Tutorial Handout
Quantum-ESPRESSO: a plane-wave pseudopotential code

In this tutorial, we will be using the Quantum-Espresso package as our first-principles plane-wave pseudopotential code. We will be searching for the equilibrium zero-temperature lattice parameter of a silicon crystal and see how changes in k-point grid sampling and plane-wave energy cutoff will affect our results.

Quantum-Espresso is a full *ab-initio* package implementing electronic structure and total-energy calculations, linear response methods (to calculate phonon dispersion curves, dielectric constants, and Born effective charges) and third-order anharmonic perturbation theory. Quantum-Espresso also contains molecular-dynamics codes for running Car-Parrinello or Born-Oppenheimer Molecular Dynamics. Today we will be using Quantum-Espresso to perform a series of total-energy calculations to demonstrate convergence behavior of a system with respect to various input parameters. Note that the code 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. Quantum-Espresso is currently at version 4.0 and is under very active development.

There are many other first-principles plane-wave pseudopotential codes that one can use. Other popular packages include ABINIT (http://www.abinit.org), CPMD (http://www.cpmd.org), and VASP (http://cms.mpi.univie.ac.at/vasp/).

Background

Understanding plane-wave cutoffs and k-point sampling

For all first-principles plane-wave pseudopotential 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* $\vec{k}$ *-points*, which measures how well your discrete grid has approximated the continuous integral over the Brilluion Zone.

Remember that higher cutoffs (k-point and plane-wave cutoffs) translate to higher accuracy, but these calculations will take longer to run. For practical applications, you need to find an acceptable balance between time and accuracy.

Convergence with respect to plane-wave basis

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:

$$\psi_{n\vec{k}}(\vec{r}) = \exp(i\vec{k}\cdot\vec{r})\, u_{n\vec{k}}(\vec{r})$$

with

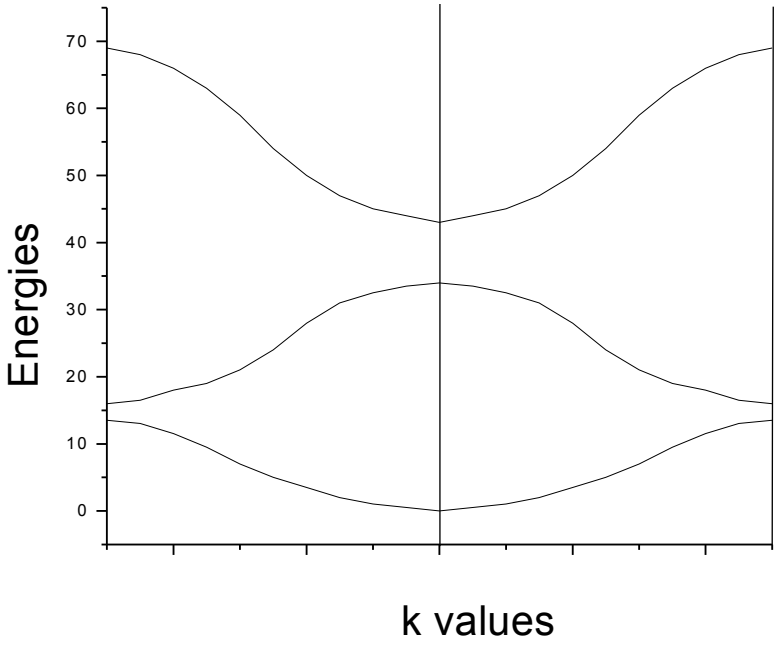
$$u_{n\vec{k}}(\vec{r}) = \sum_{G} c_G \exp(i\vec{G}\cdot\vec{r})$$

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, $\vec{G}$ sums over all (at least in principle) reciprocal lattice vectors, and c_G's are coefficients in the expansion. In our 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 by limiting the plane wave expansion to the discrete set of $\vec{G}$ 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.

It is important to remember that the plane-wave basis is only complete in the limit of an infinite number of $\vec{G}$ vectors. However, in actual calculations, we have to limit the plane wave expansion at some point (i.e. stop taking more $\vec{G}$'s). The point at which we truncate the basis set is called the *plane-wave cutoff*. Cutoffs are always given in energy units (such as Rydberg or eV) corresponding to the kinetic energy of the highest included $\vec{k}+\vec{G}$.

Convergence with respect to k-point sampling

A consequence of the Bloch theorem is that for an infinite periodic system, we need to solve a Schrödinger-like Kohn-Sham equation everywhere in the Brillouin Zone. This involves self-consistent iterative diagonalization of an M x M matrix, where M is the number of basis functions. A schematic band diagram of the first Brillouin Zone of a sample crystal is shown below.



To obtain a value for the energy of the crystal, we need to integrate over the first Brillouin zone, where the bands are occupied (and divide by the volume). However, to get the exact value for the integral, we would need to have the band energies at every possible reciprocal lattice point in the Brillouin Zone. So we instead replace the integral over the Brillouin Zone with a finite sum over discrete $\vec{k}$ -points, which is designed to reproduce the integral to a good approximation. However, you will need to make sure you have enough $\vec{k}$ -points to have a converged value for the total energy of the system. Commonly used methods for discretely sampling the Brillouin Zone with a finite number of $\vec{k}$ values include the "special-point" scheme and the Monkhorst-Pack mesh, which were described in the lectures.

Using Quantum-ESPRESSO

Getting started

Create a directory for your workshop output files, e.g.,

```
$ cd ~
$ mkdir QE
$ cd QE
```

Next, copy sample files.

```
$ cp <sample files directory>/* .
```

You should now have the sample script and pseudopotentials in your workshop directory. Take a look at the sample script file called `Si.sample.sh`. This file performs calculations at three different values for the plane-wave cutoff (25, 30, and 35 Ry). You should use this sample script as a template for your own scripts during this tutorial.

Note that using scripts will save you lots of time on calculations such as these. Not using scripts would mean that you could spend literally days sitting at a computer waiting for runs to finish.

Understanding the sample script

The contents of `Si.sample.sh` are reproduced here:

```
(1)  #!/bin/sh
(2)  ######################################################
(3)  # This is a sample script to run scf total-energy
(4)  # calculations on a unit cell of BaTiO3 using three
(5)  # different values for the input parameter
(6)  # 'ecut' (the plane-wave cutoff).
(7)  #
(8)  # Created by Brandon Wood, 21-07-08
(9)  #
(10) # You should copy this file and modify it as
(11) # appropriate for the tutorial.
(12) #
(13) ######################################################
(14) # Important initial variables for the code
(15) # (change these as necessary)
(16) ######################################################
(17)
(18) NAME='ecut'
(19) RUNDIR='$HOME/QE'
(20)
(21) ######################################################
```

```
(22)
(23) for CUTOFF in 5 10 15
(24) do
(25) cat > $RUNDIR/$NAME.$CUTOFF.in << EOF
(26) &control
(27)    calculation = 'scf',
(28)    prefix = '$PREFIX',
(29)    tprnfor = .false.,
(30)    pseudo_dir = '$RUNDIR',
(31)    outdir = '/tmp/'
(32) /
(33) &system
(34)    ibrav =  2,
(35)    celldm(1) = 10.0,
(36)    nat =  2,
(37)    ntyp = 1,
(38)    ecutwfc = $CUTOFF,
(39)    nspin = 1
(40) /
(41) &electrons
(42)    mixing_beta = 0.7
(43) /
(44)
(45) ATOMIC_SPECIES
(46) Si 28.086 Si.pbe-rrkj.UPF
(47)
(48) ATOMIC_POSITIONS (alat)
(49) Si 0.0 0.0 0.0
(50) Si 0.25 0.25 0.25
(51)
(52) K_POINTS (automatic)
(53)  3 3 3 1 1 1
(54) EOF
(55)
(56) $HOME/pw.x < $RUNDIR/$NAME.$CUTOFF.in > $NAME.
  $CUTOFF.out
(57)
(58) rm -R /tmp/Si_work*
(59) done
```

Comments on the script:

*An asterisk * indicates you the important fields that you may need to change.*

Line 1

This invokes the `sh` shell for running the script. DO NOT MODIFY THIS UNLESS YOU KNOW WHAT YOU ARE DOING.

Lines 2-16

The lines are informational comments only and will not be interpreted by the shell. Notice that comments are created by placing the # character at the beginning of a line.

****Line 18***

The variable `NAME` is the title you are giving to this set of calculations. By default, the script will generate input and output files that contain the value of this variable (see comment on Lines 26, 63). You should change this to something new (and descriptive!) for each calculation.

Line 19

The variable `RUNDIR` should contain the directory where you placed your tutorial files. Change this if you chose a subdirectory of your home folder other than '`QE`'.

****Lines 23-24, 59***

These lines initialize a `for...do...done` construction, which you can use in a shell script to iterate a variable over a set of values. In this case, the loop initializes a new variable called `CUTOFF`, has been set up to contain the current value (in Ry) for the plane-wave energy cutoff. In Line 23, `CUTOFF` is set to loop over three values—5, 10, and 15. Lines 24 and 59 designate the beginning and end of the instruction set to be executed upon each iteration of the loop.

Everything between Line 23 'do' and Line 59 'done' will be executed once for each of the three values of `CUTOFF`. Variables in shell scripts are referred to by prepending the `$` sign, so whenever '`$CUTOFF`' appears in the script (Lines 25, 38, 56), it will be replaced by its current value of 5, 10, or 15.

The sample script is designed to iterate over the plane-wave energy cutoff, but for the tutorial, you will also need to iterate over different values of the ***k***-point mesh and of the lattice parameter. You will need to modify these lines to loop over those variables instead.

****Line 25***

Creates an input file whose filename contains the variable `NAME` as well as the variable `CUTOFF`. Change this as appropriate if you use something other than the `CUTOFF` variable. If you are running a series of calculations in a loop, be sure to include the loop variable (in this case, `CUTOFF`) in the filename of your input and output files. Otherwise these files will be overwritten each time the loop executes.

Lines 26-53
The contents of the `pw.x` input file (see below). Everything before the '`EOF`' declaration (Line 54) will be included in the input file.

Line 56
This runs the pw.x executable using the input file you created in Line 25 and an output file whose filename contains the variable `NAME` as well as the variable CUTOFF. You will probably have to change this line; see notes for Line 25.

Line 58
Deletes the temporary files created by the code. No need to modify this.

Comments on parameters in the pw.x input file (Lines 26-53):

*Lines 26-53 make up the actual input file for the `pw.x` executable. Each variable in the input file is described individually below. The most important parameters, including the ones you may have to change for this tutorial, are marked with an asterisk *.*

Parameters in the `CONTROL` namelist (Lines 26-32)

Line 27 **: `calculation`**
The type of calculation to run: '`scf`' refers to a single self-consistent calculation of the total energy of a system. There are numerous other possibilities (consult the code's documentation for a complete list).

Line 28 **: `prefix`**
This is the prefix to prepend to the temporary files created by the code. There should be no need to modify this.

Line 29 **: `tprnfor`**
If set to `.true.`, this tells the code to output the Hellman-Feynman forces on the ions. Since the forces are not needed for this exercise, we set this to `.false.`

Line 30 **: `pseudo_dir`**
Specifies the location of the pseudopotential files. For this tutorial, these will be in the `$RUNDIR` directory (see Line 19).

Line 31 **: `out_dir`**
Specifies the location for the temporary files generated by the code. For this tutorial, the `out_dir` is set to `/tmp/`. (NOTE: Since these files can be quite large, this should always be set to a *locally mounted* filespace rather than one that is network mounted.)

Parameters in the `SYSTEM` namelist (Lines 33-40):

***Line 34* : `ibrav`**

This line contains information about the Bravais lattice of the simulation cell. `ibrav = 2` refers to a face-centered cubic lattice, which we are using in this tutorial. A full listing of numbers corresponding to the other Bravais lattice types can be found in the documentation for the Quantum-ESPRESSO distribution.

****Line 35* : `celldm(1)`**

This line contains information about the lattice parameter(s) of the system. Since in our example, the Bravais lattice is cubic, there is only one independent parameter—the cubic lattice parameter ***a***. NOTE: `celldm(1)` is ***a*** in bohr (not angstrom !). In order to determine the equilibrium lattice constant, you will need to run a series of calculations in which you change this parameter.

***Line 36* : `nat`**

The total number of atoms in the primitive cell (diamond lattice = 2).

***Line 37* : `ntyp`**

The number of distinct atomic species in the cell (Si = 1).

****Line 38* : `ecutwfc`**

The energy cutoff (in Ry) for the plane-wave basis. Here we are using the value of the loop variable `CUTOFF` to specify this. Larger values give increased accuracy, but calculations will take longer. Ordinarily, one should obtain an optimal value that balances these factors by performing a convergence test. Keep in mind that the value you should look at is the *energy differences* or the *forces*, not the *absolute* value of the energy (the energy will not converge unless you use very high cutoffs ! (NOTE: Here we are using a norm-conserving pseudopotential, but if you are using ultrasoft pseudopotentials instead, you must also perform an independent convergence of your system with respect to the charge-density energy cutoff (controlled by the optional variable `ecutrho`)).

***Line 39* : `nspin`**

If `nspin = 2`, the code will perform a spin-polarized (LSDA) calculation. This is important if you are dealing with a material with a magnetic moment, for instance. In spin-polarized calculations, each band energy is computed once for spin-up electrons and once for spin-down electrons, meaning calculations take twice as long. For this example, we have set `nspin = 1`, indicating that the Si crystal is nonmagnetic.

Parameters in the `ELECTRONS` namelist (Lines 41-43):

***Line 42* : `mixing_beta`**

By default, the code performs linear mixing of the charge density, which means the initial charge density in successive iterations in an scf cycle is projected based on past values. mixing_beta controls how much of this history is retained: higher

values mean less history is retained, lower values mean more history is retained. Adjusting this parameter can be an important factor for influencing the speed and quality of convergence in a calculation. Don't worry too much about this parameter for now; you should just leave it at 0.7 for this exercise.

`ATOMIC_SPECIES` Section (Lines 45-46):

One line must be included here for each atomic species (here we have only one). The first entry on each line is the element's chemical symbol, the second is the atomic mass (in amu), and the third is the name of the file containing the pseudopotential (the file's location is controlled by the `pseudo_dir` parameter in the `CONTROL` namelist).

`ATOMIC_POSITIONS` Section (Lines 48-50):

Each line contains the chemical symbol of one atom and the coordinates of that atom. Units are controlled by what is in parentheses following the header `ATOMIC_POSITIONS`. `'alat'` means you are using Cartesian coordinates, except everything is scaled by `celldm(1)`. These are the units which are most convenient for this exercise.

Other options are '`bohr`' (use absolute Cartesian coordinates in units of bohrs), '`angstrom`' (use absolute Cartesian coordinates in units of Angstroms), or '`crystal`' (the coordinates are in units of the lattice vectors of the primitive cell). Note that for a cubic cell, using `crystal` and using `alat` result in the same coordinates.

`*K_POINTS` Section (Lines 52-53):

This section controls which the ***k***-point sampling to be used in approximating the integral over the Brilluion Zone as a finite mesh. The term `automatic` in the header (Line 62) means an automatic Monkhorst-Pack grid of ***k***-points will be generated.

Line 53 consists of two sets of three numbers. The first three give the size of the ***k***-point grid along the x, y, and z directions. In the example script, an even `3 x 3 x 3` grid is set up. These numbers should be changed as necessary to ensure accurate sampling. The second set of three numbers controls the *offset* of the k-point mesh, or where the automatically generated mesh is centered in the reciprocal lattice. For this tutorial, keep the offset at `1 1 1` in order to ensure maximally efficient sampling of the Brilluion Zone.

As with `ecutwfc`, optimal values are a compromise between speed and accuracy; these should be ordinarily be determined by performing a convergence

test. Keep in mind that the value you should look at is the *energy difference* or *force*, not the *absolute* value of the energy (the energy will not converge unless you use very dense sampling)!

Our system is cubic, meaning it is the same size in all three Cartesian directions, so the sampling in each of the three directions should be equal.

Understanding the output file

Scroll through the output file of one of your calculations using `more` or your favorite text editor. 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 during the scf cycle, before the calculation is converged. Near the end of an scf cycle, there will be something like:

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

Note that 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 or just scroll down a lot. This is your total scf energy, as calculated by PWSCF.

Alternatively, you can type from the command line

```
$ grep '!' <filename here>
```

FAQ for Tutorial

What is a good threshold to use for convergence of energy and forces?

An acceptable error on energy differences is generally ~5 meV/atom, or else < 1% of the magnitude of the energy difference at convergence. Of course, there are some applications for which better convergence is required.

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 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 $E_{\text{true}} - E_{\text{current}} = X$ (here E_{current} is the energy for the current calculation). But how do you know E_{true}? A safe method is to take your energy at the highest cutoff (or ***k***-grid, or whatever) that you calculated – as long as it seems well converged, call that E_{true}.
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.

Does Quantum-Espresso use LDA or GGA pseudopotentials?

Quantum-Espresso has support for both LDA and GGA exchange-correlation functionals. In this workshop, the Si pseudopotential is a norm-conserving GGA potential (PW91).

I'm not sure how to interpret my energies. What is the 'correct' scale I should be looking for?

Remember that absolute energies do not usually mean very much because they depend on the reference state, as well as details such as the pseudopotential configuration. Instead, we are mostly interested in *energy differences*.

Why do I take symmetric *k*-point grids? Can I take asymmetric *k*-point grids?

For this workshop, we take a symmetric ***k***-point grid because the conventional unit cell is cubic, and the lattice parameter is the same in all three crystal directions. 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 and

c=2a, we might take a 8x8x4 mesh instead. Think about why this is (Note that if a lattice vector is longer in *real* space, it is shorter in ***k***-space).