Lab 3 Handout:
Materials properties with Quantum Espresso - PWscf

March 14, 2019

Continuing with Quantum Espresso: PWscf

In this lab we continue our use of the Quantum-Espresso(QE) package, introduced in Lab 2. Please refer to the Lab 2 handout. Please keep in mind that this time calculations are more complex and longer than the previous problem set.

1 Study of Iron BCC and HCP structures

In problem 1, we will compare energy of Iron (Fe) in the body centered cubic (BCC) and hexagonal close packed (HCP) structures at different pressures, in order to evaluate the transition pressure between the two phases [for details see "Possible thermal and chemical stabilization of body-centered-cubic iron in the Earth's core", Vocadlo L. et al. *Nature*, **424**(2003)536; Xianwei Sha and RE Cohen *PRB*,**73**(06)104303]. Moreover, we will calculate the ferro-, anti-ferro- and para- magnetic ground states.

First we will calculate the total energy of a bulk BCC Fe and HCP Fe. To get started the setup steps and changes to the input file for problem 1 are outlined below:

1. Pull the latest samples and scripts from the git repository and copy to your work directory.

```
$ cd ~/Software/labutil
$ git pull
$ cp * ~/WORK/Lab3/
```

2. Copy the pseudopotential from `labutil` into the correct directory:

```
$ cd ~/WORK/Lab3/
$ cp Fe.pbe-nd-rrkjus.UPF  $ESPRESSO_PSEUDO
```

The bash environment variable `$ESPRESSO_PSEUDO`, defined in your `.profile`.

3. Let’s look at the sample input file `Fe.scf.in`

```
  &control
    calculation=’scf’
    pseudo_dir = ...
    outdir = ...
    tstress = .true.
    tprnfor = .true.
    disk_io = ’none’
 /
 &system
    ibrav = 0
    nat=  ...
    ntyp= 1
    ecutwfc=  30.0
    ecutrho= 240.0
    nspin=2
    starting_magnetization(1) = 0.7
    occupations=’smearing’
    smearing=’mp’
    degauss=0.02
/
&ELECTRONS
  diagonalization= ’david’
  conv_thr       = 1e-07
  mixing_beta    = 0.5
/
&IONS
/
&CELL
/
ATOMIC_SPECIES
 Fe  55.847 Fe.pbe-nd-rrkjus.UPF
CELL_PARAMETERS {angstrom}
...
ATOMIC_POSITIONS {angstrom}
  Fe ...
  Fe ...
K_POINTS {automatic}
2 2 1  0 0 0
```

The following describes some new input parameters (not seen in Lab 2) as well as the parameters we are concerned with varying in Lab 3. As mentioned in the Lab 2 handout the full description of all parameters in the input file can be found on the Quantum Espresso website.

-
  ```
  ecutwfc=  30.0
  ecutrho= 240.0
  ```

  Since we are using ultrasoft pseudopotentials (USPP), charge density needs to be represented on a larger grid in reciprocal space, with a cutoff at least 8-12 times larger than the kinetic energy cutoff for the wavefunctions. You will notice in the scripts that the multiplier is set to 8. Just as a note, for norm conserving pseudopotentials (NCPP) it is sufficient to set the multiplier to 4.

-
  ```
  nspin=2
  ```

  This is the spin-polarization of the system. In previous calculations this was set to 1 (the default) for a non-spin-polarized calculation, i.e. bands are occupied with 2 electrons each and there is no separate calculation for spin-up and -down electrons. However, if we are to treat magnetic systems we need to take spin-polarization into account, this is turned on by setting `nspin=2`.

-
  ```
  starting_magnetization(1) = 0.7
  ```

  Because we want to search for a magnetic ground state, we put the system in an initial magnetic state with `starting_magnetization(1)`. Here the integer 1 refers to the atom *type* and the value 0.7 (can range for -1 to $+1$). And since we have defined only one type (Fe) we can only find either a ferromagnetic or a non-magnetic state. You can imagine that other types of magnetism will require different definitions of both `starting_magnetization(1)` and atom types. One beauty of computation is that you can give many different *types* of the same atom – e.g., Fe1, Fe2, etc. In the iron helper script, we have given one atom in the unit cell the label "Co", even though it is assigned an iron pseudopotential. This allows you to add "starting_magnetization(2)" to the input file to make the structure antiferromagnetic.

-
  ```
  occupations='smearing'
  smearing='mp'
  degauss=0.02
  ```

  Because we are concerned here with a metallic system, we run into some issues with Brillouin zone integration. More specifically, in a metal bands are occupied up to the Fermi energy and because of this discontinuity, Brillouin zone integration with our rather sparse **k**-point grids will lead to large discrepancies in energy with regard to **k**-point sampling (think about if the band crossed the Fermi level at a **k**-point that we have sampled, versus somewhere in between sampled points). To combat this problem (without going to enormous expensive **k**-point grids) we introduce a fictitious temperature, or "smearing". This results in fractional occupation of some of the states above the Fermi-energy and in effect make for a smoother function for the integration. To that end we use a few more parameters: First the temperature is turned on (with `occupations='smearing'`), then we decide on a type of smearing (in this case `smearing='mp'`), and finally we need to define a parameter that determines the 'amount' of smoothing (`degauss=0.02`). This value is in Ryd units and is the spread of the smearing function. `degauss` should always be tested; small enough not to introduce unphysical effects, but large enough to decrease **k**-point sampling. However, in the interest of time we will not do so for this Lab.

  - For the BCC case the structure needs only one atomic position

    ```
    ATOMIC_POSITIONS {angstrom}
      Fe 0.000000000000 0.00000000000000 0.0000000000000
    ```

    while for the HCP case, two will be needed.

  -
    ```
    K_POINTS {automatic}
    2 2 1  0 0 0
    ```

    Remember that **k**-point sampling is related to the inverse of the cell parameters. That is, for BCC, sampling should be over $2 \times 2 \times 2$, etc. grids, while for the HCP case the sampling should be $2 \times 2 \times 1$, $4 \times 4 \times 2$, etc. to reflect the difference in c versus a lattice parameters as closely as possible, in order to get a more uniform sampling mesh.

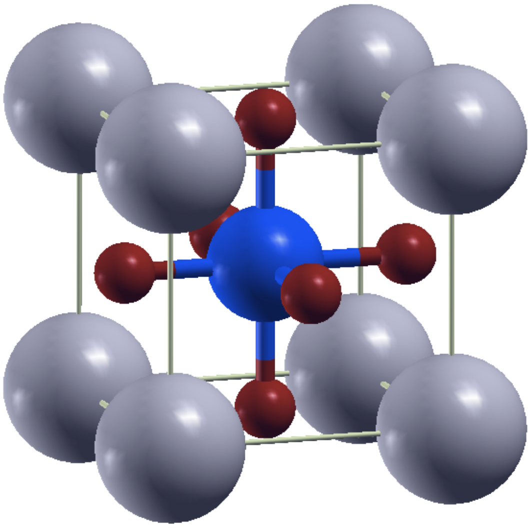
4. To run the simulation manually you can type

   ```
   $ pw.x < Fe.scf.in > Fe.scf.out
   ```

5. The script `Fe_sample.py` provides an example Python workflow using familiar functions, to help you run a series of calculations automatically. You will notice that the parameter dictionary now includes the new keywords. Note the new parameter `ncpu=2` in the `run_qe_pwscf()` function, it specifies how many cpu cores the PWscf code will use. Since some of your computations will take a few minutes, you can speed up the computations by allocating more cores to your virtual machine (in VirtualBox Settings / System / Processor) before starting it. On Google Cloud, your instance may have more cores available, depending on how you set it up.

2 Stability of perovskite Lead Titanate

In this problem, we will be looking at $PbTiO_3$ in the cubic phase and tetragonal phases. The atomic positions of the cubic perovskite structure are shown below (corners of the cube are Pb, faces contain the O cages, and center of cell is Ti).



Take a look at the provided example file `PbTiO3.rel.in`. The order of the keywords in each block is not important and may be different in your inputs. However, the order of the blocks `&control`, ..., `&cell` is important:

```
&control
   calculation = 'relax'
   tstress = .true.
   tprnfor = .true.
   pseudo_dir = ...
   outdir= ...
/
&system
   ibrav =  0
   nat =  5
   ntyp = 3
   ecutwfc = 30.0
   ecutrho = 240.0
/
&electrons
   diagonalization='david'
   mixing_beta = 0.7,
   conv_thr =  1e-07
/
&ions
/
&cell
```

```
 /
ATOMIC_SPECIES
  Pb 207.2 Pb.pz-d-van.UPF
  Ti 47.88 Ti.pz-sp-van_ak.UPF
  O 15.9994 O.pz-rrkjus.UPF

CELL_PARAMETERS {angstrom}
 3.9 0.0 0.0
 0.0 3.9 0.0
 0.0 0.0 3.9

ATOMIC_POSITIONS {angstrom}
  Pb 0.00000 0.00000 0.00000 0 0 0
  Ti 1.95000 1.95000 1.95000 0 0 0
  O 0.00000 1.95000 1.95000
  O 1.95000 0.00000 1.95000
  O 1.95000 1.95000 0.00000

K_POINTS {automatic}
  3 3 3 1 1 1
```

-
  ```
     calculation = ’relax’
  ```

  For parts (B) and (C), instead of a single self-consistent field calculation, we will be doing a `’relax’` calculation. This includes a series of SCF calculations, where each calculation finds all the forces and displaces the ions (those that are not explicitly constrained) until forces are small. The ions are allowed to move in order to reduce the total system energy. Note that for part (A), you should use `’scf’`, as in Problem 1.

-
  ```
  &ions
     ion_dynamics=’bfgs’
  /
  ```

  Since we will be using ion relaxation, we now use the `IONS` name list section. We are not, however, using real dynamics – i.e., there is no time coordinate used in the relaxations – but just searching for the minimum energy relaxations. This section is ignored by the code for the scf calculations of part (A). If interested in more `ion_dynamics` flags (other than BFGS) see the full input file description.

-
  ```
  Pb 0.0 0.0 0.0 0 0 0
  Ti 0.5 0.5 0.5 0 0 0
  ```

  You will notice that there are now three additional flags at the end of our atomic coordinates. These flags define the degrees of freedom available for those atoms during relaxation (0 = disallow motion in that direction for that atom, the default 1 = allow motion in that direction). In the example shown above, the Pb and Ti ions are fixed, while the O ions are allowed to relax. The format is:

```
atomic label  pos_x  pos_y  pos_z  allow_x  allow_y  allow_z
```

Finally, you will find that using Python scripts will save you tons of time on this problem set – i.e., not using scripts could mean that you spend literally days sitting at a computer waiting for runs to finish. You should be able to set up the appropriate scripts based on the ones we've already used for examining Fe and also the example script from last problem set. If you are still uncomfortable with writing your own Python functions, please ask for help from other students or make an appointment to work out some basic scripts for completing this problem set.

3 Metallic Alloy: the case study of CuAu

Metallic alloys form a large class of scientifically interesting and technologically important materials. Properties such as structure, phase stability, elastic behavior, magnetism and conductivity, are all related directly to the underlying electronic structure, see [S. Muller *J. of Physics: Condensed Matter* **15** (2003) R1429].

In this problem we will calculate the structural formation energy of the alloy AuCu, which is one of the most studied alloys due to its practical applications in catalysis. Moreover, it is a typical superlattice alloy with a rich phase diagram containing ordered and disordered phases across the range of compositions. The chemical compositions Cu_3Au, CuAu and $CuAu_3$, present ordered structures in a wide range of temperatures. Here we will focus on the case of CuAu, meaning a 50-50% solution of Au and Cu. The ordered phase CuAu is represented by a body-centered-tetragonal unit cell with two atoms (one Au and one Cu). This structure and this chemical composition of binary alloy is usually called $L1_0$.

For a metallic alloy, one of the most important quantities is the structural formation energy, defined as:

$$\Delta H_f(A_mB_n) = \frac{1}{m+n}(E^{tot}(A_mB_n) - mE_A^b(a_A) - nE_B^b(a_B)) \tag{1}$$

where $E_A^b(a_A)$ and $E_B^b(a_B)$ are the total energies for the elements A and B in their bulk, calculated for their equilibrium lattice constants a_A and a_B, respectively. The total energy of the binary alloy is $E^{tot}(A_mB_n)$, must correspond to the fully relaxed geometric configuration. Plotting the structural formation energy as a function of the concentration c, defined as $c = m/(n+m)$, we can evaluate phase stability at each concentration. Keep in mind that, even if the pure elements A and B have the same Bravais lattice, their alloys might have different symmetry, depending on differences in atom sizes.

First, you will calculate the total energies for the pure energies for the pure elements Au and Cu. Use your knowledge of PWscf and ASE to set up and converge these calculations. You can choose to scan over the lattice parameters to find the minimum, or use automatic variable-cell relaxation in PWscf. Your input file for Cu may look like this:

```
  &control
    calculation=’vc-relax’
    pseudo_dir = ...
    outdir= ...
    tstress = .true.
    tprnfor = .true.
 /
 &system
    ibrav = 0
    nat= 1
    ntyp= 1
    ecutwfc = 40.0
    ecutrho = 320.0
    occupations=’smearing’
    smearing=’mp’
    degauss=0.02
 /
 &electrons
    diagonalization=’david’
    conv_thr = 1e-07
    mixing_beta = 0.2
 /
 &ions
    ion_dynamics = ’bfgs’
 /
 &cell
    cell_dynamics = ’bfgs’
/
ATOMIC_SPECIES
 Cu 63.546 Cu.pz-d-rrkjus.UPF
CELL_PARAMETERS {angstrom}
 0.0 2.0 2.0
 2.0 0.0 2.0
 2.0 2.0 0.0
ATOMIC_POSITIONS {angstrom}
  Cu 0.00000 0.00000 0.00000
K_POINTS {automatic}
 8 8 8 0 0 0
```

- The keyword `calculation='vc-relax'` tells the code to optimize cell parameters as well as atomic positions using the Hellman-Feynman theorem approach. For this to work, you need to explicitly include the `&ions` and `&cell` blocks in the input, which by default set the minimization algorithm to 'bfgs' (listed above explicitly). An important point is that PWscf will first determine all the symmetry operations in the structure (you will see 48 symmetry operations for an fcc cell), and the entire set of scf and relaxation calculations will be constrained according to the initial symmetry.

- Use LDA pseudopotentials `Au.pz-d-rrkjus.UPF` and `Cu.pz-d-rrkjus.UPF`, downloaded them from the QE website pseudopotential repository. When you are interested in comparing energies for different systems, always be sure that pseudopotentials use the same exchange correlation functional, and that the same pseudopotentials are used for the same elements.

The CuAu compound, which has alternating layers of Cu and Au, should be represented as a body-centered tetragonal unit cell with a basis of two atoms. Make use of ASE to construct the cell, and visualize it with VESTA to make sure it looks right. You may want to use the VESTA menu option "objects / boundary" and set the supercell size at 2x2x2 for visualization to get a clearer view of the repeated unit cell. A sample input file `AuCu.vcr.in` is provided for your reference in the `~/Software/labutil/lab3_samples` repository. You will need to do all the necessary scripting and convergence testing yourself. Ask for help if you get stuck.