^{4}HeDFT BCN-TLS — User Manual

François Coppens

June 15, 2017

Contents

1 Prerequisites

1.1 General

- An internet connection
- a recent version of the 'Git'-program (https://git-scm.com/downloads)
- the Intel MKL/FFT– or other Fast Fourier Transform library
- a recent version of the Intel Fortran compiler (iFort)

1.2 CECAM DFT School

1.2.1 UNIX/Linux

If you have a recent UNIX/Linux distribution installed on your computer, everything you need should already be there.

1.2.2 Windows

In principle, only a good terminal emulator is needed that supports 'X11-forwarding'. Something like 'PuTTY' will work fine. You can download it here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

2 General notes on obtaining and compiling the code

A step-by-step procedure is given in Section 3.1.1

There are four main programs, each in its own repository on the GitHub website. They are respectively:

1. Static ^{4}He DFT, supporting classical– and quantum impurities

2. Static ^{4}He DFT, supporting vortical droplet states, classical– and quantum impurities

3. Time-dependent ^{4}He DFT for classical atomic impurities in an isotropic electronic state (e.g. s-states)

4. Time-dependent ^{4}He DFT for classical atomic impurities in an anisotropic electronic state (e.g. p,d-states)

To obtain the code, only one command in a terminal window is required. Change to the directory where you want the top level directory of the code to reside and execute

```
$ git clone <GitHub URL>
```

where `<GitHub URL>` needs to be substituted by on of the following URLs:

1. `https://github.com/bcntls2016/4hedft.git`
2. `https://github.com/bcntls2016/4hedft-vortex.git`
3. `https://github.com/bcntls2016/4hetddft-isotropic.git`
4. `https://github.com/bcntls2016/4hetddft-anisotropic.git`

After git has downloaded the repository from the Internet, change to the repository directory which should be named after the last part of the URL without the `.git`-part, and create a git 'work'-branch

```
$ git checkout -b work
```

This will create a new git branch called 'work' in which we can work, whilst not contaminating the 'master'-branch and prevents merge conflicts in the 'master'-branch during updates.

The next step is to create a makefile and tailor it to your machines architecture. You can use the provided 'makefile' called `makefile`. It's are optimized for the Intel AVX architecture, with some extra machine specific optimisations for EOS' architecture. You can use it as-is or as a template for your own one. Beware that if you use your own FFT-library, make sure that they are locatable by the 'make'-program. If you use Intel's MKL library you can use the environment variable `$MKLROOT` (see include makefile as an example). Start the compilation by executing

```
$ make && make clean
```

After there are no errors one of the {`4hedft, 4hedft-vortex, 4hetddft-isotropic, 4hetddft-anisotropic`} executables should be available in the code directory, unless you changed the program name in your makefile.

Make sure that all your changes are staged and committed otherwise your own customizations could be overwritten after a '`git pull`'*!*

3 Static ^{4}HeDFT

3.1 First calculation: A pure $^4\mathrm{He}_N$ droplet

The easiest calculation to do is a pure $^4\mathrm{He}_N$ droplet. A typical droplet size used to study things like desorption dynamics and collisions is about $N = 1000$ atoms and is ∼44 Å in diameter. It is large enough so that it exhibits the desired physical properties and at the same time small enough to keep it computationally feasable. This is why we shall start with a $^4\mathrm{He}_{1000}$ droplet.

The program will setup a simulation grid in cartesian coordinates. The helium density will be determined at each point of the grid, which should be large enough to contain the droplet. The mesh size should be ∼0.4 Å

But before we do that we need to pick and compile the needed program. In this case we need the static DFT program (program 1. in the list).

3.1.1 Setting up and compiling the code

We first need to logon to the EOS supercomputer. This is done using the 'ssh' program on Linux/UNIX, or with, e.g., 'PuTTY' on Windows. From the Linux command line simply execute

```
$ ssh <cecam user ID>@eos.calmip.univ-toulouse.fr
```

After you have logged into EOS over SSH through either your linux terminal or PuTTY you should be in your home directory. If you're not sure or you got lost, simply execute the command cd without any arguments. We will first setup the correct compiler suite (module), then we create a directory called 'code' where our programs will reside. Finally we will compile the code into an executable program.

```
$ module purge
$ module load intel/17.0.4
$ mkdir code
$ cd code
$ git clone https://github.com/bcntls2016/4hedft.git
$ cd 4hedft
$ make && make clean
```

3.1.2 Preparing the calculation directory

It is good practice to create a separate directory for each calculation to keep all the input– and output files together. Firt go back to your home directory as explained in the previous section. Once you are there, we are going to create a directory called 'static' that will contain all the static calculations. Then inside that one the directory 'pure-4he1000' will be created to host your first calculation.

```
$ mkdir static
$ cd static
$ mkdir pure-4he1000
$ cd pure-4he1000
```

Inside the code directory there is a folder named 'input_files' which contains the input parameters for the calculation (pure-4he1000.input) and a script to submit the calculation to the computer cluster (pure-4he1000.sbatch). We copy these files into our calculation directory and make a 'symbolic link' to the program:

```
$ cp ../../code/4hedft/input_files/pure-4he1000.input .
$ cp ../../code/4hedft/input_files/pure-4he1000.sbatch .
$ ln -s ../../code/4hedft/4hedft .
```

Before we submit our first calculation you have to be familiar with the input variables in the input-file.

3.1.3 The minimal set of input parameters

Here follows a description of a minimal set of paramaters that need to be setup for a successful calculation. There are more parameters than there are in this file. The full set of parameters that can be controled will be given and explained later.

General information

title	the title of the calculation
nthread	number of OpenMP threads to use
mode	0 – continue a calculation while reading all the input parameters from the start-density 1 – build, from scratch, a density for a pure 4He$_N$ droplet 2 – build, from scratch, a density for a 4He$_N$ droplet and a *quantum* impurity 3 – build, from scratch, a density for a 4He$_N$ droplet and a *classical* impurity 4 – continue a calculation but ignore the impurity's position in the start-density and use the position given in the input file/impurity-wave-function. Can be used to find e.g. ❇ the ground state of a droplet with an impurity (*classical only*), starting from a pure droplet, or ❇ the ground state of a displaced impurity (*classical/quantum*) under a constraint, starting from an unconstrained ground state

Grid

{x,y,z}max	the grid goes from -{x,y,z}max to {x,y,z}max so the size is 2×{x,y,z}max
n{x,y,z}	number of grid points in the {x,y,z}-direction

Droplet properties

n4	number of ^{4}He atoms
{x,y,z}c	droplet Center-of-Mass position
core4	the used functional for the ^{4}He density. This field is ignored if 'lsolid' is set to '.true.'
lsolid	switch to enable/disable the use of the 'Solid Functional'
limp	treat the impurity classically or quantum
lexternal	treat the impurity as an external field

Output control

pener	print an energy overview every $n \in \mathbb{N}$ iterations
pdenpar	print a partially converged density every $n \in \mathbb{N}$ iterations
filedenout	filename of the final converged density

Computational parameters

niter	total number of $n \in \mathbb{N}$ iterations. When n is reached, the program is stopped, even if the density is not converged
precie	precision $0 < r \in \mathbb{R} < 1$ of the error in the total energy
pafl	imaginary timestep size $\tau \in \mathbb{R}_{>0}$
lpaflv	switch to enable/disable gradual increase in imaginary timestep size
nstepp	number of imaginary timestep size increments
/	
1000 0.0001	use an imaginary timestep of 0.0001 for the first 1000 iterations
2000 0.0005	an imaginary timestep of 0.0005 for the next 2000
3000 0.0010	an imaginary timestep of 0.001 for the rest

3.1.4 Execution and output files

To start the actual calculation and assuming the name of the symbolic link you made earlier carries the same name as the executable one executes in the calculation directory

```
$ ./BCN4HeDFT < pure-4he1000.input > pure-4he1000.res 2> error.log &
```

The standard output of the executable itself will be written to the file `pure-4he1000.res` and the errors to `error.log`. The '`&`' means that the program will be executed in the background so we keep our terminal free to use for other things.

The program will now start to write some output files which will be explained in the following table

Output files

DFT4He3d.namelist.read	a complete list of all the input variables, including the ones that were not explicitly given in the input file
den-{x,y,z}.dat	density profile in the {x,y,z}-direction at the time of the last written partial density
den.out	the converged output density (if not changed in 'filedenout')
den0-{x,y,z}.dat	density profile in the {x,y,z}-direction at the start of the calculation
density.{iteration#}.out	partially converged density at time of imaginary timestep increment
partial.density{1,2,...}	partially converged density at {1,2,...}×pdenpar
pure-4he1000.error	file containing the messages to *stderr* generated by the program
pure-4he1000.res	file containing the messages to *stdout* generated by the program

3.2 A ^{4}He$_{1000}$ droplet with a classic impurity

Building the ground-state density for a droplet with a classic impurity can be done in two ways. Building the whole complex from scratch, or using an exsiting droplet and adding the impurity. The easiest method is the first, while the fastest methode is the second one; when adding the impurity to a converged pure droplet, it only needs to adapt locally, as long as the initial choice of the impurity's location is not choosen too poorly.

For each of the two cases, the following parameters need to be changed and added

`lexternal = .false.` $\Longrightarrow$ `lexternal = .true.`,

and the following six new parameters in the input file need to be set.

Added parameters

mimpur	impurity mass in unified atomic mass units (u)
rimpur	radius of the impurity. Usually the distance to the bottom of the potential well in the ^4He-X pair potential is a good choice
{x,y,z}imp	location of the center of mass of the impurity relative to the calculation grid
selec_gs	the ground state pair interaction potential to use. These can be looked-up in the file 'V_impur.f90' in the code repository
r_cutoff_gs	cut-off radius for the pair interaction potential
umax_gs	cut-off value for the pair interaction potential

As a consequence, there are also six new output files containing information about the interaction between the impurity and the droplet

Output files

uext-{x,y,z}.dat	**<put description here>**
uext-{xy,yz,xz}.dat	**<put description here>**

3.2.1 Building the complex from scratch

To build the X-^{4}He$_N$ complex we now only need to change one parameter

`mode = 1` $\Longrightarrow$ `mode = 3`

You will find an input file that is already prepared for this in the `input_files` directory called '`ck-4he1000-fs.input`'. Now setup your calculation directory like explained in section 3.1.2 and start the calculation by executing

```
$ ./BCN4HeDFT < ck-4he1000-fs.input > ck-4he1000-fs.res 2> error.log &
```

3.2.2 Building the complex from a converged pure droplet

To build the X-^{4}He$_N$ complex we need to change the parameter

`mode = 1` $\Longrightarrow$ `mode = 4`, and add one extra parameter to our input-file

	Added parameter
filedenin	filename of the converged input density of a pure droplet

Again, prepare your calculation directory like before, with one extra edition. We need either copy an input density to the calculation directory, make a symbolic link to one, or give the full relative path to one in the input file. Making a copy of an exising one is preferable since you have everything you need in one directory for future refenrence.

Be carful that the box size and number of points of the input density correspond to the box size and number of points in the input file.

An already prepared input file for this in the `input_files` directory called '`ck-4he1000-fd.input`'.

Start the calculation as before:

```
$ ./BCN4HeDFT < ck-4he1000-fd.input > ck-4he1000-fd.res 2> error.log &
```

3.3 A ^{4}He$_{1000}$ droplet with a classic impurity under a constraint

To impose a given separation between the center of mass of the droplet and the impurity, we need to do the calculation under a constraint. For a constrained calculation we advise to start with the ground state of a droplet containing an impurity at its equilibrium position, otherwise the calculation will take a long time to converge.

We need to add three new parameters to out input file.

	Added parameters
lconstraint	enables the constraint energy penalty term $\frac{1}{2} \times Intens \times (z_{imp} - < z_{He} > - z_{dist})$ (true/false)
intens	intensity of the penalty term
zdist	constrained distance of the impurity to the center of mass of the droplet in Å. Adjust 'zimp' and 'zdist' such that $z_{imp} - < z_{He} > - z_{dist} = 0$' is satisfied

An example input file is provided called `ck-constrained-4he1000.input` is provided in the usual location.

3.4 A ^{4}He$_{1000}$ droplet with a quantum impurity

When the mass of the impurity is getting too small for its position to be accurately calculated classically, we need to solve its Schrodinger equation.

In this case there is no possibility to start from an already converged pure droplet. In this case, the only three things that change with respect to the case described in section 3.2.1

```
          mode = 3    ⟹   mode = 2
     limp = .false.   ⟹   limp = .true.
lexternal = .true.    ⟹   lexternal = false
```

and add the parameter

Added parameters

fileimpout	impurity density output filename

to save the impurity wavefunction to disk.

3.5 A ^{4}He$_{1000}$ droplet with a quantum impurity under a constraint

This can be done in mode = 0 and mode = 4, since the position is calculated from the wave function, and then shifted. Then both wavefunctions are relaxed in eachothers mean field. For sake of consitancy and clarity we recommend to use mode = 4 to not confuse it with a calculation that is just a continuation of the previous one.

Compared to the calculation done in section 3.3 we need to change

```
     limp = .false.   ⟹   limp = .true.
lexternal = .true.    ⟹   lexternal = false
```

to indicate we want to treat the impurity quantal, and add the parameters

Added parameters

lconstraint_imp	indicate that the impurity wavefunction should be relaxed under a constraint
fileimpin	impurity density input filename
fileimpout	impurity density output filename

3.6 A pure ^{4}He$_{1000}$ droplet with line vortex

To create droplets hosting one or more vortices, with or without the presence of an impurity, and obtain their ground states we need an extendedversion of the static code. As before, to obtain and download the code we execute

```
$ git clone https://github.com/bcntls2016/4hedft-vortex.git
$ cd 4hedft4hedft-vortex
$ git checkout -b work
<customise or create makefile to you needs>
$ make && make clean
```

To specify the properties of the vorex, we need to add four new entries in our main input file to enable the creation of droplets hosting vortices. Also an extra imput file has to be provided to specify the location(s) of the vort(ex/ices)

4 Dynamic ^{4}HeDFT

4.1 Impurities in a spherically symmetric electronic state

4.2 Impurities in a anisotropic electronic state (p,d-states)