Module 2: Quantum Espresso Walkthrough

Energy and Geometry Optimization of the H_2 Molecule

We will be using the `PWSCF` code for quantum mechanical calculations of extended systems. The `PWSCF` program is part of the `Quantum Espresso` package. It 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. PWSCF can use *norm-conserving pseudopotentials* (PP), *ultrasoft pseudopotentials* (US-PP) and *PAW potentials* within density functional theory (DFT). The PWSCF code and the Quantum-Espresso package are freely available under the conditions of the GNU GPL. Further information (including online manual) can be found at the Quantum-Espresso website quantum-espresso.org.

EWS Pre-installation QE 7.3.1

A full installation of QE 7.3.1 is available on EWS Linux in the `/class/mse404ela/QuantumEspresso` directory. To access is, add the following line at the end of your `~/.bashrc` file:

```
export PATH=$PATH:/class/mse404ela/QuantumEspresso/qe-7.3.1/bin
```

which will add the executable directory for QE to your `$PATH`.

You may also want to use the class directory for reference. This is also automounted; you will need to

```
cd /class/mse404ela
```

in order for it to be mounted, and then you can see the subdirectories inside. For example, documentation is available at `/class/mse404ela/QuantumEspresso/qe-7.3.1/Doc/`.

QE units

Quantum Espresso uses "atomic units" (Ryd for energy, Bohr for distance). You can convert these to units you will find more convenient:

- 1 bohr = 1 a.u. (atomic unit) = 0.529177211 Å
- 1 Rydberg (Ryd) = 13.60569301 eV
- 1 kcal/mol = 43.36 meV
- 1 kJ/mol = 10.36 meV
- 1 eV =$1.602176621 \times 10^{-19}$ J
- 1 eV/Å^3 = 160.2176621 GPa
- 1 Ryd/bohr^3 = 14,710.5076 GPa

QE walkthrough: The H_2 molecule

This walkthrough tutorial describes how to calculate energies and perform geometry relaxation using the `PWSCF` code in Quantum Espresso.

First create a directory for your runs, we will then copy over the input file and scripts that we will use. Create this directory somewhere convenient in your home directory. **Do NOT create a subdirectory within the directory hierarchy holding the QE installation or in the class directory.** For example:

```
cd ~
mkdir MSE404ELA_QE
cd MSE404ELA_QE
mkdir Runs
cd Runs
mkdir H2
cd H2
```

Note: you can also do this by typing `mkdir -p ~/MSE404ELA_QE/Runs/H2` (the `-p` option tells `mkdir` to create any missing directories along the way).

1. The input files

1a. Energy calculation. We will look at the `H2.scf.inp` input file; copy this file from `/class/mse404ela/QuantumEspresso/Walkthrough` into your `Runs/H2` directory. If you are logged in remotely, you will need to use `scp` or `sftp` to perform the copy. This input file is for an H_2 molecule in a cubic simulation cell. Look at the input file, which you have just copied over to your directory, `less H2.scf.inp`, and you can scroll through the file by typing space (forward), b (backwards); hit q to quit. The file will look something like this:

```
&CONTROL
calculation = 'scf' ,
restart_mode = 'from_scratch' ,
prefix = 'H2' ,
tstress = .true. ,
tprnfor = .true. ,
pseudo_dir = '/class/mse404ela/QuantumEspresso/pseudo/' ,
outdir = '/tmp/$USER' ,
/
&SYSTEM
ibrav = 1 ,
celldm(1) = 10.0 ,
nat = 2 ,
ntyp = 1 ,
ecutwfc = 20 ,
/
&ELECTRONS
conv_thr = 1.0d-8 ,
mixing_mode = 'plain' ,
diagonalization = 'david' ,
/
```

```
ATOMIC_SPECIES
  H  1.00794  H_US.van
ATOMIC_POSITIONS bohr
  H  -0.7  0.0  0.0
  H   0.7  0.0  0.0
K_POINTS automatic
  1 1 1  0 0 0
```

There's a lot going on in this file; it will tell `PWSCF` everything it needs to run our calculation:

- `&CONTROL` is the "control" block, describing how the calculation starts, what will be done in the calculation, and where output will go.
- `calculation = 'scf'` specifies a "self-consistent field" calculation (DFT). Remember: in DFT, the potential that the electrons see is determined by the charge density, which is determined by their wavefunctions. This needs to be *self-consistent*.
- `restart_mode = 'from_scratch'` declares that we will be generating a new structure, and not reading in charge densities or wavefunctions from a file (unlike a continuation run).
- `prefix='H2'` specifies the filename prefix to be used for temporary files.
- `tstress = .true.` turns on the calculation of the stress tensor.
- `tprnfor = .true.` turns on the calculation of forces.
- `pseudo_dir = '/class/mse404ela/QuantumEspresso/pseudo'` specifies the location of the directory where you store the pseudo-potentials. If you are using a local install of QE will need to edit this line to reflect the correct path to your local pseudo directory. You can alternatively remove this line if you set the environment variable `ESPRESSO_PSUEDO` using `export ESPRESSO_PSEUDO=/class/mse404ela/QuantumEspre` the script `environment_variables` in `/class/mse404ela/QuantumEspresso` is a convenient way to do this.
- `outdir='/tmp/$USER'` defines the location of the temporary files. This should always be a local scratch disk so that large I/O operations do not occur across the network. *You will need to edit this line changing `$USER` to **your user name**, to ensure that you don't accidentally try to overwrite your colleagues' directories.*
- `/` denotes the end of a block.
- `&SYSTEM` is the "system" block that describes the geometry of the calculation.
- `ibrav` specifies the crystal system. `ibrav=1` is a simple cubic structure. 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 (txt or html) in the `/class/mse404ela/QuantumEspresso/QE6.0/Doc` directory.
- `celldm` specifies the dimensions of the cell. You will be changing this parameter. `celldm` is in atomic units, or **bohrs**. Remember that 1 bohr = 0.529177211 Å. The value will depend on the Bravais lattice of the structure. For simple cubic, `celldm(1)` = a_0. In cubic systems, $a = b = c = a_0$. Consult `INPUT_PW` for other crystal systems.
- `nat` specifies the number of atoms (each individual unique atom).
- `ntyp` specifies the number of *types* of atoms (distinct chemistry).
- `ecutwfc` is the Energy cutoff for in **Rydbergs**. This one is important; you will be changing this parameter to do a convergence study.

- `&ELECTRONS` is the "electrons" block that controls how the calculation is performed. We use iterative diagonalization, and *both* the charge density and wavefunctions are improved towards the "true" solution.
- `conv_thr` is the convergence threshold. This means that self-consistency is achieved when the energy changes by less than 10^{-8} Ryd in each cycle.
- `diagonalization` is the method for diagonalizing the Kohn-Sham Hamiltonian. Davidson is fine for now.
- `mixing_mode` is the mixing method. This dictates how the charge density is changed (mixed) from step to step towards self-consistency.
- After the keyword `ATOMIC_SPECIES`, for each `ntyp` there is a line for
  - `"atomic-symbol" "atomic-weight" "pseudo-potential"`
  - The `pseudo-potential` is the name of a file in `pseudo_dir`
- After the keyword `ATOMIC_POSITIONS units` for each `nat` there is a line for
  - `atomic-symbol x y z`
  - where `x`, `y`, `z` are given in the units specified by `units`
  - `units` can be `alat`, `bohr`, `crystal`, `angstrom`.
- After the keyword `K_POINTS`, `automatic` tells PWSCF to *automatically generate* a k-point grid. In the case of automatic generation, 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.
  - You can read the documentation for the input file in `INPUT_PW` file (txt or html) in the `/class/mse404ela/QuantumEspresso/QE6.0/Doc` directory.

1b. Geometry optimization. To relax the atomic positions of the hydrogen atoms using the forces, we use the input file `H2.relax.inp`; copy this file from `/class/mse404ela/QuantumEspresso/Walkt` into your `Runs/H2` directory.

The only difference between this input file and the previous one is the `calculation` line and the added section `&IONS` in which we use the default values for all parameters for now.

2. Running `PWSCF`

To run the PWSCF code and calculate the energy of the H_2 molecule, you will need to run the `pw.x` program. If it's in your `$PATH` already, then type

```
pw.x < H2.scf.inp > H2.scf.out
```

If `pw.x` is not in your path, you'll need to explicitly use the full path to your `pw.x` file. To relax the bondlength of the H_2 dimer, use the second input file

```
pw.x < H2.relax.inp > H2.relax.out
```

2a. Energy calculation Next, look at your output file `H2.scf.out` (use `less` to do this). As you scroll through the file, you will see a lot of information. 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 fully self-consistent (the energies are changing by more than `conv_thr`). Near the end, there will be something like:

```
! total energy = -2.26866877  ryd
```

This is your total energy, as calculated by `PWSCF`. Your number may be slightly different. 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 or grep, or just scroll down a lot. For example:

```
grep '! *total energy' H2.scf.out
```

To make sure you include the `*`; it tells grep to allow as many spaces as needed between the `!` and `total energy`. The single quotes are also required, as both `!` and `*` are special characters in unix.

Scrolling down further, we come to the computed forces experienced by the ions given the DFT calculated electronic structure computed around their (fixed) positions:

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

atom    1 type  1   force =    -0.04115054    0.00000000    0.00000000
atom    2 type  1   force =     0.04115054    0.00000000    0.00000000

Total force =     0.058196     Total SCF correction =     0.000002
```

In the energy calculation, the atoms are not allowed to move and these are the forces they experience. In the geometry optimization calculation, these forces will be used in an optimization protocol to adjust the ion position to minimize the total system energy.

At the end of the file, it will tell you how long your program took to run in terms of CPU time (total time of all processors) and "wallclock" (which is the time that you experience). The last line is something like

```
PWSCF        :     0.32s CPU         0.40s WALL
```

and all of the preceding lines are breakdowns of time for each routine

```
init_run     :      0.05s CPU      0.05s WALL (       1 calls)
electrons    :      0.19s CPU      0.22s WALL (       1 calls)
forces       :      0.01s CPU      0.01s WALL (       1 calls)
stress       :      0.02s CPU      0.02s WALL (       1 calls)

Called by init_run:
wfcinit      :      0.00s CPU      0.00s WALL (       1 calls)
potinit      :      0.03s CPU      0.03s WALL (       1 calls)

Called by electrons:
c_bands      :      0.02s CPU      0.02s WALL (       6 calls)
sum_band     :      0.02s CPU      0.02s WALL (       6 calls)
```

```
v_of_rho      :      0.14s CPU      0.15s WALL (       7 calls)
newd          :      0.01s CPU      0.01s WALL (       7 calls)
mix_rho       :      0.01s CPU      0.01s WALL (       6 calls)

...
```

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

2b. Geometry optimization. If you look at `H2.relax.out`, you will see additional information. When relaxing the structure, there will also be lines which say:

```
     End of BFGS Geometry Optimization
     Final energy   =      -2.3118661236 Ry

Begin final coordinates
ATOMIC_POSITIONS (bohr)
H       -0.725970736   0.000000000   0.000000000
H        0.725970736   0.000000000   0.000000000
End final coordinates
```

And you will see that the forces on the nuclei are decreasing towards zero as their positions are optimized. The last lines show the relaxed atom coordinates.

3. Convergence issues in first-principles calculations.

The `PWSCF` code expands the one-electron wave functions in basis functions that are plane waves. They are called "plane waves" because surfaces of constant phase are parallel planes perpendicular to the direction of propagation. The plane waves are chosen to have a periodicity compatible with the periodic boundary conditions of the simulation cell, i.e. the set of $\mathbf{G}$ vectors are integer multiples of the three primitive lattice vectors. In actual calculations, we use plane waves up to a cutoff value to make the plane wave expansion finite. The cutoff is 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. To convert $\mathbf{G}$ to energy, we construct the momentum $\hbar\mathbf{G}$, and then $E = (\hbar G)^2/2m_{\rm e}$ where $m_{\rm e}$ is the mass of the electron. If G is given in inverse bohr, then G^2 corresponds to the kinetic energy in Ryd.

We will determine the convergence of the total energy of the H_2 molecule with respect to the energy cutoff of the plane wave basis set. Open the file `H2.scf.inp` using `vi` or `emacs` and look for the parameter `ecutwfc`. Double this parameter from 20 to 40 Ryd and assess how the total energy of the H_2 molecule changes when we increase the energy cutoff of the plane wave basis.

We would like to identify the required cutoff energy for a convergence of the energy to within 2 mRy/atom (i.e. 0.004 Ry for our 2 atom system), and determine the accuracy of the bond length at this cutoff energy. To speed up the convergence calculations, we will use a script. Following is the script we will be using to check the convergence with respect to the plane wave cutoff. Copy the file `Run_Ecut.bash` into `Runs/H2`.

```
#!/bin/bash

INPUTFILE=H2.scf.inp
PWSCF=pw.x
for ecut in 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
do
  sed "/ecutwfc/s/.*/ ecutwfc=$ecut/" $INPUTFILE | $PWSCF > out
  e=$(grep '! *total energy' out | tail -n 1 | awk '{print $5}')
  echo $ecut $e
done
exit
```

The script loops over the values ecut from 10 to 80 Ry in steps of 5 Ry. (*Note:* you can use the bash utility `seq 10 5 80` to do this too; try it out!) It creates a new input file based on `H2.scf.inp` by changing the entry in the line `ecutwfc` using the bash utility `sed` (kind of like `awk`) and then *pipes this as input* to the `PWSCF` code. Finally, it extracts the energy from the file `out` using `grep` and prints them to the terminal with `echo`. *Note:* It rewrites the file `out` each time.

Run the script to determine the convergence of the total energy of the H_2 molecule with respect to the energy cutoff.

```
chmod 755 Run_Ecut.bash
./Run_Ecut.bash
```

Use `MATLAB` (or other program of your preference) to fit your data of total energy vs. cutoff energy to a decaying exponential to extrapolate out to the energy at an infinite cutoff. Consider fitting a function of the form $y = A\exp(-Bx) + C$. Since this is cannot be formulated as linear regression problem, you can use the `MATLAB` curve fitting toolbox GUI by typing `cftool` into the command prompt.

Load the y data (`Etotal`) and x data (`Ecut`) from your workspace into the tool, and specify a custom function of the form $y = a * \exp(-b * x) + c$.

If you would prefer to do your curve fitting with python, you can see an example of a jupyter notebook that does just that in `/class/mse404ela/QuantumEspresso/curve-fit.ipynb`. You should copy that example notebook into the directory where you do you analysis, and then you can load the virtual environment for our class:

```
. /class/mse404ela/mse404/bin/activate
```

You need only do this once to start the virtual environment. When you wish to leave it, just execute `deactivate`. To launch the jupyter notebook, do

```
jupyter-notebook
```

If you don't have a browser open, it will open one for you and you will see your current directory; launch the curve-fit notebook to see how it works.

Using the fitted values for a, b, and c, what is:

a. $\lim_{E_{\text{cut}}\to\infty} E_{\text{total}}$?

b. the minimum value of E_{cut} for the desired accuracy of 0.004 Ryd?

When you have found the cutoff that you will use for hydrogen, **rerun the relaxation calculation using `H2.relax.inp` with your converged cutoff** to make a DFT prediction for the hydrogen molecule bond length. The experimentally determined value is ~74 pm.

In this example, we have considered a single H_2 molecule in a large real space unit cell, meaning the interactions between periodic images of the molecule are effectively zero. For atoms and molecules with no periodic interactions, the Bloch Theorem does not apply (i.e. there is no translational invariance of the Hamiltonian for isolated entities). A molecule in a large box is effectively isolated, allowing us to perform a single k-point calculation (often called a "Gamma point" calculation, as Γ is the shorthand for $\mathbf{k} = (0, 0, 0)$). Including more k-points just captures the intermolecular interactions more accurately, and for large real space cells these are effectively zero. In your project, you will consider a condensed system, and so will need to look at the convergence of the energy with respect to both the cutoff energy *and* the k-point mesh. You can modify the automation script and input files used in this walkthrough for your project.

Organizing your runs

You can organize your runs any way you like, or you 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 this is really up to you. Be cognizant especially of the working directories (`outdir`) that you specify in your input files. Keep in mind also that—as we saw here—input files are just text files, and so you can write scripts that create and manipulate those input files in an automated way. This significantly improves the efficiency and reduces human error in running computational simulations.

FAQ

I do not understand quantum mechanics at all.

- General introductions to Quantum Mechanics: 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 www-keeler.ch.cam.ac.uk/lectures/quant_le
- Or, more complete: www.math.utah.edu/~gold/doc/quantum.pdf
- Very simple primer: www.chem1.com/acad/webtut/atomic

N.B. It is not essential know every detail of quantum mechanics to run 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.

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.

The weights of the k-points add up to 2, not 1. Yes. This is a "feature" of the code (because each electron has a "spin" degree of freedom, and there are two of them). Don't worry about it.

How is "convergence of energy" defined? You say that your energy is converged to X Rydbergs when $E_{\text{true}} - E_n = X$ (where E_n is the energy at the current set of parameters). How do you know E_{true}? In practice you might take your energy at the highest cutoff (or k-point grid) that you calculated—if that is converged, you might call that E_{true}. Alternatively, you might extrapolate to an infinite cutoff energy using curve fitting, as we did above. But note: in both of these choices, we have **not defined convergence** as the difference between two successive calculations in a sequence of increasing cutoff / k-point density.

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. From experience, we know that a good error on energy differences is ~5 meV/atom and on forces is ~10 meV/Angstrom. These are just values are just a general guide derived from many first-principles calculations in the past. Depending on the system and property you may have to use different convergence criteria.

I am having problems both converting Ryd to eV and Bohr to Å. These units page may help you:

- physics.nist.gov/cuu/Constants
- www.chemie.fu-berlin.de/chemistry/general/units_en.html

Does PWSCF use LDA or GGA? DFT or Hartree Fock? `PWSCF` uses DFT. It has both LDA and GGA.

Why do I take symmetric k-point grids? Can I take asymmetric k-point grids? The symmetry of the k-point grid should follow the symmetry of the crystal lattice; in a cubic material, all three directions are the same so the number of divisions along each direction will be the same. However, other cases (like hexagonal materials, or lower symmetry) will have different lengths of reciprocal lattice vectors. A good rule of thumb is to choose the divisions along each direction so that the "step size" (in 1/length) is approximately the same for the three directions. So that may suggest something like a $24 \times 24 \times 14$ mesh for a hexagonal closed packed crystal (as one example).

When I try to run QE I get a mysterious IOTK error. What's happening? The `IOTK` library is used by `QE` to write and read XML files. QE will periodically throw an IOTK runtime error that is not easily reproducible and poorly understood even by the QE developers (see FAQ). The easiest workaround is to just use a different workstation, or—if you are working remotely—relogin with a new ssh shell.