOM BALL RADIUS input parameter denotes the radius of ball enclosing every atom (in a.u.), inside which the mesh size is set close to MESH SIZE AROUND ATOM and coarse-grained in the region outside the enclosing balls. For the default value of 0.0, a heuristically determined value is used, which is good enough for most cases but can be a bit conservative choice for fully non-periodic and semi-periodic problems as well as all-electron problems. To improve the computational efficiency user may experiment with values of ATOM BALL RADIUS ranging between 3.0 to 6.0 for pseudopotential problems, and ranging between 1.0 to 2.5 for all-electron problems.

   For the current problem at hand involving N_2 molecule which is a medium hard pseudopotential we set the starting finite-element mesh parameters as below

```
subsection Finite element mesh parameters
  set POLYNOMIAL ORDER = 6
  subsection Auto mesh generation parameters
    set ATOM BALL RADIUS = 3
    set MESH SIZE AROUND ATOM  = 1.6
  end
end
```

   and now run the problem using the `/build/release/real/dftfe` executable on "xx" number of MPI tasks

```
mpirun -n xx ../../build/release/real/dftfe 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 thrice, once with `MESH SIZE AROUND ATOM = 1.6`, then with `MESH SIZE AROUND ATOM = 1.3`, and finally with `MESH SIZE AROUND ATOM = 1.0`. Now run one more time with `POLYNOMIAL ORDER = 7` while keeping `MESH SIZE AROUND ATOM = 1.0`. We recommend to run this final simulation with more 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. 3* we obtain convergence in energy per atom to $\mathcal{O}(10^{-5})$ accuracy, and convergence in force to $\mathcal{O}(10^{-5})$ accuracy. For your reference, we have provided the output file for *Mesh No. 3* 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.

```
subsection Postprocessing
   set WRITE DENSITY=true
end
```

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`	Total degrees of freedom per atom	Energy per atom (Hartree)	Force on atomId 0 (Hartree/Bohr)
1	6	1.6	39,595	-10.31830282	0.2947253
2	6	1.3	73,093	-10.32060716	0.2929309
3	6	1.0	141,291	-10.32084447	0.2928696
4	7	1.0	222,288	-10.32086549	0.2928277

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. 3*, 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 2.0846 Bohr after 9 geometry update steps. You can also obtain the relaxed geometry from the `atomsCartCoordCurrent.chk` file generated by the code. In case of simulations with periodic or semi-periodic boundary conditions the corresponding file generated would be `atomsFracCoordCurrent.chk`. For your reference, we have provided an output file at `/demo/ex1/ex1_b.output`, which was run using 36 MPI tasks.

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 `/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 hydrostatic 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. 1* we have obtained convergence in energy per atom to $\mathcal{O}(10^{-5})$ accuracy, and convergence in cell stress to $\mathcal{O}(10^{-7})$ accuracy. The

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

Mesh No.	`POLYNOMIAL ORDER`	`MESH SIZE AROUND ATOM`	Total degrees of freedom per atom	Energy per atom (Hartree)	Hydrostatic Cell stress (Hartree/Bohr^3)
1	5	1.6	4,394	-2.30902819	-0.000109835
2	5	1.3	7,447	-2.30899310	-0.000110529
3	6	1.3	12,663	-2.30900353	-0.000110531

output file using the mesh parameters for *Mesh No.1* 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 7.699 Bohr after two geometry updates. You can also obtain the relaxed geometry from the `domainBoundingVectorsCurrent.chk` file generated by the code.

5 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.18.

A.1 Global parameters

- *Parameter name:* `H REFINED ELECTROSTATICS`

  Default: false

  Description: [Advanced] Compute electrostatic energy on a h refined mesh after each ground-state solve. Default: false.

  Possible values: A boolean value (true or false)

- *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 some more additional information), 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 $0 \leq n \leq 5$

A.2 Parameters in section `Boundary conditions`

- *Parameter name:* `CONSTRAINTS FROM SERIAL DOFHANDLER`

  Default: false

  Description: [Developer] Create constraints from serial dofHandler.

  Possible values: A boolean value (true or false)

- *Parameter name:* `CONSTRAINTS PARALLEL CHECK`

  Default: false

  Description: [Developer] Check for consistency of constraints in parallel.

  Possible values: A boolean value (true or false)

- *Parameter name:* `FLOATING NUCLEAR CHARGES`

  Default: true

  Description: [Developer] Nuclear charges are allowed to float independent of the FEM mesh nodal positions. Only allowed for pseudopotential calculations. Internally set to false for all-electron calculations.

  Possible values: A boolean value (true or false)

- *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:* `POINT WISE DIRICHLET CONSTRAINT`

  Default: false

  Description: [Developer] Flag to set point wise dirichlet constraints to eliminate null-space associated with the discretized Poisson operator subject to periodic BCs.

  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 $0 \leq v \leq 50$

- *Parameter name:* `SMEARED NUCLEAR CHARGES`

  Default: true

  Description: [Developer] Nuclear charges are smeared for solving electrostatic fields. Default is true for pseudopotential calculations and false for all-electron calculations.

  Possible values: A boolean value (true or false)

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 providing list of k points on which eigen values are to be computed from converged KS Hamiltonian. The first three columns specify the crystal coordinates of the k points. The fourth column provides weights of the corresponding points, which is currently not used. The eigen values are written on an output file bands.out

  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 using the electron-density field. 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 $0 \leq n \leq 3$

- *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. Further, for CHK

TYPE=2 same number of MPI tasks must be used as used to create the checkpoint files. RESTART FROM CHK is always false for CHK TYPE 0.

Possible values: A boolean value (true or false)

- *Parameter name:* `RESTART MD FROM CHK`

  Default: false

  Description: [Developer] Boolean parameter specifying if the current job reads from a MD checkpoint (in development).

  Possible values: A boolean value (true or false)

- *Parameter name:* `RESTART SP FROM NO SP`

  Default: false

  Description: [Standard] Enables ground-state solve for SPIN POLARIZED case reading the SPIN UNPOLARIZED density from the checkpoint files, and use the START MAGNETIZATION to compute the spin up and spin down densities. This option is only valid for CHK TYPE=2 and RESTART FROM CHK=true. Default false..

  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)], 5(RPBE: B. Hammer, L. B. Hansen, and J. K. Nørskov, Phys. Rev. B 59, 7413 (1999)).

  Possible values: An integer n such that $1 \leq n \leq 5$

- *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 or http://www.pseudo-dojo.org/. Troullier-Martins pseudopotentials in UPF format can be downloaded from http://www.quantum-espresso.org/pseudopotentials/f 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:* `PSP CUTOFF IMAGE CHARGES`

  Default: 15.0

  Description: [Standard] Distance from the domain till which periodic images will be considered for the local part of the pseudopotential. Units in a.u.

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

- *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:* `POLYNOMIAL ORDER`

  Default: 6

  Description: [Standard] The degree of the finite-element interpolating polynomial in the Kohn-Sham Hamitonian except the electrostatics. Default value is 6 which is good choice for most pseudopotential calculations. POLYNOMIAL ORDER= 4 or 5 is usually a good choice for all-electron problems.

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

- *Parameter name:* `POLYNOMIAL ORDER ELECTROSTATICS`

  Default: 0

  Description: [Standard] The degree of the finite-element interpolating polynomial for the electrostatics part of the Kohn-Sham Hamiltonian. It is automatically set to POLYNOMIAL ORDER if POLYNOMIAL ORDER ELECTROSTATICS set to default value of zero.

  Possible values: An integer n such that $0 \leq n \leq 24$

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

- *Parameter name:* `ATOM BALL RADIUS`

  Default: 0.0

Description: [Standard] Radius of ball enclosing every atom, inside which the mesh size is set close to MESH SIZE AROUND ATOM and coarse-grained in the region outside the enclosing balls. For the default value of 0.0, a heuristically determined value is used, which is good enough for most cases but can be a bit conservative choice for fully non-periodic and semi-periodic problems as well as all-electron problems. To improve the computational efficiency user may experiment with values of ATOM BALL RADIUS ranging between 3.0 to 6.0 for pseudopotential problems, and ranging between 1.0 to 2.5 for all-electron problems. Units: a.u.

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

- *Parameter name:* `AUTO ADAPT BASE MESH SIZE`

  Default: true

  Description: [Developer] Automatically adapt the BASE MESH SIZE such that subdivisions of that during refinement leads closest to the desired MESH SIZE AROUND ATOM. Default: true.

  Possible values: A boolean value (true or false)

- *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:* `ERROR ESTIMATE WAVEFUNCTIONS`

  Default: 5

  Description: [Developer] Number of wavefunctions to be used for error estimation.

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

- *Parameter name:* `GAUSSIAN CONSTANT FORCE GENERATOR`

  Default: 0.75

  Description: [Developer] Force computation generator gaussian constant. Also used for mesh movement. Gamma(r)= exp(-(r/gaussianConstant);(gaussianOrder)).

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

- *Parameter name:* `GAUSSIAN ORDER FORCE GENERATOR`

  Default: 4.0

  Description: [Developer] Force computation generator gaussian order. Also used for mesh movement. Gamma(r)= exp(-(r/gaussianConstant);(gaussianOrder)).

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

- *Parameter name:* `GAUSSIAN ORDER MOVE MESH TO ATOMS`

  Default: 4.0

  Description: [Developer] Move mesh to atoms gaussian order. Gamma(r)= exp(-(r/gaussianConstant);(gaussianOrder)).

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

- *Parameter name:* `INNER ATOM BALL RADIUS`

  Default: 0.0

  Description: [Advanced] Radius of ball enclosing every atom, inside which the mesh size is set close to MESH SIZE AT ATOM. 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 ADAPTION`

  Default: false

  Description: [Developer] Generates adaptive mesh based on a-posteriori mesh adaption strategy using single atom wavefunctions before computing the ground-state. Default: false.

  Possible values: A boolean value (true or false)

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

  Default: 1.0

  Description: [Standard] Mesh size in a ball of radius ATOM BALL RADIUS around every atom. For pseudopotential calculations, the value ranges between 0.8 to 2.5 depending on the cutoff energy for the pseudopotential. For all-electron calculations, a value of around 0.5 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 for all-electron calculations. For pseudopotential calculations, the default value of 0.0, sets the MESH SIZE AT ATOM to be the same value as MESH SIZE AROUND ATOM. Standard users do not need to tune this parameter. Units: a.u.

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

- *Parameter name:* `NUM LEVELS`

  Default: 10

  Description: [Developer] Number of times to be refined.

  Possible values: An integer n such that $0 \leq n \leq 30$

- *Parameter name:* `TOLERANCE FOR MESH ADAPTION`

  Default: 1

  Description: [Developer] Tolerance criteria used for stopping the multi-level mesh adaption done apriori using single atom wavefunctions. This is used as Kinetic energy change between two successive iterations

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

- *Parameter name:* `TOP FRAC`

  Default: 0.1

  Description: [Developer] Top fraction of elements to be refined.

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

- *Parameter name:* `USE FLAT TOP GENERATOR`

  Default: false

  Description: [Developer] Use a composite generator flat top and Gaussian generator for mesh movement and configurational force computation.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MESH SIZES FROM ATOM LOCATIONS FILE`

  Default: false

  Description: [Developer] Use mesh sizes from atom locations file.

  Possible values: A boolean value (true or false)

A.9 Parameters in section `GPU`

- *Parameter name:* `AUTO GPU BLOCK SIZES`

  Default: true

  Description: [Advanced] Automatically sets total number of kohn-sham wave functions and eigensolver optimal block sizes for running on GPUs. If manual tuning is desired set this parameter to false and set the block sizes using the input parameters for the block sizes. Default: true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `FINE GRAINED GPU TIMINGS`

  Default: false

  Description: [Developer] Print more fine grained GPU timing results. Default: false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `GPU MEM OPT MODE`

  Default: true

  Description: [Adavanced] Uses algorithms which have lower peak memory on GPUs but with a marginal performance degradation. Recommended when using more than 100k degrees of freedom per GPU. Default: true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `SUBSPACE ROT FULL CPU MEM`

  Default: true

  Description: [Developer] Option to use full NxN memory on CPU in subspace rotation and when mixed precision optimization is not being used. This reduces the number of MPI_Allreduce communication calls. Default: true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE ELPA GPU KERNEL`

  Default: false

  Description: [Advanced] If DFT-FE is linked to ELPA eigensolver library configured to run on GPUs, this parameter toggles the use of ELPA GPU kernels for dense symmetric matrix diagonalization calls in DFT-FE. ELPA version>=2020.11.001 is required for this feature. Default: false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE GPU`

  Default: false

  Description: [Standard] Use GPU for compute.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE GPUDIRECT MPI ALL REDUCE`

  Default: false

  Description: [Adavanced] Use GPUDIRECT MPI_Allreduce. This route will only work if DFT-FE is compiled with NVIDIA NCCL library. Also note that one MPI rank per GPU can be used when using this option. Default: false.

  Possible values: A boolean value (true or false)

A.10 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:* `ATOMIC DISP COORDINATES FILE`

  Default:

  Description: [Standard] Atomic displacement coordinates input file name. The FEM mesh is deformed using Gaussian functions attached to the atoms. File format (example for two atoms): delx1 dely1 delz1 (row1), delx2 dely2 delz2 (row2). The number of rows must be equal to NATOMS. Units in a.u.

  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), v2y v2y v2z (row2), v3z 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.11 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 v2y), 5 (relax along domain vector component v3z), 6 (relax along domain vector components v2y and v3z), 7 (relax along domain vector components v1x and v3z), 8 (relax along domain vector components v1x and v2y), 9 (relax along domain vector components v1x, v2y and v3z), 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 during atomic relaxation, when the 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 OPT SOLVER`

  Default: CGPRP

  Description: [Standard] Method for Ion relaxation solver. CGPRP (Nonlinear conjugate gradient with Secant and Polak-Ribiere approach) is the default

  Possible values: Any one of CGDESCENT, LBFGS, CGPRP

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

  Default:

  Description: [Standard] File specifying the permission flags (1-free to move, 0-fixed) and external forces for the 3-coordinate directions and for all atoms. File format (example for two atoms with atom 1 fixed and atom 2 free and 0.01 Ha/Bohr force acting on atom 2): 0 0 0 0.0 0.0 0.0(row1), 1 1 1 0.0 0.0 0.01(row2). External forces are optional.

  Possible values: Any string

- *Parameter name:* `MAX LINE SEARCH ITER`

  Default: 5

  Description: [Standard] Sets the maximum number of line search iterations in the case of CGPRP. Default is 5.

  Possible values: An integer n such that $1 \leq n \leq 100$

- *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:* `REUSE DENSITY`

  Default: 0

  Description: [Standard] Parameter controlling the reuse of ground-state density during geometry optimization. The options are 0 (reinitialize density based on superposition of atomic densities), 1 (reuse ground-state density of previous relaxation step), and 2 (subtract superposition of atomic densities from the previous step's ground-state density and add superposition of atomic densities from the new atomic positions. Option 2 is not enabled for spin-polarized case. Default setting is 0.

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

- *Parameter name:* `REUSE WFC`

  Default: true

  Description: [Standard] Reuse previous ground-state wavefunctions during geometry optimization. Default setting is true.

  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.12 Parameters in section `Helmholtz problem parameters`

- *Parameter name:* `ABSOLUTE TOLERANCE HELMHOLTZ`

  Default: 1e-10

  Description: [Advanced] Absolute tolerance on the residual as stopping criterion for Helmholtz problem convergence.

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

- *Parameter name:* `MAXIMUM ITERATIONS HELMHOLTZ`

  Default: 10000

  Description: [Advanced] Maximum number of iterations to be allowed for Helmholtz problem convergence.

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

A.13 Parameters in section `Molecular Dynamics`

- *Parameter name:* `ATOMIC MASSES FILE`

  Default:

  Description: [Standard] Input atomic masses file name. File format: atomicNumber1 atomicMass1 (row1), atomicNumber2 atomicMass2 (row2) and so on. Units: a.m.u.

  Possible values: Any string

- *Parameter name:* `BOMD`

  Default: false

  Description: [Standard] Perform Born-Oppenheimer NVE molecular dynamics. Input parameters for molecular dynamics have to be modified directly in the code in the file md/molecularDynamics.cc.

  Possible values: A boolean value (true or false)

- *Parameter name:* `CHEBY TOL XL BOMD`

  Default: 1e-6

  Description: [Standard] Parameter specifying the accuracy of the occupied eigenvectors close to the Fermi-energy computed using Chebyshev filtering subspace iteration procedure.

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

- *Parameter name:* `CHEBY TOL XL BOMD RANK UPDATES FD`

  Default: 1e-7

  Description: [Standard] Parameter specifying the accuracy of the occupied eigenvectors close to the Fermi-energy computed using Chebyshev filtering subspace iteration procedure.

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

- *Parameter name:* `CHEBY TOL XL BOMD RESTART`

  Default: 1e-9

  Description: [Standard] Parameter specifying the accuracy of the occupied eigenvectors close to the Fermi-energy computed using Chebyshev filtering subspace iteration procedure.

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

- *Parameter name:* `DENSITY MATRIX PERTURBATION RANK UPDATES XL BOMD`

  Default: false

  Description: [Standard] Use density matrix perturbation theory for rank updates.

  Possible values: A boolean value (true or false)

- *Parameter name:* `DIRAC DELTA KERNEL SCALING CONSTANT XL BOMD`

  Default: 0.1

  Description: [Developer] Dirac delta scaling kernel constant for XL BOMD.

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

- *Parameter name:* `KERNEL RANK XL BOMD`

  Default: 0

  Description: [Standard] Maximum rank for low rank kernel update in XL BOMD.

  Possible values: An integer n such that $0 \leq n \leq 10$

- *Parameter name:* `MAX JACOBIAN RATIO FACTOR`

  Default: 1.5

  Description: [Developer] Maximum scaling factor for maximum jacobian ratio of FEM mesh when mesh is deformed.

  Possible values: A floating point number v such that $0.9 \leq v \leq 3$

- *Parameter name:* `NUMBER DISSIPATION TERMS XL BOMD`

  Default: 8

  Description: [Standard] Number of dissipation terms in XL BOMD.

  Possible values: An integer n such that $1 \leq n \leq 8$

- *Parameter name:* `NUMBER OF STEPS`

  Default: 1000

  Description: [Standard] Number of time steps.

  Possible values: An integer n such that $0 \leq n \leq 200000$

- *Parameter name:* `NUMBER PASSES RR SKIPPED XL BOMD`

  Default: 0

  Description: [Standard] Number of starting chebsyev filtering passes without Rayleigh Ritz in XL BOMD.

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

- *Parameter name:* `STARTING TEMP NVE`

  Default: 300.0

  Description: [Developer] Starting temperature in K for NVE simulation.

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

- *Parameter name:* `TIME STEP`

  Default: 0.5

  Description: [Standard] Time step in femtoseconds.

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

- *Parameter name:* `USE ATOMIC RHO XL BOMD`

  Default: true

  Description: [Standard] Use atomic rho xl bomd.

  Possible values: A boolean value (true or false)

- *Parameter name:* `XL BOMD`

  Default: false

  Description: [Standard] Perform Extended Lagrangian Born-Oppenheimer NVE molecular dynamics. Currently not implemented for spin-polarization case.

  Possible values: A boolean value (true or false)

- *Parameter name:* `XL BOMD KERNEL RANK UPDATE FD PARAMETER`

  Default: 1e-2

  Description: [Standard] Finite difference perturbation parameter.

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

A.14 Parameters in section `Parallelization`

- *Parameter name:* `BAND PARAL OPT`

  Default: true

  Description: [Standard] Uses a more optimal route for band parallelization but at the cost of extra wavefunctions memory.

  Possible values: A boolean value (true or false)

- *Parameter name:* `MPI ALLREDUCE BLOCK SIZE`

  Default: 100.0

  Description: [Advanced] Block message size in MB used to break a single MPI_Allreduce call on wavefunction vectors data into multiple MPI_Allreduce calls. This is useful on certain architectures which take advantage of High Bandwidth Memory to improve efficiency of MPI operations. This variable is relevant only if NPBAND>1. Default value is 100.0 MB.

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

- *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.15 Parameters in section `Poisson problem parameters`

- *Parameter name:* `MAXIMUM ITERATIONS`

  Default: 20000

  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-10

  Description: [Advanced] Absolute tolerance on the residual as stopping criterion for Poisson problem convergence.

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

A.16 Parameters in section `Postprocessing`

- *Parameter name:* `READ ATOMIC WFC PDOS FROM PSP FILE`

  Default: false

  Description: [Standard] Read atomic wavefunctons from the pseudopotential file for computing projected density of states. When set to false atomic wavefunctions from the internal database are read, which correspond to sg15 ONCV pseudopotentials.

  Possible values: A boolean value (true or false)

- *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 DENSITY OF STATES`

  Default: false

  Description: [Standard] Computes density of states using Lorentzians. Uses specified Temperature for SCF as the broadening parameter. Outputs a file name 'dosData.out' containing two columns with first column indicating the energy in eV and second column indicating the density of states

  Possible values: A boolean value (true or false)

- *Parameter name:* `WRITE LOCALIZATION LENGTHS`

  Default: false

  Description: [Standard] Computes localization lengths of all wavefunctions which is defined as the deviation around the mean position of a given wavefunction. Outputs a file name 'localizationLengths.out' containing 2 columns with first column indicating the wavefunction index and second column indicating localization length of the corresponding wavefunction.

  Possible values: A boolean value (true or false)

- *Parameter name:* `WRITE LOCAL DENSITY OF STATES`

  Default: false

  Description: [Standard] Computes local density of states on each atom using Lorentzians. Uses specified Temperature for SCF as the broadening parameter. Outputs a file name 'ldosData.out' containing NUMATOM+1 columns with first column indicating the energy in eV and all other NUMATOM columns indicating local density of states for each of the NUMATOM atoms.

  Possible values: A boolean value (true or false)

- *Parameter name:* `WRITE PROJECTED DENSITY OF STATES`

  Default: false

  Description: [Standard] Computes projected density of states on each atom using Lorentzians. Uses specified Temperature for SCF as the broadening parameter. Outputs a file name 'pdosData_x' with x denoting atomID. This file contains columns with first column indicating the energy in eV and all other columns indicating projected density of states corresponding to single atom wavefunctions.

  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.17 Parameters in section `SCF parameters`

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

  Default: false

  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. Default value is false but is internally set to true if VERBOSITY==5

  Possible values: A boolean value (true or false)

- *Parameter name:* `CONSTRAINT MAGNETIZATION`

  Default: false

  Description: [Standard] Boolean parameter specifying whether to keep the starting magnetization fixed through the SCF iterations. Default is FALSE

  Possible values: A boolean value (true or false)

- *Parameter name:* `KERKER MIXING PARAMETER`

  Default: 0.05

  Description: [Standard] Mixing parameter to be used in Kerker mixing scheme which usually represents Thomas Fermi wavevector (k_TF**2).

  Possible values: A floating point number v such that $0 \leq v \leq 1000$

- *Parameter name:* `MAXIMUM ITERATIONS`

  Default: 200

  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:* `MIXING HISTORY`

  Default: 50

  Description: [Standard] Number of SCF iteration history to be considered for density mixing schemes. 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:* `MIXING METHOD`

  Default: ANDERSON

  Description: [Standard] Method for density mixing. ANDERSON is the default option.

  Possible values: Any one of BROYDEN, ANDERSON, ANDERSON_WITH_KERKER

- *Parameter name:* `MIXING PARAMETER`

  Default: 0.2

  Description: [Standard] Mixing parameter to be used in density mixing schemes. Default: 0.2.

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

- *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-05

  Description: [Standard] SCF iterations stopping tolerance in terms of L_2 norm of the electron-density difference between two successive iterations. The default tolerance of is set to a tight value of 1e-5 for accurate ionic forces and cell stresses keeping structural optimization and molecular dynamics in mind. A tolerance of 1e-4 would be accurate enough for calculations without structural optimization and dynamics. 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.18 Parameters in section `SCF parameters/Eigen-solver parameters`

- *Parameter name:* `ALGO`

  Default: NORMAL

  Description: [Standard] In the FAST mode, spectrum splitting technique is used in Rayleigh-Ritz step, and mixed precision arithmetic algorithms are used in Rayleigh-Ritz and Cholesky factorization based orthogonalization step. For spectrum splitting, 85 percent of the total number of wavefunctions are taken to be core states, which holds good for most systems including metallic systems assuming NUMBER OF KOHN-SHAM WAVEFUNCTIONS to be around 10 percent more than N/2. FAST setting is strongly recommended for large-scale ($>$ 10k electrons) system sizes. Both NORMAL and FAST setting use Chebyshev filtered subspace iteration technique. If manual options for mixed precision and spectum splitting are being used, please use NORMAL setting for ALGO. Default setting is NORMAL.

  Possible values: Any one of NORMAL, FAST

- *Parameter name:* `ALLOW MULTIPLE PASSES POST FIRST SCF`

  Default: true

  Description: [Advanced] Allow multiple chebyshev filtering passes in the SCF iterations after the first one. Default setting is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `CHEBYSHEV FILTER TOLERANCE`

  Default: 5e-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:* `CHEBYSHEV POLYNOMIAL DEGREE SCALING FACTOR FIRST SCF`

  Default: 1.34

  Description: [Advanced] Chebyshev polynomial degree first scf scaling factor. Only activated for pseudopotential calculations.

  Possible values: A floating point number v such that $0 \leq v \leq 2000$

- *Parameter name:* `CHEBY WFC 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. For optimum work sharing during band parallelization (NPBAND > 1), we recommend adjusting CHEBY WFC BLOCK SIZE and NUMBER OF KOHN-SHAM WAVEFUNCTIONS such that NUMBER OF KOHN-SHAM WAVEFUNCTIONS/NPBAND/CHEBY WFC BLOCK SIZE equals an integer value. Default value is 400.

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

- *Parameter name:* `ENABLE HAMILTONIAN TIMES VECTOR OPTIM`

  Default: true

  Description: [Advanced] Turns on optimization for hamiltonian times vector multiplication. Operations involving data movement from global vector to finite-element cell level and vice versa are done by employing different data structures for interior nodes and surfaces nodes of a given cell and this allows reduction of memory access costs

  Possible values: A boolean value (true or false)

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

  Default: 0

  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 5-10 percent of N/2 for insulators and for metals use Nb to be 10-20 percent

of N/2. If 5-20 percent of N/2 is less than 10 wavefunctions, set Nb to be atleast 10. Default value of 0 automatically sets the number of Kohn-Sham wavefunctions close to 20 percent more than N/2. CAUTION: use more states when using higher electronic temperature.

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) and CGS(Cholesky-Gram-Schmidt Orthogonalization). Auto is the default and recommended option, which chooses GS for all-electron case and CGS for pseudopotential case. On GPUs CGS is the only route currently implemented.

  Possible values: Any one of GS, CGS, Auto

- *Parameter name:* `OVERLAP COMPUTE COMMUN CHEBY`

  Default: true

  Description: [Advanced] Overlap communication and computation in Chebyshev filtering. This option can only be activated for USE GPU=true. Default setting is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `OVERLAP COMPUTE COMMUN ORTHO RR`

  Default: true

  Description: [Advanced] Overlap communication and computation in orthogonalization and Rayleigh-Ritz. This option can only be activated for USE GPU=true. Default setting is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `REUSE LANCZOS UPPER BOUND`

  Default: false

  Description: [Advanced] Reuse upper bound of unwanted spectrum computed in the first SCF iteration via Lanczos iterations. Default setting is false.

  Possible values: A boolean value (true or false)

- *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 Cholesky-Gram-Schmidt step. Default value is 0 for which a thumb rule is used (see http://netlib.org/scalapack/slug/node106.html). If ELPA is used, twice the value obtained from the thumb rule is used as ELPA scales much better than ScaLAPACK.

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

- *Parameter name:* `SCALAPACK BLOCK SIZE`

  Default: 0

  Description: [Advanced] ScaLAPACK process grid block size. Also sets the block size for ELPA if linked to ELPA. Default value of zero sets a heuristic block size. Note that if ELPA GPU KERNEL is set to true and ELPA is configured to run on GPUs, the SCALAPACK BLOCK SIZE is set to a power of 2.

  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 diagonalization. In other words, only the eigenvalues and eigenvectors corresponding to the higher eigenstates (Number of Kohn-Sham wavefunctions minus the specified core eigenstates) are computed in the diagonalization of the projected 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. 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:* `SPECTRUM SPLIT STARTING SCF ITER`

  Default: 0

  Description: [Advanced] SCF iteration no beyond which spectrum splitting based can be used.

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

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

  Default: 10000

  Description: [Developer] This block size is used for memory optimization purposes in subspace rotation step in Cholesky-Gram-Schmidt orthogonalization and Rayleigh-Ritz steps. Default value is 10000.

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

- *Parameter name:* `USE ELPA`

  Default: true

  Description: [Standard] Use ELPA instead of ScaLAPACK for diagonalization of subspace projected Hamiltonian and Cholesky-Gram-Schmidt orthogonalization. Default setting is true.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MIXED PREC CGS O`

  Default: false

  Description: [Advanced] Use mixed precision arithmetic in overlap matrix computation step of CGS orthogonalization, if ORTHOGONALIZATION TYPE is set to CGS. Default setting is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MIXED PREC CGS SR`

  Default: false

  Description: [Advanced] Use mixed precision arithmetic in subspace rotation step of CGS orthogonalization, if ORTHOGONALIZATION TYPE is set to CGS. Default setting is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MIXED PREC CHEBY`

  Default: false

  Description: [Advanced] Use mixed precision arithmetic in Chebyshev filtering. Currently this option is only available for real executable and USE ELPA=true for which DFT-FE also has to be linked to ELPA library. Default setting is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MIXED PREC RR_SR`

  Default: false

  Description: [Advanced] Use mixed precision arithmetic in Rayleigh-Ritz subspace rotation step. Default setting is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `USE MIXED PREC XTHX SPECTRUM SPLIT`

  Default: false

  Description: [Advanced] Use mixed precision arithmetic in computing subspace projected Kohn-Sham Hamiltonian when SPECTRUM SPLIT CORE EIGENSTATES>0. Default setting is false.

  Possible values: A boolean value (true or false)

- *Parameter name:* `WFC BLOCK SIZE`

  Default: 400

  Description: [Advanced] This parameter specifies the block size of the wavefunction matrix to be used for memory optimization purposes in the orthogonalization, Rayleigh-Ritz, and density computation steps. The optimum block size is dependent on the computing architecture. For optimum work sharing during band parallelization (NPBAND > 1), we recommend adjusting WFC BLOCK SIZE and NUMBER OF KOHN-SHAM WAVEFUNCTIONS such that NUMBER OF KOHN-SHAM WAVEFUNCTIONS/NPBAND/WFC BLOCK SIZE equals an integer value. Default value is 400.

  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.