

Compiling and testing DFTB+

Bálint Aradi

21st September 2007

Abstract

This document contains detailed instructions about the compilation and testing of the DFTB+ code.

Contents

1 Prerequisites	1
2 Architecture dependent settings	2
3 Further customisation	3
4 Compiling the code	3
5 Testing the binary	5

1 Prerequisites

In order to compile the code, the following components must be present on your system:

DFTB+ source The source code can be downloaded from <http://www.dftb-plus.info>.

BLAS/LAPACK DFTB+ uses BLAS and LAPACK calls for its matrix manipulation operations. You need BLAS and LAPACK compatible libraries on you system.

Fortran 95 compiler You need a decent Fortran 95 compiler. DFTB+ relies pretty much on advanced Fortran 95 constructs. Some old compilers, not fully implementing the Fortran 95 standard, fail to compile the DFTB+. The compilers, for which a makefile is provided, should work. (Check out the release notes for the current release you are trying to compile.)

GNU Make In order to process the makefiles, you need GNU Make (version $\geq 3.79.1$). If there is no GNU Make on your system, you should download it from the [GNU site](#) and compile it by yourself (which is pretty straightforward).

cpp The Fortran 95 source files of DFTB+ are preprocessed with the standard C preprocessor.

awk The preprocessed files are postprocessed by the standard Unix tool awk.

(Python) Python is not needed for the compilation of the code, but for the testing of the resulting binary with the autotest sytem. You need Python version ≥ 2.3 for that.

2 Architecture dependent settings

First you have to unpack the downloaded source. For simplicity in this howto we assume that you unpacked the source in /tmp:

```
/tmp> tar xvfz ~/download/dftb-plus_1.0_src.tar.gz  
/tmp> cd dftb+_1.0_src/
```

At first, you'll have to configure the make system to match the needs for your architecture and compiler. The system specific makefiles can be found in the subdirectory `sysmakes/`. If you don't find a makefile appropriate for your system, create your own by using `make.generic` as template. You should check and eventually modify the following variables in the system specific makefile:

FC90 Compiler to use. Despite the variable name, you have to provide a *Fortran 95* compiler.

FC90OPT Options to pass to the Fortran 95 compiler. You should set and tune the optimisation switches here.

CPP Name of the C preprocessor (usually `cpp`).

CPPOPT Options to pass to the C preprocessor. On some architectures the C preprocessor needs extra options in order to process files containing a non-C source code (e.g. the `-traditional` switch on linux). Additionally to these options, the debug level for the compilation must be passed to the preprocessor as `-DDEBUG=$(DEBUG)`.

CPPPOST Utility to use for postprocessing the files coming out from `cpp` (e.g. making sure, lines are not longer than 132 characters, removing debug info created by `cpp`, etc). You should set this variable to

```
$(ROOT)/utils/fpp/fpp.sh <method>
```

where `<method>` can be one of the following keywords (depending on your compiler):

general The `#line` directives in the output file are kept. (Very convenient when debugging, but not all compilers can process source files with such directives.)

nocntln `#line` directives removed from continuation lines.

noln Does not add any `#line` directives when substituting CPP macros.

noln2 Removes any `#line` directives occurring in the preprocessed source.

LN Linker to use for linking the object files. (It is usually set to the same value as `FC90`.)

LNOPT Options to pass to the linker (e.g. `-static` if you want to create a statically linked binary).

LIBOPT Library related options for the linker. You should specify here the location of your LAPACK and BLAS libraries if they are not in the search path of the linker. If those libraries are for example in `/usr/local/lib`, you should set

```
LIBOPT = -L/usr/local/lib
```

LIB_BLAS Linker option for link blas (e.g. `-llapack`)

LIB_LAPACK Linker option for linking lapack (e.g. `-lblas`).

Next you have to copy in the root of the source directory the file `Makefile.user.template` to `Makefile.user` and set the variable `$(ARCH)` to your architecture (to the suffix of your makefile in the subdirectory `sysmakes/`). For example for `linux-i686-ifort` you would have to set

```
# Set your architecture. Make sure, that 'sysmakes/make.$(ARCH)' exists for
# you architecture. If not, create your own using 'sysmakes/make.generic'
# as template.
ARCH = i686-linux-ifort
```

3 Further customisation

You can do further customisation by overriding any variables used by the make system in `Makefile.user`. The following variables could be of interest: (The variable `$(ROOT)` points to the root of the DFTB+ source directory.)

OBJDIR_PARENT Directory, in which the subdirectory containing the object files and the compiled binary should be created.

Per default it is set to the directory, where you issue the make command (working directory).

OBJDIR_SUFFIX Suffix to append in the directory name for the object files after `_obj`.

Per default it is set to `_${ARCH}`.

INSTALLDIR The directory, where to copy the DFTB+ binary, when issuing `make install`.

Defaults to the working directory.

PRGDFTB_TESTDIR Location of the DFTB+ autotest suite (see next section). The autotest suite must be download separately from the code. It is not needed for the compilation, only for testing the resulting binary.

The default is `$(ROOT)/../autotest`.

PRGDFTB_TESTFILE File containing the tests to calculate.

Default: `$(PRGDFTB_TESTDIR)/tests`.

4 Compiling the code

After having customised the make system for your architecture, enter the `prg_dftb/` directory. First you should make sure, that you don't have any remnants of previous compilations, so type

```
/tmp/dftb+_1.0/src/prg_dftb> make distclean
```

Then invoke `make` without any arguments. You should see something like that:

```
aradi@core07:/tmp/dftb+_1.0_src/src/prg_dftb> make
make[1]: Entering directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'
echo "" > _dependencies

for dep in ../../lib_common/Makefile.dep ../../lib_dftb/Makefile.dep ../../lib_g
eoopt/Makefile.dep ../../lib_io/Makefile.dep ../../lib_math/Makefile.dep ../../l
ib_md/Makefile.dep ../../lib_mixer/Makefile.dep ../../lib_type/Makefile.dep ../.
./includes/Makefile.dep ../../ext_xmlf90/Makefile.dep ../Makefile.dep ../../lib_
math/Makefile.local; do cpp -traditional -DDEBUG=0 $dep >> _dependencies; done
make[1]: Leaving directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'
make[1]: Entering directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'
cpp -traditional -DDEBUG=0 -I../../lib_common/ -I../../includes ../../lib_common
/allocate.F90 | ../../utils/fpp/fpp.sh nocntln > allocate.f90
ifort -O2 -xW -ip -o allocate.o -c allocate.f90
touch _mod_allocation
cpp -traditional -DDEBUG=0 -I../../lib_common/ -I../../includes ../../lib_common
/accuracy.F90 | ../../utils/fpp/fpp.sh nocntln > accuracy.f90
ifort -O2 -xW -ip -o accuracy.o -c accuracy.f90
touch _mod_accuracy
.
.
.
cpp -traditional -DDEBUG=0 -I../ -I../../includes ../dftb+.F90 | ../../utils/fpp
/fpp.sh nocntln > dftb+.f90
ifort -O2 -xW -ip -o dftb+.o -c dftb+.f90
ifort -O2 -xW -ip -i-static -o dftb+ dftb+.o geoopt.o allocate.o stepdesc.o accu
racy.o conjgrad.o linmin.o constants.o charmanip.o initprogram.o rcm.o sort.o pe
riodic.o lapackroutines.o message.o bisection.o linkedlist.o spin.o simple_algeb
ra.o diis_mixer.o scc.o fileid.o coulomb.o blasroutines.o extcharge.o short_gamm
a.o dummy_thermostat.o md_common.o ranlux.o orbital_equiv.o simple_mixer.o slate
r_kirkwood.o thermostat.o andersen_thermostat.o temp_profile.o broyden_mixer.o f
ifo.o md_integrator.o velocity_verlet.o inputdata_.o type_geometry.o sk_interpol
ate.o interpolation.o intrinsicpr.o mixer_adt.o anderson_mixer.o scc_init.o flib
_wxml.o m_wxml_core.o m_wxml_elstack.o m_wxml_buffer.o m_wxml_dictionary.o m_wxm
l_text.o nonscc.o sk.o populations.o energies.o formatout.o eigenvects.o eigenso
lver.o parser.o oldcompat.o hsdutils2.o hsdparser.o xmlutils.o m_strings.o flib_
dom.o m_dom_nodelist.o m_dom_types.o m_dom_utils.o m_dom_document.o m_dom_namedn
odemap.o m_dom_node.o m_dom_error.o m_dom_debug.o m_dom_element.o m_dom_attribut
```

```

e.o m_dom_parse.o flib_sax.o m_xml_parser.o m_elstack.o m_buffer.o m_fsm.o m_entities.o m_dictionary.o m_charset.o m_debug.o m_reader.o m_io.o m_xml_error.o m_converters.o hsdutils.o tokenreader.o stringlist.o unitconversion.o type_geometry_hsd.o taggedout.o densitymatrix.o etemp.o hermite.o ext_ercf.o fact.o repulsive.o forces.o dftb_pls_u.o -L/usr/local/i686.Linux/lib -lmkl_lapack -lmkl_ia32 /usr/local/i686.Linux/lib/libguide.a -lpthread
rm anderson_mixer.f90 xmlutils.f90 short_gamma.f90 thermostat.f90 geoopt.f90 stepdesc.f90 spin.f90 intrinsicpr.f90 type_geometry.f90 formatout.f90 mixer_adt.f90 dummy_thermostat.f90 inputdata_.f90 hsdparser.f90 fifo.f90 populations.f90 dftb_pls_u.f90 forces.f90 sort.f90 dftb+.f90 md_common.f90 andersen_thermostat.f90 densitymatrix.f90 eigensolver.f90 extcharge.f90 conjgrad.f90 hsdutils.f90 periodic.f90 broyden_mixer.f90 unitconversion.f90 parser.f90 linmin.f90 fact.f90 orbital_equiv.f90 fileid.f90 sk.f90 nonscc.f90 ext_ercf.f90 linkedlist.f90 diis_mixer.f90 repulsive.f90 accuracy.f90 tokenreader.f90 temp_profile.f90 hermite.f90 stringlist.f90 lapackroutines.f90 md_integrator.f90 charmanip.f90 eigenvects.f90 type_geometry_hsd.f90 energies.f90 velocity_verlet.f90 scc.f90 initprogram.f90 sk_interpolate.f90 bisection.f90 constants.f90 slater_kirkwood.f90 message.f90 taggedout.f90 hsdutils2.f90 oldcompat.f90 scc_init.f90 rcm.f90 coulomb.f90 simple_algebra.f90 blasroutines.f90 etemp.f90 simple_mixer.f90 allocate.f90 interpolation.f90 ranlux.f90
make[1]: Leaving directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'

```

The resulting binary `dftb+` can be found in the directory `$(OBJDIR_PARENT)/_obj$(OBJDIR_SUFFIX)`. If you did not change those variables, you find it below `prg_dftb/_obj_$(ARCH)`, where `$(ARCH)` contains the value, you set in `Makefile.user`. By issuing

```
/tmp/dftb+_1.0/src/prg_dftb> make install
```

the binary is copied to the current directory (`prg_dftb`) or to the place specified with the variable `$(INSTALLDIR)` in `Makefile.user`.

5 Testing the binary

After compiling, you should test, if your binary delivers the right results. For that you have to download the autotest suite from dftb-plus.info. You should extract it somewhere and for convenience rename it to `autotest`. Let's assume you do that in the `/tmp/` directory again:

```

aradi@core07:/tmp> tar xzf ~/download/dftb-plus_1.0_autotest.tar.gz
aradi@core07:/tmp> mv dftb+_1.0_autotest autotest

```

In order to calculate the downloaded set of systems, you have to provide Slater-Koster files to the autotest system. These *must* be exactly the same set that were used to precalculate those systems. Please check the README file of the autotest suite for the list of needed parametrisation sets. You can download those sets from dftb.org. You should extract those sets in the `slako` subdirectory of the autotest system. Let's assume, the README says, that you need the

parametrisation sets mio-0-1, pbc-0-1 and hyb-0-1, then after downloading you would have to extract as

```
/tmp/autotest/slako> tar xzf ~/download/mio-0-1.tar.gz
/tmp/autotest/slako> tar xzf ~/download/pbc-0-1.tar.gz
/tmp/autotest/slako> tar xzf ~/download/hyb-0-1.tar.gz
```

Now, you should be able to test your system, *provided you have Python 2.3 (or newer) on your system*. Change to the prg_dftb directory of the DFTB+ source and issue make test.

```
aradi@core07:/tmp/dftb+_1.0/src/prg_dftb> make test
make[1]: Entering directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'
non-scc/Si_2: preparing .. running .. comparing .. Match.
non-scc/GaAs_2: preparing .. running .. comparing .. Match.
non-scc/HBDI-neutral: preparing .. running .. comparing .. Match.
non-scc/HBDI-cationic: preparing .. running .. comparing .. Match.
non-scc/decapentaene: preparing .. running .. comparing .. Match.
non-scc/10-0Ctube: preparing .. running .. comparing .. Match.
non-scc/10-10Ctube: preparing .. running .. comparing .. Match.
```

TEST SUMMARY

Match:

non-scc/Si_2	spin/H2O
non-scc/GaAs_2	spin/H2O-periodic
non-scc/HBDI-neutral	spin/GaAs_2
non-scc/HBDI-cationic	spin/H2
non-scc/decapentaene	geoopt/H2O
non-scc/10-0Ctube	geoopt/H2O-constr
non-scc/10-10Ctube	geoopt/H2O-nonscc
non-scc/Si41C23N35	geoopt/Vsi+O-nonscc
non-scc/Si_384	geoopt/Vsi+O
non-scc/Si_216	md/Si_8
scc/GaAs_2	md/H3
scc/SiC_64	md/Si_8-thermostat
scc/C60	md/Si_8-thermostat2
scc/H3	md/Si_8-tempprofile
scc/H2O-extchrg	dispersion/2H2O
scc/H2O-extchrg-periodic	dispersion/DNA
scc/H2O-extchrg-blur	

Status: OK

Details in:

__autotest/stderror.log
__autotest/tagdiff.log

```
=====  
make[1]: Leaving directory '/tmp/dftb+_1.0/src/prg_dftb/_obj_i686-linux-ifort'  
aradi@core07:/tmp/dftb+_1.0/src/prg_dftb>
```

The input and output for the different calculations can be found in the __autotest directory below the directory for the object files. The file __autotest/tagdiff.log summarises the comparison between the calculated and the precalculated results. __autotest/stderror.log contains the messages written to stderr during the run and some timing information.

Should any of the tests fail, you should try to recompile the code after switching off optimisation options in the FC90OPT variable of the system specific makefile. If you still experience failing tests, try to get into contact with the developers.