invDFT

Inverse Density Functional theory calculations

User Manual

Version 1.0.0

(generated October 1, 2025)

Contents

1 Introduction

invDFT is a C++ code that performs inverse Density functional theory calculations. It is based on adaptive finite-element discretization and incorporates scalable and efficient solvers. Thus invDFT can be used to compute the exact v_{xc} potential for a given ground state electron density and external potential for large molecules consisting of 100 electrons. invDFT code builds on top of the DFT-FE to harness the efficient solvers, parallel adaptive mesh handling, and the finite-element infrastructure.

1.1 Authors

invDFT is developed and maintained by the Computational Materials Physics group at University of Michigan (directed by Prof. Vikram Gavini), The code is maintained by a group of principal developers, who manage the architecture of the code and the core functionalities. The list of authors is in alphabetical order.

Principal developers

- Bikash Kanungo (University of Michigan, USA).
- Vishal Subramanian (University of Michigan, USA).

Mentors/Development Leads

- Vikram Gavini (University of Michigan, USA).

1.2 Acknowledgments

The development of invDFT open-source code has been partly funded by the Department of Energy PRISMS Software Center at the University of Michigan, and the Toyota Research Institute.

2 Useful background information

We refer to the following articles for a background of the algorithms implemented in invDFT.

1. B. Kanungo, P. M. Zimmerman, V. Gavini, Exact exchange-correlation potentials from ground-state electron densities, *Nature communications* 10.1, 4497 (2019).

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).

2. S. Das, P. Motamarri, V. Subramanian, D. M. Rogers, & V Gavini, DFT-FE 1.0: A massively parallel hybrid CPU-GPU density functional theory code using finite-element discretization, *Comput. Phys. Commun.* 280 (2022).

3. 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).

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/HIP/SYCL (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 invDFT 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 cudatoolkit/12.4 (if installing for NVIDIA GPUs)
$ module load rocm/6.3.1 (if installing for AMD 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/HIP, 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
- cudatoolkit/12.0 (NVIDIA GPUs)
- rcom/6.3.1 (AMD 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

invDFT is primarily based on the open-sorced density functional theory library DFT-FE, through which external dependencies deal.II, p4est, kokkos and BLAS-LAPACK are set. The other required external libraries, which are not interfaced via deal.II are ALGLIB, Libxc, spglib, Libxml2 and ELPA (ELPA requires ScaLAPACK). DFT-FE also links to PETSc (via deali.II), SLEPc (via deali.II) and to nccl (for GPU compilation).

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, kokkos, 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 INVDFT for exchange-correlation functionals. Download the current release from `https://libxc.gitlab.io/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/spglib/spglib.git
   ```

   and next follow the "Compiling using cmake" installation procedure described in `https://spglib.readthedocs.io/en/stable/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 ELPA 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 (2.2.0) 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 `-fPIC -fallow-argument-mismatch` for `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 the latest version elpa-2025.01.001 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/AMD GPUs and CUDA/ROCm compiler, ELPA can take advantage of GPUs. For example on OLCF Summit GPU nodes, we use the following configure line

```
../configure CXX=mpic++ FC=mpif90 CC=mpicc
CXXFLAGS="-O2 -fPIC -mcpu=power9 -mvsx -maltivec"
FCFLAGS="-O2 -fPIC -mcpu=power9 -mvsx -maltivec"
CFLAGS="-O2 -fPIC -mcpu=power9 -mvsx -maltivec"
--enable-nvidia-gpu --with-NVIDIA-GPU-compute-capability="sm_70"
--enable-gpu-streams=nvidia --with-cuda-path="$OLCF_CUDA_ROOT"
--with-cuda-sdk-path="$OLCF_CUDA_ROOT"
--prefix=elpa_install_path
LDFLAGS="-L$scalapack_lib_path -lscalapack
-L$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 --enable-cpp-tests=no
--enable-option-checking=fatal --enable-shared
```

We provide another ELPA installation example on OLCF Crusher nodes with AMD GPUs and ROCm compiler:

```
../configure CXX=hipcc CC=hipcc FC=ftn
FCFLAGS="-march=znver3 -O2 -fPIC"
CFLAGS="-march=znver3 -fPIC -O2 -I${ROCM_PATH}/include
-I${ROCM_PATH}/hip/include -I${ROCM_PATH}/hipcub/include
-I${ROCM_PATH}/rocblas/include  --amdgpu-target=gfx90a
-I${MPICH_DIR}/include" CXXFLAGS="-march=znver3 -std=c++14
-I${ROCM_PATH}/include -fPIC -O2 -I${ROCM_PATH}/hip/include
-I${ROCM_PATH}/hipcub/include -I${ROCM_PATH}/rocblas/include
--amdgpu-target=gfx90a -I${MPICH_DIR}/include" --enable-amd-gpu
--enable-gpu-streams=amd --prefix=elpa_install_path
LIBS="-L${ROCM_PATH}/lib -lamdhip64 -L${ROCM_PATH}/rocblas/lib
-lrocblas -L${MPICH_DIR}/lib -lmpi -L${CRAY_MPICH_ROOTDIR}/gtl/lib -lmpi_gtl_hsa
-L$scalapack_lib_path -lscalapack
-L/sw/crusher/spack-envs/base/opt/cray-sles15-zen3/gcc-11.2.0/openblas-0.3.17-fip547hyhcrii2xgj5rmg
-L$netlib_lapack_lib_path -llapack"
--disable-sse --disable-sse-assembly --disable-avx --disable-avx2
--disable-avx512 --enable-c-tests=no --enable-option-checking=fatal
--enable-shared --enable-cpp-tests=no --enable-hipcub
```

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.** Finally, we remark that to avoid errors during ELPA configuration step, please export the library paths of ScaLAPACK and blas libraries, for example:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$blas_lib_path

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$netlib_lapack_lib_path

export LD_LIBRARY_PATH=
$LD_LIBRARY_PATH:$ROCM_PATH/lib:$ROCM_PATH/hip/lib:$ROCM_PATH/lib:$ROCM_PATH/rocblas/lib
```

7. **p4est**: This library is used by deal.II to create and distribute finite-element meshes across multiple processors. Download the latest release tarball of p4est from https://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. **kokkos**: This library is used by deal.II for performance portability across architectures. Download the latest release tarball of kokkos from https://github.com/kokkos/kokkos, and use cmake based compilation. For example,

```
$ cd kokkos-4.3.00
$ mkdir build && cd build
$ cmake -DCMAKE_C_COMPILER=cc -DCMAKE_C_FLAGS="-O2 -fPIC"
        -DCMAKE_CXX_COMPILER=CC
        -DCMAKE_CXX_FLAGS="-O2 -fPIC"
        -DCMAKE_INSTALL_PREFIX=kokkos_install_dir_path
        ..
$ make -j 4
$ make install
```

9. **nccl/rccl (optional)**: nccl (CUDA) and rccl (ROCm) libraries are 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/rccl library requires appropriate hardware support for GPU Direct MPI communication and GPU Aware MPI library. To install nccl/rccl, use the appropriate release version (corresponding to the CUDA/ROCm compiler version) from `https://github.com/NVIDIA/nccl` (CUDA) `https://github.com/ROCmSoftwarePlatform/rccl` (ROCm) and follow installation instructions therein.

10. **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"
```

11. **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
    ```

3.1.2 Instructions for deal.II

Assuming the above p4est dependency is 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 (9.6.2) release version of deal.II. It is important to only use this version of deal.II.

2. In addition to requiring C, C++ and Fortran compilers, deal.II requires MPI library, CMake, BOOST and Kokkos. During installation, deal.II has to be compiled with an external BOOST with version >= 1.59. Compiling deal.II with the bundled BOOST will result in compilation failure of DFT-FE. To compile deal.II with external boost one must set `-DDEAL_II_FORCE_BUNDLED_BOOST=OFF` and load appropriate BOOST library or set the path to the BOOST library manually with `-DBOOST_DIR=${BOOST_DIR}`. If BOOST is not installed a priori in the machine, please refer to BOOST https://www.boost.org for installation instructions. Please do not install deal.II with GPU support as that is not needed by DFT-FE. Further, when installing deal.II for (DFT-FE with GPU support), one must use `-DDEAL_II_ALLOW_PLATFORM_INTROSPECTION=OFF` to avoid compilation failure of DFT-FE.

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:

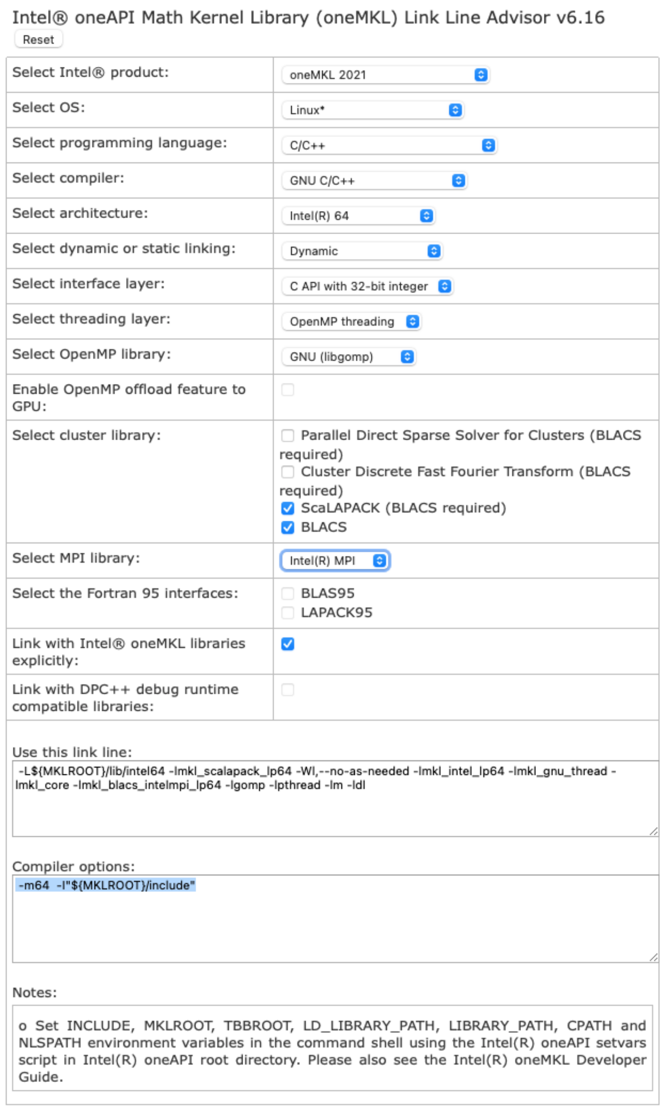
```
    -DCMAKE_CXX_STANDARD=17
    -DCMAKE_C_COMPILER=c_compiler
    -DCMAKE_CXX_COMPILER=cxx_compiler
    -DCMAKE_Fortran_COMPILER=fortran_compiler
    -DMPI_C_COMPILER=mpi_c_compiler_wrapper
    -DMPI_CXX_COMPILER=mpi_cxx_compiler_wrapper
    -DMPI_Fortran_COMPILER=mpi_fortran_compiler_wrapper
    -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
    -DKOKKOS_DIR=kokkos_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_TBB=OFF
    -DDEAL_II_WITH_TASKFLOW=OFF
    -DDEAL_II_COMPONENT_EXAMPLES=OFF
    -DDEAL_II_FORCE_BUNDLED_BOOST=OFF
    -DDEAL_II_ALLOW_PLATFORM_INTROSPECTION=OFF
```

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_CXX_STANDARD=17
-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 -std=c++17"
-DCMAKE_C_FLAGS="-march=native -std=c++17"
-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_FORCE_BUNDLED_BOOST=OFF
-DDEAL_II_ALLOW_PLATFORM_INTROSPECTION=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"
-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).



Intel® oneAPI Math Kernel Library (oneMKL) Link Line Advisor v6.16

Reset

Select Intel® product:	oneMKL 2021
Select OS:	Linux*
Select programming language:	C/C++
Select compiler:	GNU C/C++
Select architecture:	Intel(R) 64
Select dynamic or static linking:	Dynamic
Select interface layer:	C API with 32-bit integer
Select threading layer:	OpenMP threading
Select OpenMP library:	GNU (libgomp)
Enable OpenMP offload feature to GPU:	☐
Select cluster library:	☐ Parallel Direct Sparse Solver for Clusters (BLACS required) ☐ Cluster Discrete Fast Fourier Transform (BLACS required) ☑ ScaLAPACK (BLACS required) ☑ BLACS
Select MPI library:	Intel(R) MPI
Select the Fortran 95 interfaces:	☐ BLAS95 ☐ LAPACK95
Link with Intel® oneMKL libraries explicitly:	☑
Link with DPC++ debug runtime compatible libraries:	☐

Use this link line:

```
-L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -Wl,--no-as-needed -lmkl_intel_lp64 -lmkl_gnu_thread -
lmkl_core -lmkl_blacs_intelmpi_lp64 -lgomp -lpthread -lm -ldl
```

Compiler options:

```
-m64  -I"${MKLROOT}/include"
```

Notes:

o Set INCLUDE, MKLROOT, TBBROOT, LD_LIBRARY_PATH, LIBRARY_PATH, CPATH and NLSPATH environment variables in the command shell using the Intel(R) oneAPI setvars script in Intel(R) oneAPI root directory. Please also see the Intel(R) oneMKL Developer Guide.

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.

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
-DDEAL_II_WITH_TBB=OFF
-DDEAL_II_WITH_TASKFLOW=OFF
-DDEAL_II_COMPONENT_EXAMPLES=OFF
```

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 release1.0 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`.

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
   withDCCL=OFF
   ```

For GPU compilation, `withGPU`, `gpuLang`, and `gpuVendor` should be set `ON`, `"cuda"/"hip"` and `"nvidia"/"amd"` respectively. Also appropriate C++ compiler, device (denoting GPU) compilation and architecture flags must be passed to access CUDA/HIP/SYCL compiler for device code compilation. **CAUTION! GPU-enabled DFT-FE compilation must only be run on machines with GPU access.**

Currently, compilation of INVDFT requires DFT-FE be compiled without 64-bit integer support, this can be enabled by setting

```
useInt64=OFF
```

in `setupUser.sh`.

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.

modify the `setupUserPetsc.sh` script instead of the `setupUser.sh`. In `setupUserPetsc.sh`, update the dealii installation paths from the previous step as follows:

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

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 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 invDFT performance degradation.**

3.4 Obtaining and Compiling invDFT

Once the installation of DFT-FE is complete, the installation of INVDFT is straightforward.

1. Obtain the source code of the current release of INVDFT with all current patches using Git:

   ```
   % $ git clone -b main https://github.com/dftfeDevelopers/invDFT.git
   % $ cd invdft
   ```

   Do `git pull` in the `invdft` directory any time to obtain new patches that have been added since your `git clone` or last `git pull`.

2. Set paths to external libraries (DFT-FE, deal.II, ALGLIB, Libxc, spglib, Libxml2, and ELPA), C++ compiler, and C++ compiler flags in `install_invdft.sh`, which is a script to compile INVDFT 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
   withDCCL=OFF
   ```

   For GPU compilation, `withGPU`, `gpuLang`, and `gpuVendor` should be set `ON`, `"cuda"`/`"hip"` and `"nvidia"`/`"amd"` respectively. Also appropriate C++ compiler, device (denoting GPU) compilation and architecture flags must be passed to access CUDA/HIP compiler for device code compilation. To link DFT-FE, the path to the DFT-FE executable and its include dir is added as follows:

   ```
   dftfeRealDir=$dftfe_petscReal_install_dir_path
   dftfeIncludeDir=$dftfe_petscReal_include_dir_path
   ```

   To link PETSc and SLEPc, the dealii installation paths from the previous step is added as follows:

   ```
   dealiiPetscRealDir=$dealii_petscReal_install_dir_path
   ```

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

   ```
   $ bash $invdft_source/install_invdft.sh
   ```

   Please use the full directory path for `$invdft_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:

   ```
   $invDFT_build_dir/release/real/invDFT_exe
   ```

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

   ```
   $ bash $dftfe_source/install_invdft.sh
   ```

   which will create the following debug mode executables:

   ```
   $dftfe_build_dir/debug/real/invDFT_exe
   ```

4 Running invDFT

After compiling invDFT as described in Section 3, we will have the executable in the build directory — `$invDFT_build_dir/release/real/invDFT_exe`. To run the executable in parallel mode, use the command line arguments as follows:

```
  mpirun -n N ./invDFT_exe dftfeParameterFile.prm invDFTParameters.prm
```

to run with N processors.

4.1 Structuring the input file

In the above, the input files have `.prm` as the extension. These files contains input parameters. The parameters detailed in `dftfeParameterFile.prm` corresponds to the parameters required by DFT-FE while `invDFTParameters.prm` corresponds to the parameters required by the invDFT. Generic parameters that are also required for performing the forward DFT calculations are provided in `dftfeParameterFile.prm`. An elaborate description of all available parameters is provided in dftfe-manual, while the parameters that are specific to invDFT are provided in invDFT-manual. Parameters that are specifically required for the inverse DFT calculations are provided in `invDFTParameters.prm`. Section A, provides a detailed description of all the different parameters that can be provided in `invDFTParameters.prm`. The input parameters can be of multiple types. (`string, double, integer, bool etc.`). All input parameters of invDFT are also conveniently indexed at the end of this manual in Section A.3.

The format used to provide inputs for the paramters is common to both the parameter files. 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 Li-H example

Now we will walk you through an example for Li-H in the `/demo/` folder. We note that the demo example in the `/demo/` folder do not cover all the input parameter options. To get full list of input parameters see Section A and the output . All input parameters are also conveniently indexed at the end of this manual in Section A.3.

4.2.1 Li-H inverse calculation

Let us first discuss the inversion of the of the Li-H example. This example is inside `demo/Li_H`, where we perform the inversion of Li-H molecule using CI ground state density. This is an all-electron calculation involving fully non-periodic boundary conditions. For the inversion, we will use the There are two input parameter files– `dftfeParameterFile.prm` and `invDFTParameters.prm`. `dftfeParameterFile.prm` provides the parameters that are required by DFT-FE. The parameters provided as input are required to perform the forward calculation.

To determine the appropriate mesh parameters DFT-FE by doing a total energy convergence study with respect to finite-element mesh discretization. The procedure to set different parameters are as follows.

1. Set the solver mode in order to perform a single ground-state calculation

   ```
   set SOLVER MODE = GS
   set VERBOSITY = 5
   set USE GPU  = true
   ```

   The parameter "VERBOSITY" determins the amount of data that is printed to the output file. The parameter "USE GPU" determines if the arithmetically intensive operations are perfomed on the GPUs.

2. The geometry of the Li-H molecule (LiH) system is set using input parameters under `Geometry` subsection

   ```
   subsection Geometry
     set NATOMS=2
     set NATOM TYPES=2
     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 57.6 a.u as the edge length. Accordingly, the "domainVectors.inp" file is formatted as

     ```
     57.6 0.0 0.0
     0.0 57.6 0.0
     0.0 0.0 57.6
     ```

     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

```
3 3 0.0 0.0 0.000
1 1 0.0 0.0 3.0139242
```

where each line corresponds to “atomic-charge valence-charge x y z”. Since this is an all-electron calculation, the valence-charge must correspond to the atomic-charge.

We require Cartesian coordinates for fully non-periodic simulation domain.

3. 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 but inverse DFT calculations are compatible with only fully non-periodic calculations.

4. Set the required DFT functional input parameters for the ground state calculation

```
subsection DFT functional parameters
  set EXCHANGE CORRELATION TYPE   = LDA-PW
  set PSEUDOPOTENTIAL CALCULATION = false
end
```

where

- The choice of “LDA-PW” for `EXCHANGE CORRELATION TYPE` corresponds to “Perdew-Wang 92 functional [PRB. 45, 13244 (1992)]”.

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

```
subsection SCF parameters
  set MIXING PARAMETER = 0.2
  set TEMPERATURE                      = 500
  set TOLERANCE                        = 1e-8
  set COMPUTE ENERGY EACH ITER = false
  set MIXING METHOD= ANDERSON
  set STARTING WFC             = RANDOM
  subsection Eigen-solver parameters
      set NUMBER OF KOHN-SHAM WAVEFUNCTIONS = 5
      set CHEBYSHEV FILTER TOLERANCE = 1e-10
      set ENABLE HAMILTONIAN TIMES VECTOR OPTIM  = false
  end
end
```

where

- “0.2” set for `MIXING PARAMETER` is the mixing parameter to be used in the mixing scheme.
- “500” set for `TEMPERATURE` is the Fermi-Dirac smearing temperature in Kelvin.

- "1e-8" set for `TOLERANCE` is the SCF stopping tolerance in terms of L2 norm of the electron-density difference between two successive iterations. Such a stringent tolerance is required to get accurate ground state FE density which is required to perform the delta-rho correction.
- "5" 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`.
- "1e-10" set for `CHEBYSHEV FILTER TOLERANCE` is the tolerance to which the chebyshev filter is solved.
- "false" set for `ENABLE HAMILTONIAN TIMES VECTOR OPTIM` as the INVDFT is not compatible with optimised Hamiltonian times vector.

6. DFT-FE and by extension INVDFT as mentioned before employs finite-element basis. These basis are piecewise polynomial functions. The four important FE discretization related parameters in DFT-FE, which the user needs to set are the POLYNOMIAL ORDER, POLYNOMIAL ORDER ELECTROSTATICS, MESH SIZE AT ATOM, MESH SIZE AROUND ATOM, INNER ATOM BALL RADIUS, and ATOM BALL RADIUS.

   First, the POLYNOMIAL ORDER sets the order of the piecewise continuous FE interpolating polynomial that is used in the eigen value problem, with higher values affording faster convergence rates with respect to discretization. For all-electron calculations it is advisable to have POLYNOMIAL ORDER of 4 or 5 and then set a smaller mesh size. POLYNOMIAL ORDER ELECTROSTATICS sets the order of the piecewise continuous FE interpolating polynomial order that is used for the electrostatics problem. For simulations involving heavier elements, it is advisable to set POLYNOMIAL ORDER ELECTROSTATICS to be POLYNOMIAL ORDER + 2.

   Second, the MESH SIZE AT ATOM input parameter sets the size (in Bohr units) of the FE mesh element around the atoms within the distance set by INNER ATOM BALL RADIUS.

   Third, the MESH SIZE AROUND ATOM input parameter sets the size (in Bohr units) of the FE mesh element between INNER ATOM BALL RADIUS and ATOM BALL RADIUS.

```
subsection Finite element mesh parameters
  set POLYNOMIAL ORDER=4
  set POLYNOMIAL ORDER ELECTROSTATICS = 4
  subsection Auto mesh generation parameters
    set MESH SIZE AT ATOM = 0.06
    set MESH SIZE AROUND ATOM = 0.45
    set ATOM BALL RADIUS=6.0
    set INNER ATOM BALL RADIUS = 0.5
  end
end
```

Finally the parameters in `invDFTParameters.prm` are discussed below.

1. Firstly, we set the global parameter "SOLVER MODE" to INVERSE so that inverse DFT calculations.

   Set the solver mode in order to perform a single ground-state calculation.

```
set SOLVER MODE = INVERSE
```

2. The below parameters determine the BFGS algorithm that is used to update the v_{xc}.

```
subsection Inverse DFT parameters
   set TOL FOR BFGS = 1e-12
   set BFGS LINE SEARCH = 1
   set TOL FOR BFGS LINE SEARCH = 1e-6
   set BFGS HISTORY = 100
   set BFGS MAX ITERATIONS = 10000
end
```

3. The below parameters determine the restart capapbilities and the output frequency of the restart files. The files are written into the “FOLDER FOR VXC DATA” folder. If the “FOLDER FOR VXC DATA” folder is not present, it is created.

   ```
   set READ VXC DATA = false
   set POSTFIX TO THE FILENAME FOR READING VXC DATA = vxcData_alpha1_100
   set WRITE VXC DATA = true
   set FOLDER FOR VXC DATA = vxcDataOut
   set POSTFIX TO THE FILENAME FOR WRITING VXC DATA = vxcData_alpha1
   set FREQUENCY FOR WRITING VXC = 40
   ```

4. the v_{xc} is solved in a separate linear FE mesh that is uniform near the atoms. The parameter “VXC MESH SIZE NEAR ATOM” determines the uniform element size, while “VXC MESH DOMAIN SIZE” determines the extent to which the uniform mesh size. To ensure accurate results, the “VXC MESH DOMAIN SIZE” has to be sufficiently large to ensure that the v_{xc} has Dirichlet boundary conditions imposed beyond the uniform mesh. This can lead to small oscillations in the far-field if this is not satisfied. These oscillations will usually be observed at the region where the mesh size changes.

   ```
   set VXC MESH DOMAIN SIZE = 6.0
   set VXC MESH SIZE NEAR ATOM = 0.06
   ```

5. The below parameters provide the names of the Gaussian files that contain the inforation of the CI ground-state energy and the Gaussian density for the delta rho correction. The “GAUSSIAN ATOMIC COORD FILE” contains the coordinates of the atoms in Å.

   ```
   set READ GAUSSIAN DATA AS INPUT = true
   set SET FERMIAMALDI IN THE FAR FIELD AS INPUT = true
   set GAUSSIAN DENSITY FOR PRIMARY RHO SPIN UP = DensityMatrix
   set GAUSSIAN DENSITY FOR PRIMARY RHO SPIN DOWN = none
   set GAUSSIAN DENSITY FOR DFT RHO SPIN UP = DensityMatrixSecondary
   set GAUSSIAN DENSITY FOR DFT RHO SPIN DOWN = none
   set ATOMIC ORBITAL ATOMIC COORD FILE = AtomicCoords
   set GAUSSIAN S MATRIX FILE = SMatrix
   ```

   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 dftfeParameterFile.prm invDFTParameters.prm > outpu
   ```

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

4.2.2 Li-H Post processing

Here in we describe the

1. First, we set the global parameters to Post process mode.

```
    set SOLVER MODE=POST_PROCESS
```

2. Second, we set the filename that need to be read and post processed to enable visualization.

```
subsection Inverse DFT parameters
    set POSTFIX TO THE FILENAME FOR READING VXC DATA = vxcData_alpha1_680
end
```

3. Third, we set the Post processing parameters to interpolate to a .vtu file format, which can later be visualized using Paraview. The “WRITE VTU FILE” parameter determines if the .vtu file is written. The “FILENAME FOR OUTPUT” parameter provides the names of the output files.

```
subsection  POST PROCESS
  set WRITE VTU FILE = true
  set FILENAME FOR OUTPUT=vxcData_alpha1_680_output
end
```

5 Finding answers to more questions

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

- invDFT is primarily based on the DFT-FE. If you have particular questions about DFT-FE, contact the mailing lists described at `https://groups.google.com/g/dftfe-user-group`.
- If you have specific questions about invDFT that are not suitable for public and archived mailing lists, you can contact the primary developers and mentors:
  - Vishal Subramanian: `vishalsu@umich.edu`.
  - Bikash Kanungo: `bikash@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.3.

A.1 Global parameters

- *Parameter name:* `SOLVER MODE`

  Value: INVERSE

  Default: INVERSE

  Description: [Standard] invDFT SOLVER MODE: If INVERSE: performs inverse DFT calculation. If POST_PROCESS: interpolates the Vxc from an input file to a set of points. If FUNCTIONAL_TEST: run a functional test on the adjoint solve.

  Possible values: INVERSE|POST_PROCESS|FUNCTIONAL_TEST

A.2 Parameters in section `POST PROCESS`

- *Parameter name:* `WRITE VTU FILE`

  Value: false

  Default: false

  Description: [Standard] Writes the V_{xc} into a VTU file for visualisation.

  Possible values: A boolean value (true or false)

- *Parameter name:* `INTERPOLATE TO POINTS`

  Value: false

  Default: false

  Description: [Standard] Interpolates V_{xc} to a set of points and writes to a file.

  Possible values: A boolean value (true or false)

- *Parameter name:* `READS POINTS FROM FILE`

  Value: false

  Default: false

  Description: [Standard] if the points to which the V_{xc} has to be interpolated should be read from file.

  Possible values: A boolean value (true or false)

- *Parameter name:* `FILENAME FOR POINTS`

  Value: .

  Default: .

  Description: [Standard] Name of the file from which the points are to be read.

  Possible values: Any string

- *Parameter name:* `FILENAME FOR OUTPUT`

  Value: .

  Default: .

  Description: [Standard] Name of the file to which the interpolated values are written.

  Possible values: Any string

- *Parameter name:* `STARTING X`

  Value: -20

  Default: -20

  Description: [Standard] Starting point for X axis.

  Possible values: A floating point value

- *Parameter name:* `STARTING Y`
  Value: -20
  Default: -20
  Description: [Standard] Starting point for Y axis.
  Possible values: A floating point value
- *Parameter name:* `STARTING Z`
  Value: -20
  Default: -20
  Description: [Standard] Starting point for Z axis.
  Possible values: A floating point value
- *Parameter name:* `ENDING X`
  Value: -20
  Default: -20
  Description: [Standard] ENDING point for X axis.
  Possible values: A floating point value
- *Parameter name:* `ENDING Y`
  Value: -20
  Default: -20
  Description: [Standard] ENDING point for Y axis.
  Possible values: A floating point value
- *Parameter name:* `ENDING Z`
  Value: -20
  Default: -20
  Description: [Standard] ENDING point for Z axis.
  Possible values: A floating point value
- *Parameter name:* `NUMBER OF POINTS ALONG X DIRECTION`
  Value: 1000
  Default: 1000
  Description:[Standard] Number of points along x direction.
  Possible values: An integer n such that $1 \leq n \leq 100000$
- *Parameter name:* `NUMBER OF POINTS ALONG Y DIRECTION`
  Value: 1000
  Default: 1000
  Description:[Standard] Number of points along y direction.
  Possible values: An integer n such that $1 \leq n \leq 100000$
- *Parameter name:* `NUMBER OF POINTS ALONG Z DIRECTION`
  Value: 1000
  Default: 1000
  Description:[Standard] Number of points along z direction.
  Possible values: An integer n such that $1 \leq n \leq 100000$

A.3 Parameters in section `Inverse DFT`

- *Parameter name:* `TOL FOR BFGS`

  Value: 1e-12

  Default: 1e-12

  Description: [Standard] tol for the BFGS solver convergence

  Possible values: A floating point value.

- *Parameter name:* `BFGS LINE SEARCH`

  Value: 1

  Default: 1

  Description: [Standard] Number of times line search is performed before finding the optimal lambda.

  Possible values: An integer n such that $1 \leq n \leq 20$

- *Parameter name:* `NET CHARGE`

  Value: 0

  Default: 0

  Description: [Standard] NET CHARGE ON THE SYSTEM.

  Possible values: An integer n such that $-20 \leq n \leq 20$

- *Parameter name:* `Solve Additional States During Cheb Filtering`

  Value: 0

  Default: 0

  Description: [Standard] Additional states on top of HOMO level that is solved to Cheb tol during chebyshev filtering

  Possible values: An integer n such that $0 \leq n \leq 20$

- *Parameter name:* `TOL FOR BFGS LINE SEARCH`

  Value: 1e-6

  Default: 1e-6

  Description: [Standard] tol for the BFGS solver line search

  Possible values: A floating point value.

- *Parameter name:* `BFGS SOLVER TYPE`

  Value: CP

  Default: CP

  Description: [Standard] The BFGS algorithm used to converge to the exact V_{xc}.

  Possible values: CP|SECANT_FORCE_NORM|SECANT_LOSS

- *Parameter name:* `BFGS HISTORY`

  Value: 100

  Default: 100

  Description: [Standard] Number of times line search is performed before finding the optimal lambda.

  Possible values: An integer n such that $0 \leq n \leq 10000$

- *Parameter name:* `BLOCK SIZE OF INTERPOLATE`

  Value: 10

  Default: 10

  Description: [Standard] Blocksize of the interpolate.

  Possible values: An integer n such that $0 \leq n \leq 1000$

- *Parameter name:* `BFGS MAX ITERATIONS`

  Value: 10000

  Default: 10000

  Description: [Standard] Max number of iterations in BFGS.

  Possible values: An integer n such that $0 \leq n \leq 10000$

- *Parameter name:* `USE DELTA RHO CORRECTION`

  Value: true

  Default: true

  Description:[Standard] Flag to determine if the delta rho correction is to be applied from a file or not.

  Possible values: A boolean value (true or false).

- *Parameter name:* `READ VXC DATA`

  Value: true

  Default: true

  Description:[Standard] Flag to determine if the initial V_{xc} is read from a file or not.

  Possible values: A boolean value (true or false).

- *Parameter name:* `POSTFIX TO THE FILENAME FOR READING VXC DATA`

  Value: .

  Default: .

  Description:[Standard] Post fix added to the filenames from which the V_{xc} data is read.

  Possible values: Any string.

- *Parameter name:* `WRITE VXC DATA`

  Value: true

  Default: true

  Description:[Standard] Write V_{xc} data so that it can be read later.

  Possible values: A boolean value (true and false).

- *Parameter name:* `FOLDER FOR VXC DATA`

  Value: .

  Default: .

  Description: [Standard] Post fix added to the filenames in which the V_{xc} data is written.

  Possible values: Any string value.

- *Parameter name:* `FREQUENCY FOR WRITING VXC`

  Value: 20

  Default: 20

  Description: [Standard] Frequency with which the Vxc data is written to the disk

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

- *Parameter name:* `INITIAL TOL FOR CHEBYSHEV FILTERING`

  Value: 1e-6

  Default: 1e-6

  Description: [Standard] The tolerance to which the chebyshev filtering is solved to initially. The tolerance is progressively made tighteer as the loss decreases.

  Possible values: Any floating point value.

- *Parameter name:* `ADAPTIVE FACTOR FOR ADJOINT`

  Value: 1000.0

  Default: 1000.0

  Description: [Standard] The tol to which the adjoint problem is solved is chosen as the minimum of the tol used for the previous iteration and the tol used for Cheb filtering by adaptive factor.

  Possible values: Any floating point value.

- *Parameter name:* `ADAPTIVE FACTOR FOR CHEBYSHEV FILTERING`

  Value: 100.0

  Default: 100.0

  Description: [Standard] Chebyshev filterting tol is chosen as the minimum of the tol for the preivous iteration and the loss unweighted divided by the adptive factor.

  Possible values: Any floating point value.

- *Parameter name:* `RHO TOL FOR CONSTRAINTS`

  Value: 1e-6

  Default: 1e-6

  Description: [Standard] The tol for rho less than which the initial guess of V_{xc} is not updated.

  Possible values: Any floating point value.

- *Parameter name:* `VXC MESH DOMAIN SIZE`

  Value: 6.0

  Default: 6.0

  Description: [Standard] The distance of the bounding box from the atoms in which the V_{xc} mesh is refined.

  Possible values: Any floating point value.

- *Parameter name:* `VXC MESH SIZE NEAR ATOM`

  Value: 0.3

  Default: 0.3

  Description: [Standard] The mesh size near atom for the V_{xc} mesh.

  Possible values: Any floating point value.

- *Parameter name:* `INITIAL TOL FOR ADJOINT PROBLEM`

  Value: 1e-11

  Default: 1e-11

  Description: [Standard] The initial tol to which the adjoint problem is solved. This tol is adaptively reduced as the iterations proceed based on the loss.

  Possible values: Any floating point value.

- *Parameter name:* `MAX CHEBYSHEV PASSES`

  Value: 100

  Default: 100

  Description: [Standard] The maximum number of chebyshev passes allowed in each inverse iteration.

  Possible values: An integer n such that $10 \leq n \leq 10000$.

- *Parameter name:* `MAX ITERATIONS FOR ADJOINT PROBLEM`

  Value: 5000

  Default: 5000

  Description: [Standard] The maximum number of iterations allowed in MinRes while solving the adjoint problem.

  Possible values: An integer n such that $10 \leq n \leq 10000$.

- *Parameter name:* `ALPHA1 FOR WEIGHTS FOR LOSS FUNCTION`

  Value: 0.0

  Default: 0.0

  Description: [Standard] The parameter used for weight assigned to loss. The weight at a point is assigned based on $\frac{1}{\rho^{\alpha 1}+\tau} + \rho^{\alpha 2}$.

  Possible values: A floating point value.

- *Parameter name:* `ALPHA2 FOR WEIGHTS FOR LOSS FUNCTION`

  Value: 0.0

  Default: 0.0

  Description: [Standard] The parameter used for weight assigned to loss. The weight at a point is assigned based on $\frac{1}{\rho^{\alpha 1}+\tau} + \rho^{\alpha 2}$.

  Possible values: A floating point value.

- *Parameter name:* `FACTOR FOR LDA VXC`

  Value: 0.0

  Default: 0.0

  Description: [Standard] The factor with which the V_{xc} LDA is added to the Hamiltonian. By setting to 1, we compute the delta V_{xc}

  Possible values: A floating point value between 0.0 and 1.0.

- *Parameter name:* `TAU FOR WEIGHTS FOR LOSS FUNCTION`

  Value: 1e-2

  Default: 1e-2

  Description: [Standard] The parameter used for weight assigned to loss. The weight at a point is assigned based on $\frac{1}{\rho^{\alpha}+\tau}$.

  Possible values: A floating point value.

- *Parameter name:* `TAU FOR WEIGHTS FOR SETTING VX BC`

  Value: 1e-2

  Default: 1e-2

  Description: [Standard] The parameter used for weight assigned for V_x. The weight at a point is assigned based on $\frac{\rho}{\rho+\tau}$.

  Possible values: A floating point value.

- *Parameter name:* `TAU FOR WEIGHTS FOR SETTING FABC`

  Value: 1e-2

  Default: 1e-2

  Description: [Standard] The parameter used for weight assigned to set FA BC. The weight at a point is assigned based on $\frac{\rho}{\rho+\tau}$. This parameter is used to transition from V_{xc} to V_{fa} in the far field.

  Possible values: A floating point value.

- *Parameter name:* `TOL FOR FRACTIONAL OCCUPANCY`

  Value: 1e-8

  Default: 1e-8

  Description: [STANDARD] tol for checking fractional occupancy.

  Possible values: A floating point value.

- *Parameter name:* `TOL FOR DEGENERACY`

  Value: 0.002

  Default: 0.002

  Description: [STANDARD] tol for checking fractional occupancy.

  Possible values: A floating point value.

- *Parameter name:* `USE LB94_X IN INITIAL GUESS`

  Value: true

  Default: true

  Description: [Standard] Flag to determine if LB 94_X is used in initial guess. If set to false, then LDA_X is used.

  Possible values: A boolean value (true or false).

- *Parameter name:* `READ FE DENSITY DATA`

  Value: false

  Default: false

  Description: [Standard] Flag to determine if the FE density is read from a file.

  Possible values: A boolean value (true or false).

- *Parameter name:* `READ FE DENSITY DATA WITH SPIN`

  Value: false

  Default: false

  Description: [Standard] Flag to determine if the FE density is read from a file.

  Possible values: A boolean value (true or false).

- *Parameter name:* `FE DENSITY FILENAME`

  Value: .

  Default: .

  Description: [Standard] File name containing the spin polarised FE GS density.

  Possible values: Any string.

- *Parameter name:* `USE MEM OPT FOR TRANSFER`

  Value: false

  Default: false

  Description: [Standard] Flag to determine if the shape func values are stored while doing data transfer.

  Possible values: A boolean value (true or false).

- *Parameter name:* `READ GAUSSIAN DATA AS INPUT`

  Value: true

  Default: true

  Description: [Standard] Flag to determine if the initial V_{xc} is read from a file which is written in gaussian format.

  Possible values: A boolean value (true or false).

- *Parameter name:* `READ SLATER DATA AS INPUT`

  Value: false

  Default: false

  Description: [Standard] Flag to determine if the initial V_{xc} is read from a file which is written in Slater format.

  Possible values: A boolean value (true or false).

- *Parameter name:* `SET FERMIAMALDI IN THE FAR FIELD AS INPUT`

  Value: true

  Default: true

  Description: [Standard] Flag to determine if the initial V_{xc} has fermi-amaldi as the far field in the input.

  Possible values: A boolean value (true or false).

- *Parameter name:* `FACTOR FOR FERMIAMALDI`

  Value: 1.0

  Default: 1.0

  Description: [Standard] Factor for FEMIAMALDI in the far field.

  Possible values: Any floating point value.

- *Parameter name:* `GAUSSIAN DENSITY FOR PRIMARY RHO SPIN UP`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the gaussian code. This is the density for which the V_{xc} is computed. In case of spin un polarised, provide half the total density.

  Possible values: Any string.

- *Parameter name:* `GAUSSIAN DENSITY FOR PRIMARY RHO SPIN DOWN`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the gaussian code. This is the density for which the V_{xc} is computed. In case of spin un polarised, this is not used.

  Possible values: Any string.

- *Parameter name:* `GAUSSIAN DENSITY FOR DFT RHO SPIN UP`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the gaussian code. This density is used for computing the delta rho correction. In case of spin un polarised, provide half the total density.

  Possible values: Any string.

- *Parameter name:* `GAUSSIAN DENSITY FOR DFT RHO SPIN DOWN`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the gaussian code. This density is used for computing the delta rho correction. In case of spin un polarised, this file is not used.

  Possible values: Any string.

- *Parameter name:* `ATOMIC ORBITAL ATOMIC COORD FILE`

  Value: .

  Default: .

  Description: [Standard] File name containing the coordinates of the atoms. These coordinates will be used by the Gaussian/Slater code. This has to compatible with the input coordinates file.

  Possible values: Any string.

- *Parameter name:* `GAUSSIAN S MATRIX FILE`

  Value: .

  Default: .

  Description: [Standard] File containing the S matrix.

  Possible values: Any string.

- *Parameter name:* `SLATER DENSITY FOR PRIMARY RHO SPIN UP`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the SLATER code. This is the density for which the V_{xc} is computed. In case of spin un polarised, provide half the total density.

  Possible values: Any string.

- *Parameter name:* `SLATER DENSITY FOR PRIMARY RHO SPIN DOWN`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the SLATER code. This is the density for which the V_{xc} is computed. In case of spin un polarised, this is not used.

  Possible values: Any string.

- *Parameter name:* `SLATER DENSITY FOR DFT RHO SPIN UP`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the SLATER code. This density is used for computing the delta rho correction. In case of spin un polarised, provide half the total density.

  Possible values: Any string.

- *Parameter name:* `SLATER DENSITY FOR DFT RHO SPIN DOWN`

  Value: .

  Default: .

  Description: [Standard] File name containing the density matrix obtained from the gaussian code. This density is used for computing the delta rho correction. In case of spin un polarised, this file is not used.

  Possible values: Any string.

- *Parameter name:* `SLATER S MATRIX FILE`

  Value: .

  Default: .

  Description: [Standard] File containing the S matrix.

  Possible values: Any string.

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.