An introduction to density functional theory for experimentalists

Tutorial 1.2

Hands-on session

Exercise 1

▶ Repeat the steps illustrated during Tutorial 1.1, in particular:

1 Login into your account, set the modules in the file `.bashrc`, and create your working directory

2 Download the QE software package, unzip, configure, and make the executable `pw.x`

3 Download the pseudopotential for silicon, `Si.pz-vbc.UPF`

4 Create the input files `silicon-1.in` and submission script `job-1.pbs`

5 Submit this job and check the output file `silicon-1.out`

For all these steps you can directly copy/paste the instructions from the PDF document of Tutorial 1.1 (or type everything if you are patient).

Exercise 2

We want to explore one important convergence parameter of DFT calculations, the planewave kinetic energy cutoff `ecutwfc`.
To this aim we create a new directory:

```
$ cd ~/scratch/summerschool ; mkdir tutorial-1.2 ; cd tutorial-1.2
```

and we copy over the important files generated in the previous exercise:

```
$ cp ../tutorial-1.1/pw.x ./

$ cp ../tutorial-1.1/Si.pz-vbc.UPF ./

$ cp ../tutorial-1.1/silicon-1.in ./silicon-3.in

$ cp ../tutorial-1.1/job-1.pbs ./job-3.pbs
```

Using `vi` we modify the input variable `ecutwf` to 5 Ry (note 1 Ry = 13.6058 eV). After this change, line #23 of `silicon-3.in` should read:

```
ecutwfc    = 5.0,
```

In order to be consistent with the new names of the input file we must also modify the job submission script `job-3.pbs` using `vi`, so as to have `silicon-3.in` and `silicon-3.out`.
Now we submit the job to the cluster as usual:

```
qsub job-3.pbs
```

When the job is finished we can analyze the output file `silicon-3.out`. This output file contains the most important information regarding your run. Throughout this school we will learn the meaning of the various sections of this file gradually. For now we concentrate only on a few simple aspects.

First of all we can check that we are using the local density approximation (LDA) to DFT. To see this, open the output file using `vi`, and search for the words `Exchange-correlation`. To activate the search function in `vi` we simply press `/` and enter the search word. You will find:

```
    Exchange-correlation      =  SLA  PZ   NOGX NOGC ( 1  1  0  0 0 0)
```

Here `SLA` stands for 'Slater exchange', `PZ` stands for Perdew-Zunger parametrization of the LDA, `NOGX` and `NOGC` say that density gradients are not taken into account (the meaning of this will become clear in Lecture 5.2). The numbers are internal codes of `pw.x`.

Now we search for the words `kinetic-energy cutoff`. This should be indentical to the value of `ecutwfc` set in the input file. This parameter is the kinetic energy cutoff of the planewaves basis set, and will be introduced formally in Lecture 2.2. This parameter sets the number of planewaves in which every Kohn-Sham wavefunction is expanded (ie the number of Fourier components of each wavefunction). The number of planewaves corresponding to the cutoff `ecutwfc` can be found by searching for `Kohn-Sham Wavefunctions`. You will see the following line:

```
      Kohn-Sham Wavefunctions         0.00 Mb     (      58,    4)
```

This means that we have 4 Kohn-Sham wavefunctions (corresponding to the 4 valence bands of silicon), and that each wavefunction is expressed as a linear combination of 58 planewaves.
Now we want to look at the most important quantity in the output file, the DFT total energy. Search for the marker `!` in the output file. You should find:

```
!    total energy              =     -15.60437814 Ry
```

This value should be taken with caution: it contains an offset which arises from the use of pseudopotentials (see Lecture 2.2), and it is not referred to vacuum, since there is no vacuum reference when we perform a calculation in a infinitely-extended crystal. This means that the absolute value of DFT total energies in extended solids is not meaningful; what is meaningful is the **total energy difference** between two configurations.

Finally we look at the timing: search for the word '`PWSCF        :`' (mind the colon and the blanks). You should find:

```
     PWSCF        :      0.08s CPU          0.20s WALL
```

The number on the left is the CPU-time, that is the execution time as measured on each individual CPU. The number on the right is the 'wall-clock' time, and indicates the actual time elapsed from the beginning to the end of the run.

Now we want to study how the total energy, the number of planewaves, and the timing vary as a function of the planewaves cutoff `ecutwfc`.
▶ Repeat the above steps by setting `ecutwfc` to 5, 10, 15, 20, 25, 30, 35, 40 in the input file. It is convenient to generate separate input/output files and then search for the energy, the number of planewaves, and the CPU time in each output file. You can collect the results for example by creating a text file using `vi exercise2.txt` and entering your results one by one. You should be able to construct a file looking like this [please note that your numbers will not be identical since this was executed on a different cluster]:

```
$ more excercise2.txt
```

```
# ecutwfc (Ry)   planewaves   energy (Ry)    time (s)
    5               58        -15.60437814    0.07
   10              153        -15.77558550    0.10
   15              274        -15.83422234    0.10
   20              416        -15.84721988    0.16
   25              580        -15.85087570    0.18
   30              763        -15.85182962    0.19
   35              959        -15.85235153    0.23
   40             1185        -15.85268618    0.27
```

Note: If you do not want to change each input file manually you can use the following loop to generate the files:

```
$ cat > loop.tcsh << EOF
foreach ECUTWFC ( 5 10 15 20 25 30 35 40 )
  sed "s/5\\.0/\${ECUTWFC}/g" silicon-3.in > silicon-\${ECUTWFC}.in
end
EOF
$ tcsh loop.tcsh
```

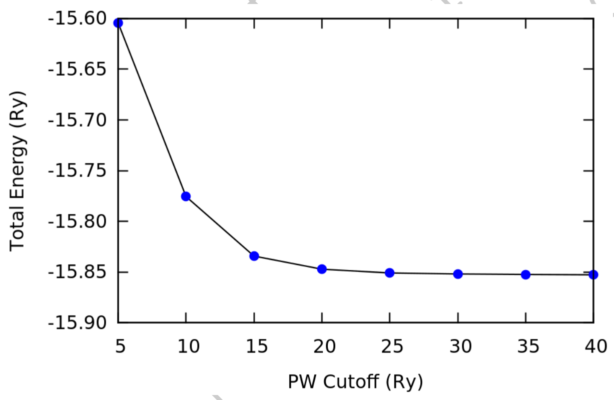
Furthermore you can use the command `grep` in order to extract the information that you are looking for automatically. For example:

```
$ grep "\!" silicon-5.out
```

```
!    total energy              =     -15.60437814 Ry
```

At this point we can analyze our results. For this you can either use `gnuplot` directly on the cluster, or you can transfer the file `exercise2.txt` using the command `scp` or the program `filezilla`, and then plot the data using your favourite software (eg `Origin` or `Excel`).

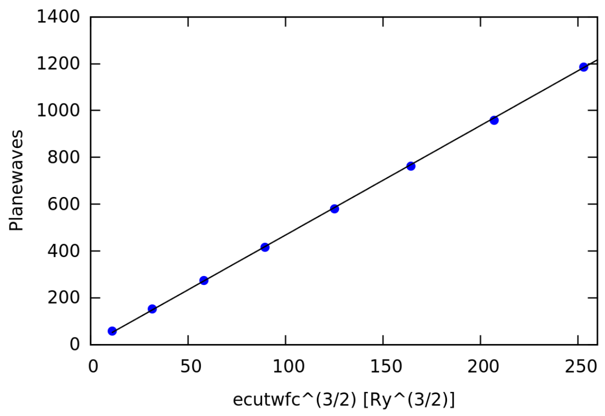
For the total energy you should find something like this:



Here we see that, using a cutoff of 25 Ry, we obtain a total energy which is only 12 meV/atom higher than our best-converged value at 40 Ry. In principle we should test even higher values of `ecutwfc` (the correct result is obtained in the limit of this cutoff reaching infinity), but in practice 25 Ry is good enough for this tutorial. Most quantities that can be computed using DFT depend critically on this cutoff, therefore it is **very important** to alway perform this test when running DFT calculations.

If the planewaves cutoff is so important, why can we not use a very large value to be on the safe side? The answer is that the higher the cutoff, the more time-consuming the calculation. You can test this directly by plotting the CPU time vs. the cutoff using the values inside the file `exercise2.txt`.
In this example the runtime is below 1 sec, therefore the choice of the cutoff is not important in practice. However, in most DFT calculations a careful choice of cutoff can save us weeks of computer time.

The longer times required for higher cutoffs relate to the fact that we are performing linear algebra operations using larger vectors to describe the wavefunctions.
▶ Verify that the number of planewaves increases with the cutoff.
▶ Verify that a plot of the number of planewaves vs. `ecutwfc`$^{3/2}$ yields a straight line:



▶ Can you explain the origin of this relation between the cutoff and the number of planewaves?
Note: in order to answer this question you will need to go through Lecture 2.2 first.

Exercise 3

We now want to explore one other convergence parameter of DFT calculations for crystals, the Brillouin zone sampling K_POINTS (to be described in Lecture 2.2).

In the input file `silicon-3.in` we had requested a uniform sampling of Bloch wavevectors **k** by setting `4 4 4 1 1 1`. This means that we want to slice the Brillouin zone in a $4 \times 4 \times 4$ grid, and we shift this grid by half a grid spacing in each direction (`1 1 1`). This shift is used because it usually provides a better sampling. So now we expect the code to work with exactly $4 \times 4 \times 4 = 64$ **k**-vectors.

▶ Now search for '`number of k points`' in the output file `silicon-4.in`. You should find:

```
     number of k points=     10
```

Therefore the code is using only 10 **k**-points instead of the expected 64 points. The reason for this difference is that many points in our grid are equivalent by symmetry. The code recognizes these symmetries and only performs explicit calculations for the inequivalent points.

▶ Determine how the total energy of silicon varies with the number of **k**-points, using the same procedure as in Exercise 2. Consider the following situations for the input parameters `K_POINTS`: `1 1 1 0 0 0` / `2 2 2 0 0 0` / `4 4 4 0 0 0` / `8 8 8 0 0 0` / `16 16 16 0 0 0`. For these calculations you can use our 'converged' cutoff `ecutwfc = 25.0` Ry.
Note In this case you will need to set the execution flag `-npool` inside the submission script to 1, for example: `-n 8 pw.x -npool 1` instead of the original `-n 4 pw.x -npool 4`.

▶ Repeat the last operation, this time using nonzero shifts, eg `1 1 1 1 1 1` / `2 2 2 1 1 1` / `4 4 4 1 1 1` and so on.
You should be able to construct two files similar to the following ones:

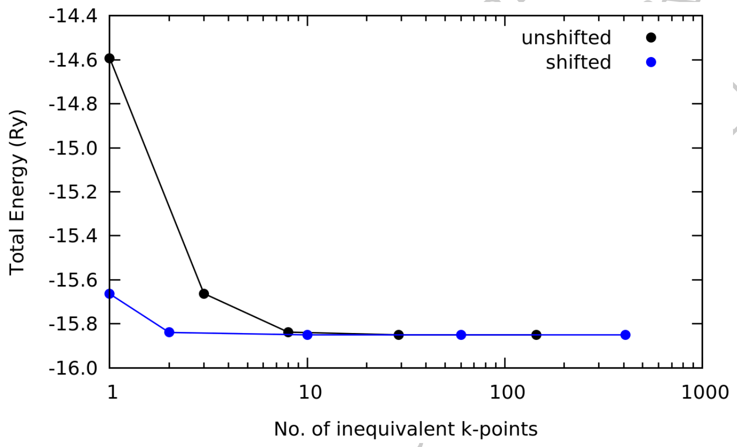
```
$ more excercise3a.txt

#  grid      shift     energy (Ry)  inequiv. k-points   time (s)
  1  1  1    0 0 0   -14.59239650     1                  0.20
  2  2  2    0 0 0   -15.66198445     3                  0.25
  4  4  4    0 0 0   -15.83707680     8                  0.21
  8  8  8    0 0 0   -15.85079066    29                  0.56
 16 16 16    0 0 0   -15.85108168   145                  2.41
```

```
$ more excercise3b.txt

#  grid      shift     energy (Ry)  inequiv. k-points   time (s)
  1  1  1    1 1 1   -15.66368666     1                  0.10
  2  2  2    1 1 1   -15.83894845     2                  0.10
  4  4  4    1 1 1   -15.85087570    10                  0.22
  8  8  8    1 1 1   -15.85108292    60                  1.04
 16 16 16    1 1 1   -15.85108131   408                  6.88
```

▶ Plot the total energy as a function of the number of inequivalent **k**-points in each calculation, both for the case of the unshifted grid (0 0 0) and the shifted grid (1 1 1). You should obtain something similar to the following (note the logarithmic scale in the horizontal axis):



Here we can see that by using the 4 4 4 1 1 1 grid we obtain a total energy which is already very good, <2 meV/atom away from our best value at 8 8 8 1 1 1. We also see that the shifted grid converges faster than the unshifted grid.

▶ Plot the CPU time vs. the number of inequivalent **k**-points and verify that the time scales approximately linearly with the number of such points.

Exercise 4

In this exercise we want to explore the **scaling** of DFT calculations as a function of system size.

The input file `silicon-1.in` has been modified to generate 5 new input files which you can download and unpack as follows:

```
$ wget http://giustino.materials.ox.ac.uk/tutorial_1.2_exercise_4.tgz
$ tar xfz tutorial_1.2_exercise_4.tgz ; ls tutorial_1.2_exercise_4

silicon-4.1.in  silicon-4.2.in  silicon-4.3.in  silicon-4.4.in  silicon-4.5.in
```

These files correspond to supercells of silicon, containing one primitive unit cell (`silicon-4.1.in`), a $2{\times}2{\times}2$ supercell (`silicon-4.2.in`), and so on, up to $5{\times}5{\times}5$ primitive unit cells (`silicon-4.5.in`). By looking inside these input files you can check that we have a number of Si atoms ranging from 2 to 250.

▶ Now run `pw.x` using these five different input files, and extract the CPU time in each case. The procedure for running jobs is the same as in the previous exercises.
Your data should look similar to the following (note that the timing of each job will depend on the cluster, but the relative times are meaningful)

```
# atoms   cputime (s)
     2      0.09
    16      0.44
    54      4.53
   128     34.82
   250    218.78
```

▶ Plot the CPU time as a function of the number of atoms. Can you identify a simple law relating these two quatities?

Generally speaking DFT calculations scale with the cube of the number of atoms, $T_{\rm CPU} = {\rm const} \cdot N^3$ (N = number of atoms). This can be verified directly by plotting the above data using the cube of the first column: this plot should give an approximately straight line.
The take-home message here is that if we double the size of our system, then our DFT calculation will require approximately 8 times longer to complete (eg 1 week$\rightarrow$ 8 weeks).