Lab 2: Handout
PWSCF: a first-principles energy code

We will be using the PWSCF code as our first-principles energy code. PWSCF is a first-principles energy code that uses pseudopotentials (PP) and ultrasoft pseudopoentials (US-PP) within density functional theory (DFT). It includes linear response methods, which can be used to calculate phonon dispersion curves, dielectric constants, and Born Effective charges. It also implements third-order anharmonic perturbation theory. PWSCF is free for academics. Further information (including online manual) can be found at the PWSCF website http://www.pwscf.org/. PWSCF is currently at version 1.2.0. We will be using version 1.1.2, which can be found here: http://www.pwscf.org/download.htm#previous.

There are many other first-principles codes that one can use. These include:

ABINIT
http://www.abinit.org
DFT Plane wave pseudopotential (PP) code. ABINIT is extremely well coded, and free for academics. It includes linear response methods.

VASP
http://cms.mpi.univie.ac.at/vasp/
DFT Ultra-soft pseudopotential (US-PP) and PAW code. US PP's are faster than corresponding PP codes, with similar accuracy. An industry standard. Small 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. Nice interface. Small cost for academics (~$500)

Gaussian
http://www.gaussian.com
Hartree-Fock (HF) code with DFT also. One of the most popular HF codes. Calculates numerous properties of interest to chemists. Moderate cost for academics (~$3000)

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

This is a tutorial on how to get **energies** using PWSCF.

Some helpful conversions:

1 **bohr** = 1 **a.u.** (atomic unit) = 0.529177249 **angstroms**.
1 **Rydberg** (R_∞) = 13.6056981 **eV**
1 **eV** =1.60217733 x 10^{-19} **Joules**

SINGAPORE STUDENTS

Your lab instructors will help you connect to the computers. They will help you copy (or link) the pw.x program and all of the input files.

CAMBRIDGE STUDENTS

Your lab instructors will help you connect to Armageddon, and forward you to the appropriate node.

MIT STUDENTS

Log onto Athena.
Add the course locker.

```
Athena$ add 3.320
```

Next create a directory to store your files (ask if you need help). For example, Assuming you've already created the 3.320 directory,

```
Athena$ cd ~<your username here>
Athena$ cd 3.320
Athena$ mkdir LAB2
Athena$ cd LAB2
Athena$ mkdir PROBLEM1
Athena$ cd PROBLEM1
```

In this lab, you do not want to copy the pw.x file, because it is too big. Link it instead:

```
Athena$ cp ~3.320/LAB2/GaP.scf.in .
Athena$ ln –s ~3.320/bin/pw.x pw.x
```

Note: Don't forget a space between GaP.scf.in and the period. The period at the end of cp "~3.320/LAB1/gulp1* ." must be typed. It means you are

copying the files to your current directory. You can also copy over the gulp file , instead of linking.

MIT and Singapore, and Cambridge students

Problem 1 concerns convergence issues in first-principles calculations. They are sometimes called *ab initio* calculations, which means "from the beginning" in Latin. They are called this because (in theory) first-principles calculations only rely on the atomic number and build up everything from there. Below is some background for problem 1A and 1B; if you do not think you need it, skip to the section which titled "summary of background".

Background for problem 1A

Remember that we are dealing with infinite systems (aka periodic boundary conditions). This means that we can use the Bloch theorem to help us solve the Schrödinger equation. The Bloch theorem says:

$$\psi_{n\vec{k}}(\vec{r}) = \exp(i\vec{k}\cdot\vec{r})u(\vec{r})_{n\vec{k}}$$

with

$$u_{n\vec{k}}(\vec{r}) = \sum_{G} c_G \exp(i\vec{G}\cdot\vec{r})$$

In these equations, $\psi(\vec{r})$ is the wavefunction, $u(\vec{r})$ is a function that is periodic in the same way as the lattice, $\vec{G}$ sums over an infinite number of reciprocal lattice vectors, and c_G's are coefficients in the expansion. In this case, our *basis functions* (ie what we expand in) are planewaves, or exponentials. They are called "plane waves" because surfaces of constant phase are parallel *planes* perpendicular to the direction of propagation.

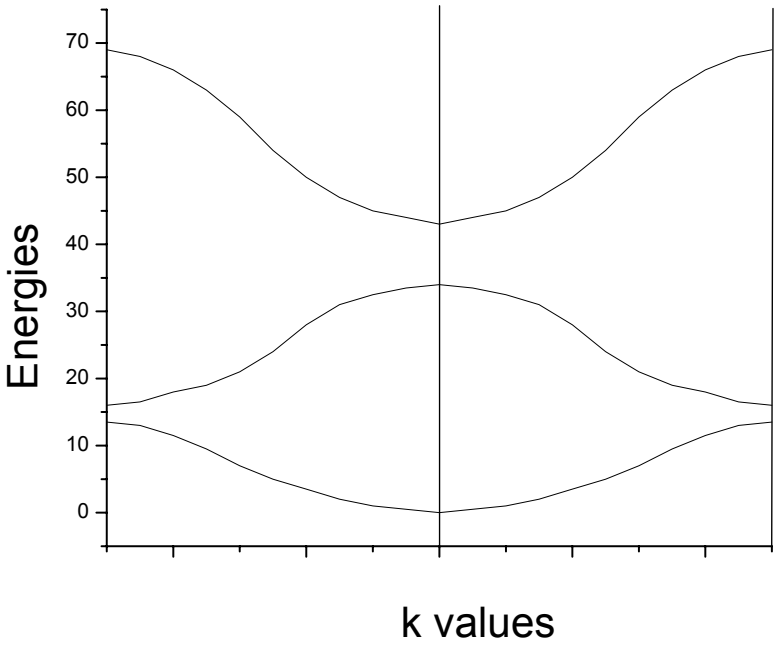
In actual calculations, we have to stop the expansion at some point (i.e. stop taking more $\vec{G}$'s). In first-principles calculations, this is called the *energy cutoff*. Cutoffs are always given in energy units such as Rydberg or eV.

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 will be to test cutoff convergence issues. You can always take a higher cutoff than you need, but the calculation will take longer.

Background for Problem 1b.

Because of the Bloch theorem, instead of solving the Schroedinger equation with an infinite number of MxM matrices (M is the number of basis functions), you solve it for a finite number of $\vec{k}$ values, and get a value for E at each $\vec{k}$. 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, don't worry about it too much. Just know that to get a value for *E*, the energy of the crystal, you need to integrate the values of E over the first Brillouin zone, where the bands are occupied, and divide by the volume. Numerically, this means calculating sum over E for all $\vec{k}$ points, where the bands are occupied, then divide by the number of $\vec{k}$ points. Thus, summing over a finite number of $\vec{k}$ points is an approximation to performing an integral. You will need to make sure you have enough $\vec{k}$ -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 wavefunction expansion. The second is *number of* $\vec{k}$ *-points*, which measures how well your discrete grid has approximated the continuous integral.

The input files

We will look at the GaP.scf.in input file, which we will use later for problem 1. This is an input file for GaP. GaP is a face-centered cubic structure with Ga at 0 0 0 and P at 0.25 0.25 0.25. These atoms are at special positions, and fixed by symmetry. Ga and P can be switched if desired. GaP looks like this.

(Image taken from Inorganic Chemistry by Shriver, Atkins and Langford)

In our case Zn=P and S=Ga (we have switched positive and negative ions from this picture).

Look at the input file, which you have just copied over to your directory.

To look at this file, type

```
Athena$ less gaas.scf.in
```

You can scroll through the file by typing space (to go forward), b (to go backwards) or q (to quit).

The file will look something like this:

```
1     GaP
2     GalliumPhosphide
3      &input
4     ibrav=  2, celldm(1) =10.3043, nat=  2, ntyp= 2,
5     pseudop(1) = 'Ga.gon',
6     pseudop(2)='P.gon',
7     pseudo_dir='/afs/athena.mit.edu/course/3/3.320/LAB2/,
8     tmp_dir='.',
9     ecut(1) =5.0,
10     beta(1) = 0.7,
11     tr2 =  1.0d-12,
12     lforce=.true., lstres=.true.,
13     &end
14    0.00 0.00 0.00 1
15    0.25 0.25 0.25 2
16    'Ga' 1 1 1.0
17    'P' 1 2 1.0
18    0
19     3 3 3
20     0 0 0
```

Lines numbers are added for reference.

Line 1:

This is title information.

Line 2:

This is a label. You can label what you want, the program doesn't care.

Line 3:

&input starts the main input information, which contains flags and a lot of other stuff.

Lines 4-13

These are different keywords. Note that commas separate them all.

ibrav – ibrav gives the crystal system. Ibrav=2 is a F-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 INPUT_PW file.

celldm – celldm defines the dimensions of the cell. **You will be changing this parameter**. celldm is in atomic units, or bohrs. 1 Remember 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) =a_0. In cubic systems, a=b=c=a_0. Consult INPUT_PW for other crystal systems.

nat – number of atoms (each individual unique atom)

ntyp – number of types of atoms EXAMPLE: for $BaTiO_3$, nat=5, ntyp=3.

pseudop(1) and pseudop(2) - array of filenames which contain pseudopotential 1 and 2

pseudo_dir - location of pseudopotentials directory.

tmp_dir – temporary directory.

ecut – Energy cutoff for pseudopotentials. **This one is important you will be changing this parameter.**

beta - Mixing factor. Don't worry about this for now.

tr2 - Threshold for self-consistency. Don't worry about this for now.

lforce – Whether or not to calculate forces on atoms.

lstress – Whether or not to calculate stresses on atoms (for volume relaxations)

&end – end input flag information

Lines 14-15
This are the fractional positions of the atoms, and the type. So,
Atom #2 is in position x=0.25 y=0.25 z=0.25 and is type 2 (which is As)
You will edit this later.

Lines 16-17
This is the atom label and which pseudopotential type it corresponds to.

Line 18
This is the label for kpoints. A '0' means you will be inputting a $\vec{k}$ -point grid, instead of inputting individual $\vec{k}$ -points.

Line 19-20

This is the $\vec{k}$ -point grid (2x2x2) and offset (0 0 0). **You will be changing the $\vec{k}$ -point grid.** Don't worry about the offset.

You can read the documentation for the input file in the

INPUT_PW file. For Cambridge students, this will be in the HW2 directory. For MIT students, this will be in the LAB2 directory of the locker. Singapore students will be given this file by the TA.

You should skim this.

Running the code

To run the program type

```
Athena$ pw.x <[filenamein]>[filenameout]
```

In this case, type

```
Athena$ pw.x <GaP.scf.in>GaP.scf.out
```

Organizing your runs

You can organize your runs any way you like, or don't have to organize them at all. One way is to make directories for each problem you do, and name your output files accordingly. Good organization may save you headache in the long run (but is totally up to you).

Problem 1

Problem 1 will test the energy convergence with respect to energy cutoff.
Look at the input file, and look for the parameter *ecut*. The initial setting for *ecut* is 5 Rydbergs.

Now let's run PWSCF. At the prompt, type

```
Athena$ pw.x <GaP.scf.in>GaP.scf.out.05
```

(you can name the output whatever you like). Now, look at the output files.

```
Athena$ less GaP.scf.out.05
```

Scroll through this file. The beginning will just recap the configuration that is being calculated. Then there is some information about the pseudopotentials 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      =      -17.30666196 ryd
```

(your number may be slightly different). Note, the final occurance 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

```
Athena$ grep "!    total energy"
```

To make sure you have the correct spacings, cut and paste the ! total energy part. At the end of the file, it will tell you how long your program took to run, and how much memory it used.

```
  PWSCF        :    50.80s CPU time
 init_run     :     2.30s CPU
   electrons    :    41.04s CPU

   electrons    :    41.04s CPU
   c_bands      :    36.92s CPU (       6 calls,   6.153 s avg)
   sum_band     :     3.91s CPU (       6 calls,   0.652 s avg)
   v_of_rho     :     0.07s CPU (       7 calls,   0.010 s avg)
     .... etc..
Dynamical memory:   4.70Mb current,   8.28Mb maximum
```

Your numbers may be slightly different from these. You should start to develop a feel for how long your runs take, and how much memory they will use.

There will also be a line which says:

```
Cartesian axes

     site n.     atom                  positions (a_0 units)
         1        Ga     tau( 1) = (  0.00000  0.0000  0.0000  )
         2        P     tau( 2) = (  0.2500  0.2500  0.2500  )

     number of k points=    3
                       carth. coord. in units 2pi/a_0
        k(   1) = (   0.0000   0.0000   0.0000), wk =   0.2500000
        k(   2) = (   0.5000  -0.5000   0.5000), wk =   1.0000000
        k(   3) = (   0.0000  -1.0000   0.0000), wk =   0.7500000
```

This records the number of unique k-points that were calculated. The input here has a 2x2x2=8 kpoint mesh, however some of these kpoints 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 gaas.scf.in file, and rerun the calculation. A good increase in cutoff would be ~5 ryd. It is a good idea to save each output file separately. That is, run

```
Athena$ pw.x <GaP.scf.in>GaP.scf.out.10
```

If you have increased the cutoff to 10 Ryd. To finish problem 1, repeat and continue this procedure.

Problem 2

Problem 2 will test the convergence of the energy with increasing $\vec{k}$ -point grid. We will be testing $\vec{k}$ -point grid convergence and cutoff convergence separately. So, set your cutoff to something lower, such at ~15 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 $\vec{k}$ -point separately, however we assume these are small.

To increase the size of the $\vec{k}$ -point grid, change *3 3 3* to *4 4 4*. This is on the 19th line (second to last). Rerun pwscf, and record the total energy.

You will also want to record the number of unique $\vec{k}$ -points (that is, unique by symmetry). This is near the beginning, and looks something like this:

```
Cartesian axes

     site n.     atom                  positions (a_0 units)
         1        Ga     tau( 1) = (  0.0000  0.0000  0.0000  )
         2        P      tau( 2) = (  0.2500  0.2500  0.2500  )

     number of k points=   4
```

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

Problem 3 and Problem 4

In problems 1 and 2, the forces on Ga and P 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 the P atom +0.10 (fractional) in the z direction. To do this, edit z coordinate of the P anion in the input file. After you edit the file, the bottom will look something like this:

```
…
&end
0.00 0.00 0.00 1
0.25 0.25 0.35 2
'Ga' 1 1 1.0
'P' 1 2 1.0
0
 3 3 3
 0 0 0
```

Now edit the cutoff so that it is ~15 Ryd, and make the $\vec{k}$ -point grid ~3x3x3. Rerun PWSCF, 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:

     atom   1 type  1 force =   0.000  0.000    0.04597661
     atom   2 type  1 force =   0.000  0.000   -0.04597661

     Total force =  0.091953  Total SCF correction =     0.000009
     forces       :     0.03s CPU
```

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 cutoffs and $\vec{k}$ -points.

Problem 6 and Problem 7

Given the information you learned above, you should be able to do problem 6 and 7. Good luck!

Scripting

To speed up calculations, we can use scripts.

```
1  #!/bin/sh
2  for ecut in   60 65 70
3  do
4     rm -rf fileigk filewfc  flmixpot
5     cat>GaP.in<<EOF
6       GaP
7       GalliumPhosphide
8       &input
9         ibrav=  2, celldm(1) =10.3043, nat=  2, ntyp= 2,
10        pseudop(1) = 'Ga.gon',
11        pseudop(2)='P.gon',
12        pseudo_dir = '/home/mit3320/HW2',
13        tmp_dir='.',
14        ecut(1) =$ecut,
15        beta(1) = 0.7,
16        tr2 =  1.0d-12,
17        lforce=.true., lstres=.true.,
18     &end
19     0.00 0.00 0.00 1
20     0.25 0.25 0.25 2
21     'Ga' 1 1 1.0
22     'P' 1 2 1.0
23     0
24     3 3 3
25     0  0 0
26    EOF
28   pw.x<GaP.in>GaP.out.$ecut
29 done
```

Line 1: This just tells the computer what kind of script you are using

Lines 2, 3, and 27 – This sets up a for-do loop. The numbers "60", "65", and "70" mean it will run energy cutoffs of 60, 65, and 70 Ryd.. Those numbers are assigned to the variable "ecut". The word "done" means the end of the loop

Lines 5,26 – This means to put what follows (lines 6-25) into the GaP.in file, until the EOF is reached (on line 26).

Lines 6-25 – this is just the Ga.scf.in file, except for line 14, which has the variable ecut. Note to denote the value of ecut, we put a dollar sign in front of it ($ecut). Remember we are changing the $\vec{k}$ -grid between each run. Every time the script loops through, it substitutes a value for $ecut that it gets from line 2.

Line 26 – This runs pw.x for each file, and gives each file a different output name (in this case GaP.out.65, GaP.out.65, and GaP.out.70)

The script can be called by running the script name. For instance, this script is ecut.script. If your computer is giving you a "bad interpreter" error, type
"chmod u+x <scriptname here>

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 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.sfu.ca/chemcai/QUANTUM/Quantum_Primer.html

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 Ga at 000 and P 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, and to remove the output files (which give wavefunction information, etc..) between runs.

I don't like scripts.

Use scripts! They will save you a lot of time in the long run.

How do I kill my script?

Type ps aux. This will tell you which jobs you are running- look at the numbers on the left. Type "kill -9 <number here>" to kill your job. You will need to kill the job AND the script (kill the script first).

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.

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 energies is ~5 meV.
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.