Lab 2 Handout: Quantum Espresso DFT Code

1 The First-Principles Code: Quantum Espresso

We will be using the PWscf code included in the Quantum-Espresso package as our first-principles code. Quantum-Espresso is a full *ab initio* package implementing electronic structure and energy calculations, linear response methods (to calculate phonon dispersion curves, dielectric constants, and Born effective charges) and third-order anharmonic perturbation theory. Inside this package, PWscf is the code we will use to perform total energy calculations. PWscf uses both norm-conserving pseudopotentials (PP) and ultrasoft pseudopotentials (US-PP), within density functional theory (DFT). In addition to basic self-consistent plane-wave calculations (PWscf module) Quantum-Espresso has capabilities for ab-initio molecular dynamics, vibrational and optical spectroscopy, dielectric and magnetic response, and tools for ionic and electronic transport. Quantum-Espresso is free under the conditions of the GNU GPL. Further information (including online manual) can be found at the Quantum-Espresso website (`http://www.quantum-espresso.org/`. There are many other first-principles codes that one can use, for a full up-to-date listing see

`https://en.wikipedia.org/wiki/List_of_quantum_chemistry_and_solid-state_physics_software`

The rest of this handout will show how to get energies and forces using the PWscf code in Quantum-Espresso. Some helpful unit conversions are the following:

1 bohr = 1 a.u. (atomic unit) = 0.529177249 Å (Angstrom)
1 Ry (Rydberg) = 13.6056981 eV
1 eV = $1.60217733 \times 10^{-19}$ J

2 How to run PWscf

2.1 Environment setup

For the computations related to Lab 2 you will be using the same VirtualBox VM environment. Please refer to the document on the class website for the initial setup of the VM, in case you need to repeat the first steps. The Quantum Espresso code is located in `/home/bond/Software/qe-6.0`, and the compiled binary executable is already linked to `/home/bond/bin/pw.x`. The code includes an extensive set of examples demonstrating its different capabilities in `/home/bond/Software/qe-6.0/Examples/PW`.

- First we will need to setup the environment for this lab.

  Edit your `/.profile` file to specify a new environment variable that tells the scripts the location of pseudopotentials for the DFT calculations:

  ```
  export ESPRESSO_PSEUDO=$HOME'/Software/qe-6.0/pseudo'
  ```

  and check that you have the variable pointing to the Quantum Espresso PWscf binary:

  ```
  export PWSCF_COMMAND=$HOME'/bin/pw.x'
  ```

  For these changes to take effect, you will need to close and restart the terminal shell windows. Alternatively, you can run the following command to update your environment: `$ source ~/.profile`

- Each atom in a DFT calculation is typically modeled with a pseudopotential, that smoothly approximates the chemically irrelevant core electrons without affecting the valence electrons. The pseudopotential library for Quantum Espresso is here:

  `http://www.quantum-espresso.org/pseudopotentials/`

  We will need an LDA pseudopotential for Germanium for this lab, so let's download it

  ```
  $ cd /home/bond/Software/qe-6.0/pseudo
  ```

  ```
  $ wget http://www.quantum-espresso.org/wp-content/uploads/upf_files/Ge.pz-bhs.UPF
  ```

- Now download the workflow automation tools and Python plugins for Quantum Espresso that you will be adapting to solve exercises in this lab.

  ```
  bond@vmint ~ $ cd ~/Software/labutil/
  ```

  ```
  bond@vmint ~/Software/labutil $ git pull
  ```

  If you changed the plugins and samples in this directory while working on the previous assignment, the git command will notice this and will prompt you to 'merge' your changes. Do not do this. Instead, backup any of your changes by renaming this folder to something else, for instance:

  ```
  bond@vmint ~/Software/labutil $ cd ..
  ```

  ```
  bond@vmint ~/Software $ mv labutil/ my_labutil
  ```

  Then repeat the initial procedure steps to set up a fresh repository (see previous lab):

  ```
  bond@vmint ~/Software $ git clone https://github.com/bkoz37/labutil.git
  ```

  If, however, git does not detect any changes when you do `git pull`, it will simply download and update the files.

  Copy the sample files to your working directory.

  ```
  bond@vmint ~/Software/labutil/lab2_samples $ cp * ~/WORK/Lab2/
  ```

- You are now ready to run DFT calculations.

2.2 Running a PWscf example

Once you are in your working directory for this lab, take a look at the sample file `pwscf.in`. It contains the input information needed to do run to compute the energy of a germanium(Ge) unit cell using PWscf. If you edit the file with your favourite editor (vi, emacs, nano etc). It will look like this

```
&control
      calculation = 'scf'
      tstress = .true.
      tprnfor = .true.
      pseudo_dir = '/home/bond/Software/qe-6.0/pseudo'
      outdir='/home/bond/WORK/Lab2/Problem1/test'
   /
   &system
      ibrav = 0
      nat = 2
      ntyp = 1
      ecutwfc = 30
   /
   &electrons
      diagonalization = 'david'
      mixing_beta = 0.5
      conv_thr = 1e-07
   /
   K_POINTS automatic
      4 4 4  0 0 0
   ATOMIC_SPECIES
     Ge 72.61 Ge.pz-bhs.UPF
   CELL_PARAMETERS {angstrom}
     0.0 2.5 2.5
     2.5 0.0 2.5
     2.5 2.5 0.0
   ATOMIC_POSITIONS {angstrom}
     Ge 0.00000 0.00000 0.00000
     Ge 1.25000 1.25000 1.25000
```

The most relevant lines for the calculations related to this lab are the following:

- The line

  ```
  calculation = 'scf'
  ```

  indicates the task to be performed. That this is a 'self-consistent field' calculation (scf) to find the energy of the system using the iterative Kohn-Sham procedure. You

can change this line to do non-self-consistent calculations, if for example we wanted to compute the band structure of a material. In that case you would specify `calculation = 'bands'` to run the `pw.x` in the same folder as the results of the scf run. It will perform a non-self-consistent calculation to get the eigenvalues at specified k-points (see below) based on the charge density that is already present in the folder.

- The line

  ```
  tprnfor = .true.
  ```

  indicates that forces will be printed out. A similar line appears for the stress.

- The line

  ```
  pseudo_dir = '/home/bond/Software/qe-6.0/pseudo'
  ```

  indicates the location of the pseudopotential file that describes the behaviour of inner electrons for each Ge atom . The code will look for the pseudopotential file in that directory.

- The line

  ```
  outdir_dir = '/home/bond/WORK/Lab2/Problem1/test'
  ```

  indicates the directory where temporary files (wavefunction, charge density) will be written. We will be running each calculation and writing the output file to the same folder, for convenience.

- The lines

  ```
  ibrav = 0
  nat = 2
  ntyp = 1
  ```

  indicate that our unit cell will be completely specified by its cell vectors using the `CELL_PARAMETERS` block, as described below. We also tell the code that we have two atoms ($nat = 2$) of the same ($ntyp = 1$) species in it. These parameters will be automatically filled in by our Python plugin, based on the information it finds in the structure object.

- The line

  ```
  ecutwfc = 30.0
  ```

  indicates that a plane-wave kinetic energy cutoff of 30 Ry will be used. Plane-waves with higher energies will not be included into the basis to describe our wavefunctions. This number will need to be changed until you are sure that the calculations are well converged with respect to the size of the basis (see next section).

- `Ge 72.61 Ge.pz-bhs.UPF` indicates the atom label (Ge), atomic mass unit number for Ge and the name (choice) of pseudopotential to be used.
- The lines

```
ATOMIC_POSITIONS {angstrom}
Ge 0.00000 0.00000 0.00000
Ge 1.25000 1.25000 1.25000
```

  indicate where the two atoms in the unit cell are located.

- The lines

```
K_POINTS automatic
4 4 4 0 0 0
```

  indicate that the Morkhost-Pack grid of k-points to be used contains $4 \times 4 \times 4$ points, and it is not displaced from the origin. This number will need to be changed until you are sure that the calculations are well converged with respect to the size of the k-points grid (see next section).

- The full listing and description of all input file keywords is located here:

  `http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PW.html`

We are ready now to run our first PWscf calculation. Just type

```
bond@vmint ~/WORK/Lab2 $ pw.x < pwscf.in > pwscf.out
```

After a few seconds if everything goes well you should have in your directory a file called `pwscf.out`. All relevant information is there. To look at the total energy of the system type in command line

```
grep total pwscf.out
```

At the end you will find lines containing something like

```
!    total energy              =   -15.99179135 ryd
      total   stress  (a.u.)                          (kbar)     P=  -20.27
```

so here you can see the total value of the energy for the cell (your number might be slightly different) and the total stress. Note that the final occurrence of total energy will have an exclamation point by it. This something that you can use to hunt for it manually, for example by typing

```
grep '^!' *.out
```

which will search all files terminated by .out in the current directory, looking for lines that start with an exclamation point.

At the end of the file, it is written how long your program took to run. For example:

```
PWSCF        :     0.34s CPU          0.36s WALL
```

Your numbers may be different from these.

There will also be lines which say:

```
number of k points=    8
cart. coord. in units 2pi/alat
k(1) = (  0.0000000  0.0000000  0.0000000),  wk = 0.0312500
k(2) = ( -0.2500000  0.2500000 -0.2500000),  wk = 0.2500000
k(3) = (  0.5000000 -0.5000000  0.5000000),  wk = 0.1250000
k(4) = (  0.0000000  0.5000000  0.0000000),  wk = 0.1875000
k(5) = (  0.7500000 -0.2500000  0.7500000),  wk = 0.7500000
k(6) = (  0.5000000  0.0000000  0.5000000),  wk = 0.3750000
k(7) = (  0.0000000 -1.0000000  0.0000000),  wk = 0.0937500
k(8) = ( -0.5000000 -1.0000000  0.0000000),  wk = 0.1875000
```

This records the number of unique k-points that were calculated. Some points from original $4 \times 4 \times 4 = 64$ mesh are symmetry related(identical) and have the same energy so the code doesn't compute them more than once. They are then weighted differently. The weights add up to 2, an idiosyncrasy of this program. Your numbers will be different if you use a different k-point mesh.

2.3 Using functional workflows to run PWscf multiple times

In order to check the numerical convergence of your calculations with respect to energy cutoff and number of k-points, you will have to run PWscf multiple times with different values for those parameters. Instead of creating many different input files by hand and using them one by one to run PWscf, we will do this more efficiently by using automated Python plugins that will create input files for you, run the binary code, and parse out the necessary results from the output file. This way all your work is contained within Python functions connected together into a reusable workflow. An example that you can start from and modify for this lab is now in your folder `~/WORK/Lab2/`, where it is safe to be modified. (Note: do not modify files in your `labutil` folder, it is only used to download lab samples.)

By modifying the Python workflow, you can loop over plane-wave cutoffs, k-point grids, and lattice constants. You can create your own naming scheme for labeling directories containing each calculation. Use the `runpath` variable in the script and set it dynamically as you like.

3 Numerical convergence issues

3.1 Plane waves

Remember that we are dealing with infinite systems using periodic boundary conditions. This means that we can use the Bloch theorem to help us solve the Schrdinger equation. The Bloch theorem states that

$$\psi_{n\mathbf{k}} = e^{i\mathbf{kr}} u_{n\mathbf{k}}(\mathbf{r}),$$

with

$$u_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} c_{\mathbf{G}} e^{i\mathbf{G}\mathbf{r}}.$$

In these equations, $\psi_{n\mathbf{k}}(\mathbf{r})$ is the wavefunction, $u_{n\mathbf{k}}(\mathbf{r})$ is a function that is periodic in the same way as the lattice, the sum goes over all (at least in principle) reciprocal lattice vectors $\mathbf{G}$, and the $c_{\mathbf{G}}$ are coefficients in the basis expansion. In this case, our basis functions (i.e., what we expand in) are plane waves. They are called plane waves because surfaces of constant phase are parallel planes perpendicular to the direction of propagation. Remember that limiting the plane-wave expansion to the infinite, but numerable and discrete set of G vectors that are integer multiples of the three primitive lattice vectors, we are selecting plane waves that have a periodicity compatible with the periodic boundary conditions of our direct lattice.

In actual calculations, we have to limit the plane wave expansion at some point (i.e., stop taking more values of $\mathbf{G}$). This is called the planewave cutoff. Cutoffs are always given in energy units (such as Rydberg or eV) corresponding to the kinetic energy of the highest $\mathbf{G}$. Note: The units of reciprocal lattice are the inverse of the direct lattice, or 1/length. However, we can convert 1/length to energy units (remember that $\lambda\nu = c$ and that $E = h\nu$, where λ is a wavelength, ν is a frequency, and E is an energy).

Problem 1 is designed to test cutoff convergence issues. You can always take a higher cutoff than you need, but the calculations will take longer.

3.2 K-points

Because of the Bloch theorem, we need to solve a Schrödinger-like Kohn-Sham equation (i.e. iterate selfconsistently the diagonalization of a $M \times M$ matrix, where M is the number of basis functions) everywhere in the first Brillouin zone (if you dont know what a Brillouin zone is, its time to go over the concepts of direct and reciprocal lattice). In practice, we do this for a finite number of $\mathbf{k}$ values (e.g., the coarse grained k-point "Monkhorst-Pack" mesh), and get a value for E at each $\mathbf{k}$. To obtain a value for E, the energy of the crystal, we need to integrate over the first Brillouin zone, where the bands are occupied (and divide by the volume). Thus, summing over a finite number of $\mathbf{k}$ points is an approximation to performing an integral. You will need to make sure you have enough $\mathbf{k}$-points to have a converged value for the energy.

3.3 Summary about numerical convergence

For all first-principles calculations, you must pay attention to two numerical convergence issues. The first is the energy cutoff, which is the cutoff for the wavefunction expansion. The second is number of k-points, which measures how well your discrete grid has approximated the continuous integral.

4 Hints for Solving the Problems

In order to solve the problems in Problem Set 2, you will have to run several PWscf calculations. You can organize your runs any way you like. One way is to make directories for each calculation that you do, and name your folders accordingly. Or you can modify the plugins to do it differently. Good organization may save you headache in the long run (but is totally up to you).

4.1 Problems 1 and 2

In problem 1 we check the convergence of the total energy with respect to the plane-wave cutoff. All the variables except the plane-wave cutoff should keep the same value in all the runs. Check the total energies to decide when the calculations are converged.

Problem 2 will test the convergence of the energy with increasing the density of the k-point grid. We will be testing k-point grid convergence and cutoff convergence separately. So, set your cutoff to something lower, such as 30 Ry (or some other cutoff that you have used already in Problem 1). There are some cross effects in testing cutoff and k-point separately, however we assume these are small. You may also want to test denser grids. You will also want to record the number of unique k-points (that is, unique by symmetry). This is near the beginning of the output file, and looks something like this:

```
number of k points=    8
```

Your calculation will scale roughly as the number of unique k-points. You can verify this with the timing information.

4.2 Problems 3 and 4

In problems 1 and 2, the forces on Ge are zero in the x, y, and z directions. This is because of symmetry, which cancels out forces. In problems 3 and 4, we will create forces by displacing a Ge atom +0.05 (fractional coordinates) in the z direction. To do this, remember how to move at atom using ASE (see IPython notebook). The forces will appear in the output file after the total energies. It will look something like this:

```
 Forces acting on atoms (Ry/au):

     atom    1 type  1   force =     0.00000000    0.00000000    0.11794007
     atom    2 type  1   force =     0.00000000    0.00000000   -0.11794007

     Total force =     0.235880     Total SCF correction =     0.000005
```

The numerical value for your forces may be slightly different from the above example. Forces are given in units of Ry/bohr, but the Python plugin converts them to eV/atom. Record this number, and retest the convergence issues with respect to cutoffs and k-points.

5 FAQ

- **I get a message: Note: The following floating-point exceptions are signalling: IEEE DENORMAL**

  This just a low-level warning from the GNU compiler not being happy with a floating point numeric convention in the PWscf code. Ignore it.

- **How precisely do I need to get the lattice parameter?**

  Lattice parameters are typically listed to within 0.01 Å. There are applications when higher precision is required; this is *not* one of them.

- **The energies when you move an atom (the force calculations) are higher than when you don't?**

  This is correct. Remember equilibrium has the lowest energy. Equilibrium for this structure has Ge atoms at (0, 0, 0) and at (0.25, 0.25, 0.25). As a side note the forces will give you an idea of how far you are from equilibrium (they tell you which direction the atoms "want" to move). The stresses tell you which direction the cell parameters "want" to change to reach equilibrium.

- **My energy versus lattice constant plot is jagged.**

  There are a number of solutions to this; the easiest is to raise the energy cutoff.

- **The weights of the k-points add up to 2, not 1.**

  Yes. This is a feature of the code. Dont worry about it.

- **How is "convergence of energy" defined?**

  You say that your energy is converged to X Rydbergs when $E_{\text{true}} - E_n = X$ (E_n is the current energy). How do you know E_{true}? In practice you might take your energy at the highest cutoff (or k-grid, or whatever) that you calculated — if that seems converged, you might call that E_{true}. The most important thing is that you do *not* define convergence as $E_{n+1} - E_n$, where n is a step in energy cutoff (or k-grid, or whatever).

  You do need to be careful though. It is possible to get false or accidental convergence as well. That is, your energy at a $2 \times 2 \times 2$ k-grid may be the same as the energy at a $8 \times 8 \times 8$ k-grid, but the energy at a $4 \times 4 \times 4$ might be very different from both of these. In this case, you aren't really converged at a $2 \times 2 \times 2$ k-grid.

- **I don't understand convergence of energy and forces. It seems that, as a percentage of the absolute value, energies converge much faster.**

  Sometimes you are interested in an absolute value, rather than a percentage value. For instance, let's say you can measure the length to within 1 mm. If you measure the length of an ant, in terms of percentage error, you may be off by 50% or more. If you measure the length of an elephant, in terms of absolute error, you may be off by 0.01%. But usually you don't care as long as you are to within 1 mm. Errors on forces are the

same. Don't worry about the percentage errors so much. You could always arbitrarily decrease the percentage errors on the forces by taking a bigger displacement. From experience, we know that a good error on energy differences is around 5 meV/atom. From experience, we also know that a good error on forces is 10 meV/Å. These are just values that we know, because we have done many first-principles calculations in the past. This is what you should look for.

- **Does PWscf use LDA or GGA? DFT or Hartree Fock?**

  PWscf uses DFT. It has both LDA and GGA implemented, but in this lab we only use LDA.

- **My energies are really different from Lab 1. What is the "correct" scale I should be looking for?**

  Remember that absolute energies do not usually mean very much. We are mostly interested in energy differences. Also, the reference energies are different. In lab 1, the reference energy (when atoms are infinitely far apart) is 0. In lab 2, this is not the case. The absolute energies can shift a lot, depending on which reference energies you take.

- **Why do I take k-point grids with the same number of points per direction? Can I take other k-point grids?**

  For this material, we take the same number of k-points per direction because the three lattice directions are equivalent. This is not always true. The "best" k-point meshes are those that sample k-space evenly in all directions. Thus, for a tetragonal cell (with $a = b$, $c = 2a$, and all angles 90 degrees) we might take a $8 \times 8 \times 4$ mesh. Think about why this is so. (Note that if a lattice vector is longer in real space, it is shorter in k-space).