First-principles Lab Instructions

We will be using 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. Quantym-Espresso also contains two molecular-dynamics codes, CPMD (Car-Parrinello Molecular Dynamics) and FPMD (First-Principles Molecular Dynamics). 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).

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 or http://www.pwscf.org/. Quantum-Espresso is currently at version 2.1.2. and is under very active development.
There are many other first-principles codes that one can use. Here is a non-comprehensive list:

ABINIT
http://www.abinit.org
DFT Plane wave pseudo-potential (PP) code. ABINIT is also distributed under the GPL license.

CPMD
http://www.cpmd.org
DFT Plane wave pseudopotential code. Car-Parrinello molecular dynamics. GPL.

SIEST A
http://www.uam.es/departamentos/ciencias/fismateriac/siesta/
DFT in a localized atomic base.

VASP
http://cms.mpi.univie.ac.at/vasp/
DFT Ultrasoft pseudo-potential (US-PP) and P A W code. US PP' s are faster and more complex than corresponding PP codes, with similar accuracy. Moderate cost for academics (~$2000)

WIEN2k
http://www.wien2k.at/
DFT Full-Potential Linearized Augmented Plane-Wave (FLAPW). FLAPW is the most accurate implementation of DFT, but the slowest. Small cost for academics (~$500)

Gaussian
http://www.gaussian.com

Quantum Chemistry Code - includes Hartree-Fock and higher-order correlated electrons approaches. Moderate cost for academics (~$3000). Has recently been extended to solids.

Crystal
http://www.cse.clrc.ac.uk/cmg/CRYSTAL
HF and DFT code. Small cost for academics (~$500).

This is a tutorial on how to get **energies** using the PWSCF code in Quantum-Espresso. Some helpful conversions:
1 **bohr** = 1 **a.u.** (atomic unit) = 0.529177249 **angstroms.**
1 **Rydberg** = 13.6056981 **eV**
1 **eV =**1.60217733 x 10-19 **Joules**

Problem 1 concerns convergence issues in first-principles calculations. Below is some background for problem 1A and 1B; if you do not think you need it, skip to the "summary of background" section.

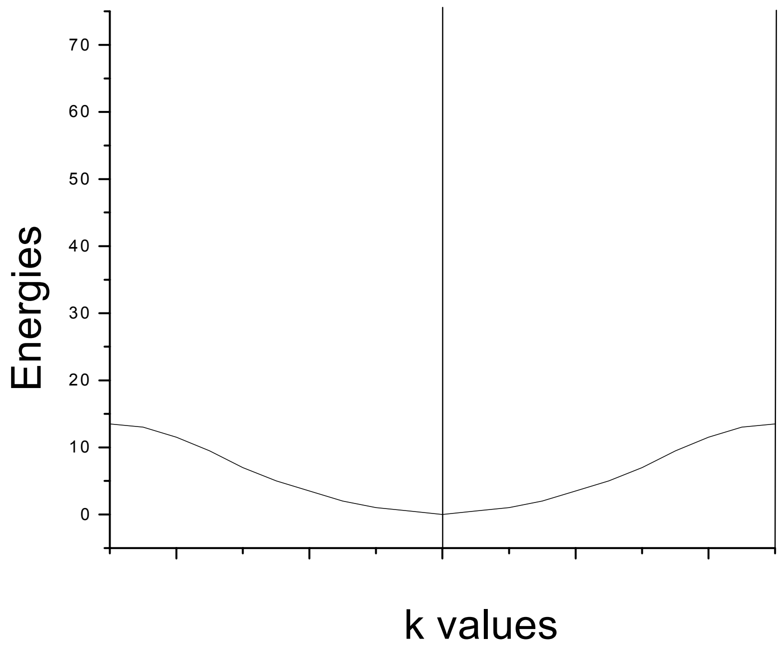
Background for problem 1A
Remember that we are dealing with infinite systems using periodic boundary conditions. This means that we can use the Bloch theorem to help us solve the Schrödinger equation. The Bloch theorem says:
$\psi_{n\vec{k}}(\vec{r}) = \exp\left(i\vec{k}\cdot\vec{r}\right) u_{n\vec{k}}(\vec{r})$
with
$u_{n\vec{k}}(\vec{r}) = \sum_G c_G \exp(i\vec{G}\cdot\vec{r})$
In these equations, $\psi_{n\vec{k}}(\vec{r})$ is the wavefunction, $u_{n\vec{k}}(\vec{r})$ is a function that is periodic in the same way as the lattice, G sums over all (at least in principle) reciprocal lattice vectors, and *c_G's* are coefficients in the expansion. In this case, our *basis functions* (ie 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 G 's). 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 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 $\lambda\nu = c$, and $h\nu = E$. λ is wavelength, and is in units of 1/length)

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

Background for Problem 1B.
Because of the Bloch theorem, we need to solve a Schrödinger-like Kohn-Sham equation (i.e. iterate selfconsistently the diagonalization of a MxM matrix, where M is the number of basis functions) everywhere in the Brillouin Zone. In practice, we do it for a finite number of *k* values (e.g. the coarse grained Monkhorst-Pack mesh), and get a value for E at each *k* . This is seen in the schematic below.



The picture goes over the first Brillouin zone. (If you don't know what a Brillouin zone is, it's time to go over the concepts of direct and reciprocal lattice). 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 *k* points is an approximation to performing an integral. You will need to make sure you have enough *k* -points to have a converged value for the energy.

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

Notes on the input files: (more info in espresso-5.0/PW/Doc/INPUT_PW.html)
&control declares the control block.
calculation = 'scf' tells PWSCF that this will be a self-consistent field calculation.
restart_mode = 'from_scratch', declares that we will be generating a new structure.

prefix='diamond', declares the filename prefix to be used for temporary files.
tstress = .true. is a flag to calculate the stresses.
tprnfor = .true. is a flag to calculate the forces.
pseudo_dir = ‘path’ defines the location of the directory where you store the pseudo-potentials.
outdir='path' defines the location of the temporary files. This should always be a local scratch disk so that large I/O operations do not occur across the network.
/ denotes the end of a block.

The system block ibrav – gives the crystal system. ibrav=2 is a face-centered cubic structure. This is used because the symmetry of the structure can reduce the number of calculations you need to do. If you need other crystal systems, consult the documentation celldm – defines the dimensions of the cell. **You will be changing this parameter.** celldm is in atomic units, or bohrs. Remember that **1 bohr** = 0.529177249 **angstroms.** The value will depend on the Bravais lattice of the structure (see below for a key). For FCC, celldm(1) = *a0* . In cubic systems, *a=b=c= a0*.
nat – number of atoms (each individual unique atom). Note that diamond has two unique atoms in the smallest assymetric unit.
ntyp – number of types of atoms ecutwfc – Energy cutoff for pseudo-potentials. **This one is important; you will be changing this parameter.**

diagonalization – diagonalization method. Use the default for now.
mixing_beta - Mixing factor. Don’t worry about this for now.
mixing_mode – Mixing method. Don't worry about this for now.
conv_thr – Convergence threshold. Use the default for now.

Atomic species declaration After the keyword ATOMIC_SPECIES, for each ntyp enter atomic symbol atomic weight pseudo-potential
Atomic positions After the keyword ATOMIC_POSITIONS, for each nat enter atomic symbol x y z where x,y,z are given as fractional coordinates of the conventional cell.

k-point selection after the keywork K_POINTS, “automatic” tells PWSCF to automatically generate a k-point grid. The format of the next line is nkx nky nkz offx offy offz where nk* is the number of intervals in a direction and off* is the offset of the origin of the grid.

Using the Scripts

The provided scripts allow you to submit your jobs with “sbatch run_script.sh” Quite often the scripts have for loops or lists in them to help you run many simulations at once. You may have to define what those lists will do or fill in variables as part of the assignment.

These are where you will change the range of the three main parameters you want to test (lattice parameter *a,* energy cutoff *ecut* and k-point grid *k,* in respectively the variables LISTA, LISTECUT and LISTK). For instance, here we want to test the variation of the total energy of diamond using different energy cutoffs, so we put in LISTECUT the value '30 35 40 45 50 60', and the code will sequentially calculate the energy for each of the cutoffs 30

Ryd, 35 Ryd, 40 Ryd, etc. **You should change these lines, depending on which parameter you want to test.** But keep in mind that when you put a range for a certain parameter (e.g. here the cutoff), you should keep fixed the two other parameters to a single value (e.g. here we put only one value both in LISTA and LISTK).

Problem 1
Problem 1 will test the energy convergence with respect to energy cutoff. Look at the script, and look for the parameter *LISTECUT.* Erase everything on this variable after "30", that is, put *LISTECUT='30'.*
Now let's run the script. At the prompt, type

$ **sbatch run_C_scf.sh** and wait until the script is submitted and executed (this might take some time)

Now, look at the output file
$ **more C.scf.6.6.30.4.out**

Scroll through this file. The beginning will just recap the configuration that is being calculated. Then there is some information about the pseudo-potentials that PWSCF just read in. The next part tells you about intermediate energies that PWSCF calculates, before the calculation is converged. Near the end, there will be something like:

```
    !    total energy             =   -22.51880584 ryd
```

(your number may be slightly different). Note, the final occurrence of "total energy" will have an exclamation point by it, something you can use to hunt for it. You can skip to this right away by using search functions (in vi, type esc, /total energy, enter), or just scroll down a lot. This is your total energy, as calculated by PWSCF. Or you can type

$ **grep "! total energy" <filename here>** To make sure you have the correct spacings, cut

and paste the "! total energy" part
from your output file. You can also do this:

$ **grep "\!" *.out**

Which will grep all total energies from all files terminated by ".out" in the current directory. The "!" is a special character, which is negated by using "\".
At the end of the file, it will tell you how long your program took to run, and how much memory it used. You should start to develop a feel for how long your runs take, and how much memory they will use.

There are also lines that record the number of unique k-points that were calculated. The input may have a 4x4x4=64 k-point mesh, however some of these k-points have the same energy because of the symmetry of the crystal. 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

Now, increase the cutoff in the run_C_scf.sh file, and rerun the calculation. A good increase in cutoff would be ~10 ryd. To finish problem 1, put a list of different cutoffs in LISTECUT and check all the resulting energies.

Problem 2

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 Ryd (or some other cutoff that you have used already in Problem 1). If you have saved all of your output files, there should be no need to rerun an initial calculation.

There are some "cross effects" in testing cutoff and k-point separately, however we assume these are small.

To increase the size of the k-point grid, edit the script and change the values in *LISTK:* put '6 8' instead of '4'. Resubmit the script, and record the total energies for the two grids 6x6x6 and 8x8x8. 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, 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.

Problem 3

In problems 1 and 2, the forces on C are 0 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 C atom +0.05 (fractional) in the z direction. To do this, edit z coordinate of the C ion in the input file. After you edit the script, the ATOMIC_POSITIONS section will look something like this:

```
ATOMIC_POSITIONS
 C 0.00 0.00 0.00
 C 0.25 0.25 0.3
```

Now put '30' in *LISTECUT* so that the cutoff is 30 Ryd, and put '4' in *LISTK* so that the k-point grid is 4x4x4. Rerun the script, and record the forces. 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    0.25834978
   atom   2 type  1   force =    0.00000000    0.00000000   -0.25834978
   Total force =    0.365362      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 Ryd/bohr. Record this number, and retest the convergence issues with respect to cutoff energies.

Problem 6 and Problem 7

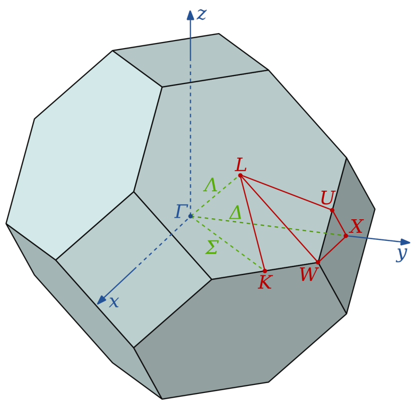
Given the information you learned above, you should be able to do problem 6 and 7. For problem 7, put only one value in both *LISTK* and *LISTECUT* (the converged values), and a range of values in *LISTA.*

Problem 8

Run the run_sibands.sh script and you'll get a number of output files. Then you must run "/apps/quantum-espresso/6.5.0/gcc-8.3.0_openmpi-3.1.4/bin/plotband.x" and follow the sequence show below. Highlighted values are ones you will enter.

```
--
prompt> /apps/quantum-espresso/6.5.0/gcc-8.3.0_openmpi-3.1.4/bin/plotband.x
Input file > bands.dat
Reading 8 bands at 36 k-points
Range: -5.6680 16.4950eV Emin, Emax > -6.0 10.0
high-symmetry point: 0.5000 0.5000 0.5000
high-symmetry point: 0.0000 0.0000 0.0000
high-symmetry point: 0.0000 0.0000 1.0000
high-symmetry point: 0.0000 1.0000 1.0000
high-symmetry point: 0.0000 0.0000 0.0000
output file (xmgr) > si.bands.xmgr
bands in xmgr format written to file si.bands.xmgr
output file (ps) > si.bands.ps
Efermi > 6.337
deltaE, reference E (for tics) 1.0, 6.337
bands in PostScript format written to file si.bands.ps
--
```

Then you can convert the ps file to a PDF using the command "ps2pdf si.bands.ps" and this will create a file called si.bands.pdf. Note that the vertical lines in this graph correspond to the high symmetry points called out during the interactive plotband.x portion described above. You should use this list of high symmetry points to label your graph. High symmetry points for the first Brillouin zone for an FCC material are shown and labeled below; the diamond cubic zone is nearly identical so you can use the same labeling approach.



FCC First Brillouin Zone (Wikipedia)

Problem 9

In this one you will find the lattice parameter of $BaTiO_3$ using the run_BaTiO3_latparam.sh script using a self consistent calculation. Then using the equilibrium lattice parameter from this script (lowest energy) you will distort the vertical (z position) of the Ti atom and relax the O atoms using the run_BaTiO3_distortTi.sh script. This is slow because it is an optimization routine. Once you have the minimum energy Ti distortion position, you'll run the run_BaTiO3_distortTiO.sh script using the lattice parameter and Ti distortion position found in previous steps and relax the position of the Ti and O atoms and find the final atom positions (available in the out file) and their corresponding energy.

FAQ for HW 2

I do not understand quantum mechanics at all.

General introductions to Quantum Mechanics
http://walet.phy.umist.ac.uk/QM/LectureNotes/QM.html
In particular look at Chapters 1, 2, 3, 8, 11. Operators, expectation values, bra-kets: Paragraphs 1.1 and 1.2 of
http://www-keeler.ch.cam.ac.uk/lectures/quant_letter.pdf
Or, more complete:
http://www.math.utah.edu/~gold/doc/quantum.pdf
Very simple primer:
http://www.chem1.com/acad/webtut/atomic/
Note, it is not always important to know every detail of quantum mechanics to run a quantum mechanics code.

How precisely do I need to get the lattice parameter?

Lattice parameters are typically listed to within 0.01 Angstroms. 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 C at 0 0 0 and at .25 .25 .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 E vs. lattice constant plot is jagged.

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

I don't like scripts.

Use scripts! They will save you a lot of time in the long run. This is anyway the only way to use the queuing system.

How do I kill my script?

Type squeue –u username (enter your username). This will tell you which jobs you are running – look at the numbers on the left. Type "scancel <number here>" to kill your job.

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

Yes. This is a "feature" of the code. Don't worry about it.

How is "convergence of energy" defined?

You say that your energy is converged to X Rydbergs when Etrue-En=X (n is the current energy). How do you know Etrue? 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 Etrue. The most important thing is that you do NOT define convergence as En+1-En, where n is a step in k-point, energy cutoff, 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 2x2x2 k-grid may be the same as the energy at a 8x8x8 k-grid, but the energy at a 4x4x4 might be very different from both of these. In this case, you aren’t really converged at a 2x2x2 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 1mm. 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 1mm.
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 ~5 meV/atom. From experience, we also know that a good error on forces is 10 meV/Angstrom. 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.