^{4}HeDFT BCN-TLS — User Manual

François Coppens

June 27, 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 to download and compile the code 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 optimised 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 customisations could be overwritten after a '`git pull`'!

3 Static ^{4}HeDFT

3.1 First calculation: A pure ^4He_N droplet

The easiest calculation to do is a pure ^4He_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 feasible. This is why we shall start with a $^4He_{1000}$ 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
```

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. First 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 controlled will be given and explained later.

General information

title	the title of the calculation
nthread	number of OpenMP threads (CPU's) to use
mode	0 – continue a previous calculation while reading all the input parameters from the density. An input density should be provided. This allows to stop and restart a calculation 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 Modes {2,3,4} are not relevant for a pure droplet but only mentioned here for completeness.

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. Typically ∼40 Å
n{x,y,z}	number of grid points in the {x,y,z}-direction. Typically ∼200. The number needs to be a 'magical number', dictated by the Fast Fourier library. Ideally $n = 2^k$, or $n = 2^k + 2^l$, $l < k$.

Droplet properties

n4	number of ^{4}He atoms. Normally $n4 = 1000$
{x,y,z}c	droplet Center-of-Mass position. Usually $(0, 0, 0)$ unless the problem requires something else.
core4	the used ^{4}He-functional. This field is set to 'OT' automatically if 'lsolid' is set to '.true.' OT This is the default, no α_S term in the Orsay-Trento functional[1] OTC Full Orsay-Trento functional[1] OP Orsay-Paris functional[2]
lsolid	switch to *.true./.false.* to enable/disable the use of the Solid Functional[3]. As said above already. When enabled, 'OT' is used regardless the value of *core4*
limp	treat the impurity classically or quantum
lexternal	treat the classical impurity as an external field

Output control

pener	print an energy overview every *pener* $\in \mathbb{N}$ iterations
pdenpar	print a partially converged density every *pdenpar* $\in \mathbb{N}$ 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 of the convergence. $0 < precie \in \mathbb{R} \ll 1$ is the size of the error in the total energy of the helium. The calculation stops when the size of the error in the total energy becomes less or equal to this value
pafl	imaginary timestep size $\tau \in \mathbb{R}_{>0}$. Usually $\tau \sim 0.05$
lpaflv	switch to allow for the imaginary time-step size to change during the calculation
nstepp	number of imaginary time-step changes
/	
1000 0.0001	use an imaginary time-step of 0.0001 for the first 1000 iterations
2000 0.0005	an imaginary time-step of 0.0005 for the next 2000
3000 0.0010	an imaginary time-step of 0.001 for the rest

[1]F. Dalfovo and A. Lastri *et al*, Phys. Rev. B **52**, 1193 (1995)
[2]J. Dupont-Roc *et al*, J. of Low Temp. Phys. Vol. **81**, 1, 31–44 (1990)
[3]F. Ancilotto, M. Barranco *et al*, Phys. Rev. B **72**, 214522 (2005)

3.1.4 Execution and output files

We can start the calculation in two ways. The first way starts the calculation on the current machine.

```
$ ./4hedft < pure-4he1000.input > pure-4he1000.res 2> pure-4he1000.err &
```

The '`&`' means that the program will be executed in the background so we keep our terminal free to use for other things.

The second way is to use a submission script that uses the software installed on the computer cluster to schedule the job and execute it on a free computation node. The script is provided with the code and you already copied it in Section 3.1.2: `pure-4he1000.sbatch`. To submit the job execute

```
$ sbatch pure-4he1000.sbatch
```

You can view the status of your job by invoking

```
$ squeue -u <cecam user ID>
```

The standard output of the executable itself will be written to the file `pure-4he1000.res` and the errors to `pure-4he1000.err`.

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	1-dimensional density profile in the {x,y,z}-direction at the time of the last written partial density
den.out (or 'filedenout')	the converged output density file
den0-{x,y,z}.dat	1-dimensional 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

To check if the calculation is running okay we need to check if the error in the total energy of the system is decreasing. This can be done by looking into the `pure-4he1000.res` file, looking for lines like

```
ITERATIVE PROCEDURE
Total Energy (He).........    -5400.344271 K  +/-  -1.7363E-06 K
```

This is printed every `pener` iterations. As long as the error is steadily decreasing, the calculation is running fine. The calculation will terminate whenever this value becomes smaller or equal to whatever value `precie` is set to.

3.2 A droplet with a classical impurity $X{-}^4\mathrm{He}_N$

Building the ground-state density for a droplet with a classical impurity can be done in two ways. Building the whole system from scratch, or using an existing droplet and adding the impurity. The easiest method is the first one, 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 location is not chosen too poorly.

For either case, 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). E.g, potassium is ∼39 u
rimpur	radius of the impurity. Usually the distance to the bottom of the potential well in the $X{-}^4$He pair potential is a good choice, about 3 − 4 Å
{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, usually set such that it corresponds to an energy value slightly higher than the maximum energy deposited into the system
umax_gs	cut-off value for the pair interaction potential (see r_cutoff_gs)

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	1-dimensional cuts of the external potential (the impurity) and the helium at the start of the calculation
uext-{xy,yz,xz}.dat	same as uext-{x,y,z}.dat, but 2-dimensional

3.2.1 Building a $X{-}^4\mathrm{He}_N$ system from scratch

To build a $X{-}^4\mathrm{He}_N$ system 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-from_scratch.input`'. Now setup your calculation directory like explained in section 3.1.2 and start the calculation by executing

```
$ cd
$ cd static
$ mkdir ck-4he1000-fs
$ cd ck-4he1000-fs
$ cp ../../code/4hedft/input_files/ck-4he1000-from_scratch.input .
$ cp ../../code/4hedft/input_files/ck-4he1000-from_scratch.sbatch .
$ ln -s ../../code/4hedft/4hedft 4hedft
$ sbatch ck-4he1000-from_scratch.sbatch
```

Check the status of the job as before

```
$ squeue -u <cecam user ID>
```

and monitor the calculation itself by evaluating the output in `ck-4he1000-from_scratch.input`

```
$ tail -f ck-4he1000-from_scratch.input
```

3.2.2 Building a $X-{}^4\mathbf{He}_N$ system from a converged pure ${}^4\mathbf{He}_N$ droplet

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

$$\texttt{mode = 1} \implies \texttt{mode = 4},$$

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 addition. We need provide the file containing a converged density of a pure ${}^4\mathrm{He}_N$ droplet. This can be done in 3 ways:

i) copy the density into the calculation directory,

ii) make a symbolic link,

iii) give the full path in the input file.

We will use option ii) this time.

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-from_density.input`'.

Prepare the directort and start the calculation as before:

```
$ cd static
$ mkdir ck-4he1000-fs
$ cd ck-4he1000-fs
$ cp ../../code/4hedft/input_files/ck-4he1000-from_density.input .
$ cp ../../code/4hedft/input_files/ck-4he1000-from_density.sbatch .
$ ln -s ../../code/4hedft/4hedft 4hedft
$ ln -s ../pure-4he1000/den.out den.in
$ sbatch ck-4he1000-from_density.sbatch
```

3.3 A $^4\mathrm{He}_N$ droplet with a classical impurity under a constraint

To impose a given separation between the center of mass of the droplet and the impurity, e.g. to determine the energy dependence on the distance of 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 the input file.

Added parameters

lconstraint	enables the constraint-penalty energy term $\frac{1}{2}\lambda_C\left[z_{imp} - \langle z_{He}\rangle - z_{dist}\right]$ (true/false)
intens ($= \lambda_C$)	intensity of the constraint penalty term. Default value is set to $\lambda_C = 3000$
zdist	constrained distance of the impurity to the center of mass of the droplet in Å. Adjust `zimp` and `zdist` such that $z_{imp} - \langle z_{He}\rangle - z_{dist} = 0$ is satisfied. $\langle z_{He}\rangle$ Is obtained from the results (`.res`) file of a previous X−$^4\mathrm{He}_N$ calculation

An example input file is provided called `ck-4he1000-constrained.input` is provided at the usual location. It calculates the ground-state of a K−$^4\mathrm{He}_{1000}$ system where the potassium is displaced by 1 Å. Prepare the calculation directory and start the calculation as explained in the previous sections.

3.4 A $^4\mathrm{He}_N$ 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. The only three things that change with respect to the case described in section 3.2.1

$$\begin{aligned} \texttt{mode = 3} &\implies \texttt{mode = 2} \\ \texttt{limp = .false.} &\implies \texttt{limp = .true.} \\ \texttt{lexternal = .true.} &\implies \texttt{lexternal = false} \end{aligned}$$

and add the parameter

Added parameters

fileimpout	impurity density output filename

to save the impurity wave-function to disk. After this, proceed as before. An example input-file and launch script are provided named `qk-4he1000.input` and `qk-4he1000.sbatch`.

3.5 A ^{4}He$_N$ 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 wave-functions are relaxed in each others mean field. For sake of constancy 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 wave-function should be relaxed under a constraint
fileimpin	impurity density input filename
fileimpout	impurity density output filename

The corresponding input-file and launch script are called `qk-4he1000-constrained.input` and `qk-4he1000-constrained.sbatch`.

3.6 A pure ^{4}He$_N$ droplet hosting a vortex line

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

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

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

	Added parameters
lvortex	`.true./.false.` enable/disable the vortex imprinting
vortex_axis	choose the axis of which the vortex line will be parallel to: {`X,Y,Z`}
nv	number of vortex lines
omega	angular velocity of the vortex. Usually $\omega = 1$

and an extra input file `vortex.input`

	vortex.input parameters
{xv(n),xv(n),yv(n)}, {yv(n),zv(n),zv(n)}	the coordinates of the vortex core one set of coordinates for each n

Make sure that the name of the namelist corresponds to the axis of the vortex line, so:

`vortex_axis = 'Z'` $\Longleftrightarrow$ `&vortex_z`
`vortex_axis = 'Y'` $\Longleftrightarrow$ `&vortex_y`
`vortex_axis = 'X'` $\Longleftrightarrow$ `&vortex_x`

An set of example input files has been provide in `~/code/4hedft-vortex/input_files/`, named `pure-4he1000-vortex.input` and `vortex.input`. A launch script, `pure-4he1000-vortex.sbatch`, to start the calculation is also there. The example will calculate the ground state of a $^4\mathrm{He}_N$ droplet hosting one vortex line along the Z-axis, with the vortex core positioned at $(x, y) = (0, 0)$.

Prepare and start the calculation as explained in the previous sections.

4 Dynamic ^{4}HeDFT

From a pragmatic point of view, the dynamics is much simpler than the statics. It only needs an input density and some initial parameters. It doesn't care if the density is converged or not or if it is an eigen state of the Hamiltonian. It will simply take the input data and propagate it in time, even if the inputs are silly or make no physical sense. As long as the errors keep withing bounds, it keeps on going. It assumes the user knows what it's doing.

In the following two sections we will give to simple dynamical processes as examples. The simplest will be a head-on collision between a heliophilic atomic impurity (Xe) and a $^4\mathrm{He}_{1000}$ droplet. The second more involved example will demonstrate the dynamical process of photo-exciting a heliophobic atomic impurity (K in a $K{-}^4\mathrm{He}_{1000}$ system) from the K(4s) state to an excited X(4P) state.

4.1 Impurities in a spherically symmetric electronic state

We will now prepare and execute a simulation of a head-on collision between a xenon atom and a $^4\mathrm{He}_{1000}$ droplet at 200 m/s. For this calculation we need two things:

1. Time-Dependent ^{4}He DFT for classical atomic impurities in an isotropic electronic state (Program 3. in the list in Chapter 2), and
2. a converged density of a pure droplet so that our Xe has something to collide with.

Downloading and compiling the code is done exactly as before. For this it is assumed you are in your home-directory again. You can always return to your home directory by invoking `cd` and then press enter. Execute the following series of commands

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

After this you should have an executable called '`4hetddft-isotropic`'.

There is an example input file in `~/code/4hetddft-isotropic/input_files` called '`xe-he1000-head_on.input`' that we will use to start the calculation. Some of the fields that are in the input file are omitted here because they are already explained in the static chapter, while others are included again because they have changed meaning.

New (or changed) input parameters

mode	the different read modes to start or continue a calculation. 0 – start a dynamic calculation. You have to provide a file containing a density or wave function from a converged static calculation. 2 – continue a dynamic calculation. You have to provide a previously written file containing wave-function of a droplet and an impurity in an isotropic electronic state 3 – same as `mode = 2` but using a file containing a wave-function of a droplet and an impurity in an an-isotropic electronic state. In this case the electronic state of the impurity will be read, but no longer propagated in time. You can use this mode to study the dynamics of an impurity that underwent a transition from an an-isotropic electronic state. These states will be discussed in the next section.
selec	related to the `selec_gs` field in the statics. This field selects the He-X interaction potential. Can be looked up in `V_impur.f90`. E.g. for Xe this is `'Xe_He'`
r_cutoff	He-X interaction potential cutoff distance
umax	He-X interaction potential cutoff energy
v{x,y,z}imp	initial velocity of the impurity in ps/Å ($1\,\text{ps/Å} = 100\,\text{m/s}$).
deltatps	the time step in picoseconds. A typical value is 5×10^{-4}
filerimp	name of the output file containing the impurity location as a function of time. Default name is `rimp.out.#`, where the `#` is an ID to distinguish between different runs. But you are free to name it as you want.
filevimp	name of the output file containing the impurity velocity as a function of time. Default name is `vimp.out.#`
fileaimp	name of the output file containing the impurity accelleration as a function of time. Default name is `aimp.out.#`, where the `#` is an ID to distinguish between different runs. But you are free to name it as you want.
pcurr	this field is related to `pdenpar` discussed before but now it prints a complex wave function instead of a helium density. The filenames of these files are `density.#.dat`, where the `#` is an integer ID.
tzmean	sets the location of the mean of the absorbtion potential (see DFT Guide Section III.E. 'Absorbing potential at the box boundaries'). A good rule of thumb is to use `tzmean` = `zmax`-2 Å

To prepare our calculation directory and start the simulation execute the following commands

```
$ cd
$ mkdir dynamic
$ cd dynamic
$ mkdir xe-he1000-head_on
$ cd xe-he1000-head_on

$ cp ../../code/4hetddft-isotropic/input_files/xe-he1000-head_on.input .
$ cp ../../code/4hetddft-isotropic/input_files/xe-he1000-head_on.sbatch .
$ ln -s ../../code/4hetddft-isotropic/4hetddft-isotropic 4hetddft-isotropic
$ ln -s ../../static/pure-4he1000/den.out den.in
$ sbatch xe-he1000-head_on.sbatch
```

As usual, you can view the status of your job by invoking

```
$ squeue -u <cecam user ID>
```

Check that is has the status '`RUNNING`'. When this is the case, the program will generate the following output files.

	Output files
density.#.dat	file containing the full 3-dimensional wave-function of the helium droplet and also the position and velocity of the atomic impurity, if present (otherwise these are set to 0). The file has a header containing information on the number of iterations, the used time-step, the current evolution time, etc, during the time of writing. This information can be extracted when needed. The '`#`' in the filename is an integer ID that labels the files. The time difference between two files is always `deltatps`$\times$`pcurr`.
filerimp	output file containing the impurity position as a function of time. It is structured in 4 columns: time(ps), x(t), y(t), z(t). The coordinates are revered to the origin of the grid.
filevimp	output file containing the impurity velocity as a function of time. It is structured in 4 columns: time(ps), v_x(t), v_y(t), v_z(t). The coordinates are revered to the origin of the grid.
fileaimp	output file containing the impurity accelleration as a function of time. It is structured in 4 columns: time(ps), a_x(t), a_y(t), a_z(t). The coordinates are revered to the origin of the grid.
hedenini-{x,y,z}	1-dimensional cuts along the box's axis of the initial helium density.
*.err	file containing the run-time errors
*.res.#	file containing the results. `#` is an integer ID labeling the different runs of the program in case a calculation has bean interrupted, either intentional or unintentional.
timec.dat	file containing the time coordinates
uext.dat	Fourier transform of the helium-impurity interaction potential
uimp-{x,y,z}	helium-impurity interaction potential

How to monitor the calculation using the output files will be explained in Section 4.3.

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

We will now prepare and execute a simulation of a head-on collision between a xenon atom and a $^4\mathrm{He}_{1000}$ droplet at 200 m/s. For this calculation we need two things:

1. Time-Dependent ^{4}He DFT for classical atomic impurities in an isotropic electronic state (Program 3. in the list in Chapter 2), and
2. a converged density of a pure droplet so that our Xe has something to collide with.

Downloading and compiling the code is done exactly as before. For this it is assumed you are in your home-directory again. You can always return to your home directory by invoking `cd` and then press enter. Execute the following series of commands

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

After this you should have an executable called '`4hetddft-isotropic`'.

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

New (or changed) input parameters

mode	the different read modes to start or continue a calculation. 0 – start a dynamic calculation. You have to provide a file containing a density or wave function from a converged static calculation. 1 – continue a dynamic calculation from an isotropic electronic state. You have to provide a previously written file containing wave-function of a droplet and an impurity in an isotropic electronic state 2 – continue a dynamic calculation from an an-isotropic state. You have to provide a previously written file containing wave-function of a droplet and an impurity in an an-isotropic electronic state 3 – NO IDEA WHAT THIS IS SUPPOSED TO DO YET
Lstate	orbital angular momentum state, `'P'`, or `'D'`
Ldiag_jz	diagonalise J_z in its subspace to address its 2-fold degeneracy and to allow for the positive eigenvalues of J_z. Per default the negative ones are chosen.
Ljz	projection of the total angular momentum on the z-axis. Values can be $\pm 1/2$ and $\pm 3/2$. Make sure that if you choose the positive values that `Ldiag_jz=.true.`
instate	state label in the L–S coupled basis, 0 ($\Pi_{1/2}$), 1 ($\Pi_{3/2}$), 2 ($\Sigma_{1/2}$)
Als_P	value of the L-S coupling constant A_{LS}
selec_pi	as the other `selec`-fields before. Helium interaction potential with an impurity in an electronic Π-state
r_cutoff_pi	potential cut-off distance
umax_pi	potential cut-off energy for the Π-potential
selec_sigma	helium interaction potential with an impurity in an electronic Σ-state
r_cutoff_sigma	potential cut-off distance for the Σ-potential
umax_sigma	potential cut-off energy for the Σ-potential

Output files

lambda.out	time evolution of the impurity's electronic state. The file is structured in
projections.dat	projections

4.3 Monitoring a dynamic calculation

4.4 Continuing a previous dynamic calculation

	New input parameters
icurr	ID of the current input wave-function. If file containing the droplet's wave function (and λ-vector if applicable) that you are going to use to continue is called '`density.123.dat`', then `icurr=123`
iter0	current number of iterations. This number is included in the header of the wave-function file.
time0	current evolution time This number is also included in the wave-function file

5 (Visualising output data)

5.1 (1-dimensional data: gnuplot, python/matplotlib)

5.2 (2-dimensional data: python/matplotlib)

5.3 (3-dimensional data: ParaView)