Magnetic Properties from First Principles

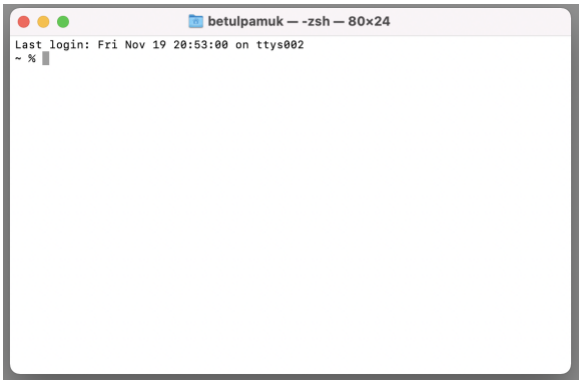
Nov 24, 2021, 9:00 AM → Nov 26, 2021, 8:20 PM Europe/Istanbul

Hands-on tutorial of first-principles calculations

Betül Pamuk
PARADIM, Cornell University

1. Installing Quantum Espresso Software Package

Search for the "Terminal" application in a computer that has a Linux (e.g. Ubuntu, Suse, …) or a Mac OS:



At this point, you can either run the rest of these exercises on your local machine or connect to a supercomputer to run these calculations. I will be connecting to a cluster at Cornell to give you an example of how this process works.

To connect to the cluster, you need to use the `ssh` command:

```
~ % ssh -Y bp385@tardis3.cac.cornell.edu
Last login: Fri Nov 19 20:22:22 2021 from 74.69.56.131
bp385@tardis3:~$
```

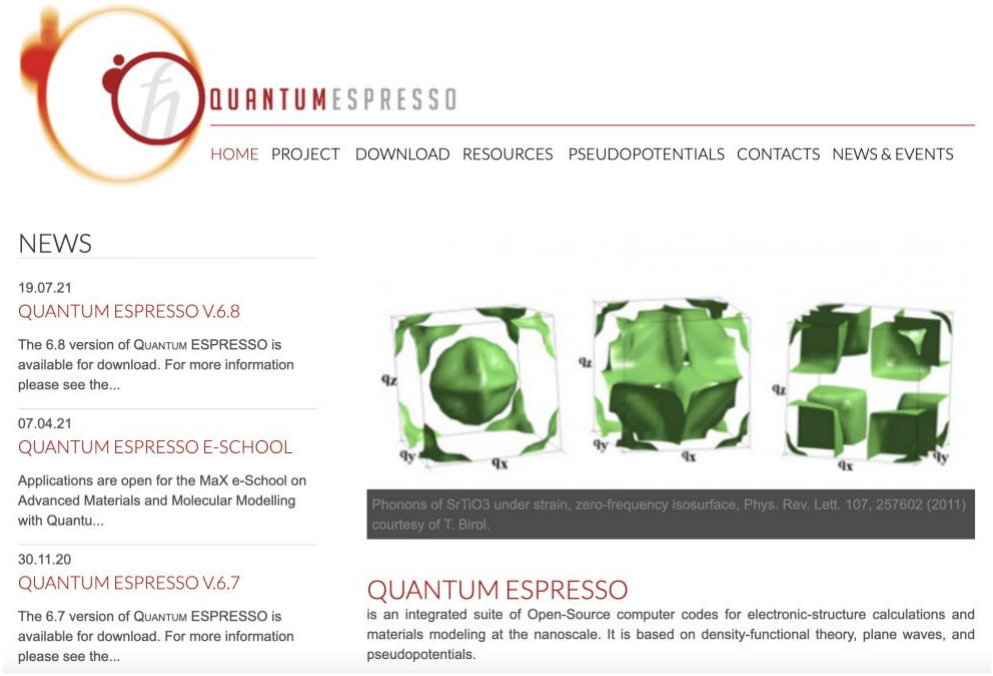
where `bp385` is my username and `tardis3.cac.cornell.edu` is the IP address of the cluster that I am connecting to.

Now we can make a new folder where we will run our calculations and go into that folder:

```
$ mkdir MPFP
$ cd MPFP/
$
```

We will be using Quantum Espresso (QE) software, which is an open-source package that runs first-principles calculations using planewave basis sets and pseudopotentials.

The website is: http://www.quantum-espresso.org/



The latest version can be found at: http://www.quantum-espresso.org/download. Let's now download the latest version of QE:

```
$wget https://github.com/QEF/q-e/archive/refs/tags/qe-6.8.tar.gz
```

and unpack the compressed file:

```
$ tar xzvf qe-6.8.tar.gz
```

Now we can go inside this directory and list the files inside:

```
$ cd q-e-qe-6.8/
$ ls -lrt
total 200
drwxr-xr-x.  6 bp385 Domain Users   141 Jul 19 05:11 XSpectra
drwxr-xr-x.  4 bp385 Domain Users  4096 Jul 19 05:11 XClib
drwxr-xr-x.  3 bp385 Domain Users  4096 Jul 19 05:11 UtilXlib
drwxr-xr-x.  2 bp385 Domain Users  4096 Jul 19 05:11 upflib
drwxr-xr-x. 86 bp385 Domain Users  4096 Jul 19 05:11 test-suite
drwxr-xr-x.  7 bp385 Domain Users   193 Jul 19 05:11 TDDFPT
-rw-r--r--.  1 bp385 Domain Users  4290 Jul 19 05:11 README.md
-rw-r--r--.  1 bp385 Domain Users  2481 Jul 19 05:11 README_GPU.md
drwxr-xr-x.  5 bp385 Domain Users   124 Jul 19 05:11 QEHeat
drwxr-xr-x.  5 bp385 Domain Users   106 Jul 19 05:11 PWCOND
drwxr-xr-x.  7 bp385 Domain Users   156 Jul 19 05:11 PW
```

```
drwxr-xr-x.  2 bp385 Domain Users  4096 Jul 19 05:11 pseudo
drwxr-xr-x.  7 bp385 Domain Users   151 Jul 19 05:11 PP
drwxr-xr-x.  7 bp385 Domain Users   136 Jul 19 05:11 PHonon
drwxr-xr-x.  6 bp385 Domain Users   123 Jul 19 05:11 NEB
drwxr-xr-x.  2 bp385 Domain Users  4096 Jul 19 05:11 Modules
-rw-r--r--.  1 bp385 Domain Users 12055 Jul 19 05:11 Makefile
drwxr-xr-x.  2 bp385 Domain Users  4096 Jul 19 05:11 LR_Modules
-rw-r--r--.  1 bp385 Domain Users 42741 Jul 19 05:11 logo.jpg
-rw-r--r--.  1 bp385 Domain Users 18009 Jul 19 05:11 License
drwxr-xr-x.  3 bp385 Domain Users  4096 Jul 19 05:11 LAXlib
drwxr-xr-x.  8 bp385 Domain Users   163 Jul 19 05:11 KS_Solvers
drwxr-xr-x.  3 bp385 Domain Users  4096 Jul 19 05:11 install
drwxr-xr-x.  2 bp385 Domain Users   110 Jul 19 05:11 include
drwxr-xr-x.  5 bp385 Domain Users   106 Jul 19 05:11 HP
drwxr-xr-x. 13 bp385 Domain Users   251 Jul 19 05:11 GWW
drwxr-xr-x.  5 bp385 Domain Users   101 Jul 19 05:11 GUI
drwxr-xr-x.  4 bp385 Domain Users  4096 Jul 19 05:11 FFTXlib
drwxr-xr-x.  8 bp385 Domain Users   232 Jul 19 05:11 external
drwxr-xr-x.  7 bp385 Domain Users   216 Jul 19 05:11 EPW
-rw-r--r--.  1 bp385 Domain Users  5279 Jul 19 05:11 environment_variables
drwxr-xr-x.  4 bp385 Domain Users  4096 Jul 19 05:11 Doc
drwxr-xr-x.  2 bp385 Domain Users   248 Jul 19 05:11 dft-d3
drwxr-xr-x.  4 bp385 Domain Users  4096 Jul 19 05:11 dev-tools
drwxr-xr-x.  5 bp385 Domain Users   106 Jul 19 05:11 CPV
drwxr-xr-x.  6 bp385 Domain Users   145 Jul 19 05:11 COUPLE
-rw-r--r--.  1 bp385 Domain Users  1108 Jul 19 05:11 CONTRIBUTING.md
-rwxr-xr-x.  1 bp385 Domain Users  2357 Jul 19 05:11 configure
-rw-r--r--.  1 bp385 Domain Users 28889 Jul 19 05:11 CMakeLists.txt
drwxr-xr-x.  3 bp385 Domain Users  4096 Jul 19 05:11 cmake
drwxr-xr-x.  2 bp385 Domain Users  4096 Jul 19 05:11 clib
drwxr-xr-x.  6 bp385 Domain Users   168 Jul 19 05:11 atomic
drwxr-xr-x.  2 bp385 Domain Users   159 Jul 19 05:11 archive
```

For today's exercises, we will need the main program pw.x and some post-processing tools. To be able to compile the code and have this executable, the code needs to know where the compilers and the related libraries are located in your system. QE can do this automatically.

If you are doing this in your local machine, some of the useful programs for this purpose are: `gcc, gfortran, openmpi, gnuplot, make, vim, tcsh,` as well as `xcrysden,` and `vesta`
For example in an Ubuntu machine, you can install them by typing
`sudo apt install gcc`
and entering your root password.

If you are doing this in a cluster, you need to make sure that the modules that are relevant for fortran and mpi are loaded. For example, you can do this by typing:
`module avail`
to see the available modules and loading the relevant ones, for example:
```
module load intel
module list
```

Now we can create the file with the relevant information for the compilers and libraries.

```
$ ./configure
```

This will look for the needed compilers and libraries and create the `make.inc` file. You can enter this file and make necessary edits if needed. Today, we will continue with the default settings. Next, we can compile the executable (which takes about 5-10 minutes) by typing:

```
$ make pw
```

Now you can find the executable `pw.x` in the bin directory

```
$ ls bin/
```

To compile the post-processing (pp) tools (e.g. `bands.x`) we would need to repeat this procedure:

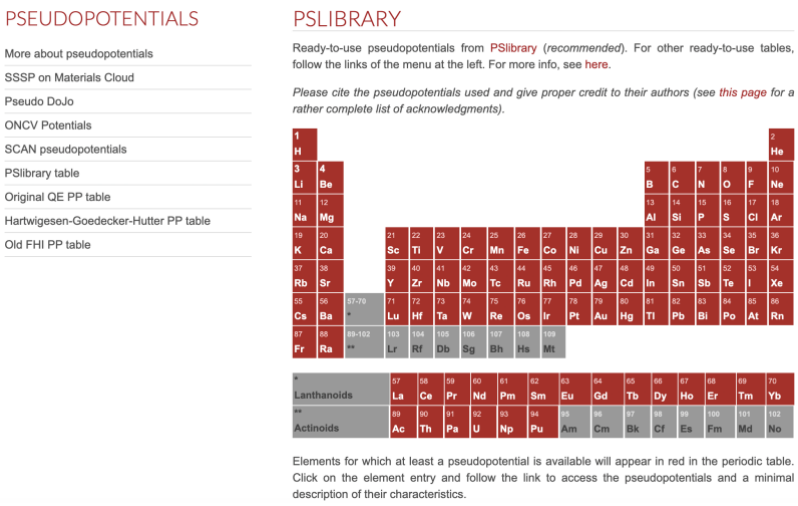
```
$ make pp
```

At this point it would be useful to create a shortcut for QE executables:

```
$ cd bin/
$ pwd
/home/fs03/bp385/MPFP/q-e-qe-6.8/bin
$ cat >> ~/.bashrc << EOF
export QE="/home/fs03/bp385/MPFP/q-e-qe-6.8/bin"
EOF
$ source ~/.bashrc
```

Finally, we can familiarize ourselves with some important aspects of the QE website.

QE is based on using pseudopotentials and it has production-quality pseudopotentials in its PSlibrary:
http://www.quantum-espresso.org/pseudopotentials



A list of all the input parameters and a brief description of them is listed under the website Resources→Documentation→Input Data Description→PWscf
http://www.quantum-espresso.org/Doc/INPUT_PW.html

Now let's go back to our main directory where we will be performing the rest of the calculations

```
$ cd ~/MPFP/
```

2. Running a test calculation

Today we will be looking at the structure of graphite and graphene. Let's first run a test calculation to make sure that our pw.x executable is running without any problems. We will first create a directory to run our test calculation:

```
$ mkdir test_run; cd test_run
```

Here, we will download the needed pseudopotential for the carbon atom in our system. For simplicity, we will use of the original pseudopotentials (but any pseudopotential would work for the purposes of this tutorial) with the LDA functional:

```
$ wget http://www.quantum-espresso.org/upf_files/C.pz-vbc.UPF
```

Next, we can create a sample input file for an scf calculation:

```
$ cat << EOF > graphite.scf.in
&control
calculation = "scf",
prefix = "graphite",
pseudo_dir = "./",
outdir = "./"
/
&system
ibrav = 4,
celldm(1) = 4.60925103211943136
celldm(3) = 2.67485154304358451840
nat = 4,
ntyp = 1,
ecutwfc = 100.0,
/
&electrons
```

```
/
ATOMIC_SPECIES
C 12.011 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.00 0.00 0.25
C 0.00 0.00 0.75
C 0.333333 0.666666 0.25
C 0.666666 0.333333 0.75
K_POINTS automatic
6 6 2 1 1 1
EOF
```

Now we are ready to run our test job. You can run this in series on your local computer using the shortcut we have created for the executables:

```
$ $QE/pw.x < graphite.scf.in > graphite.scf.out
```

or in a cluster, you can run this in parallel using MPI, for example using 4 processors:

```
$ mpirun -n 4 $QE/pw.x < graphite.scf.in > graphite.scf.out
```

The output should look like:

```
     Program PWSCF v.6.8 starts on 20Nov2021 at 15:25:51

     This program is part of the open-source Quantum ESPRESSO suite
     for quantum simulation of materials; please cite
         "P. Giannozzi et al., J. Phys.:Condens. Matter 21 395502 (2009);
         "P. Giannozzi et al., J. Phys.:Condens. Matter 29 465901 (2017);
         "P. Giannozzi et al., J. Chem. Phys. 152 154105 (2020);
          URL http://www.quantum-espresso.org",
     in publications or presentations arising from this work. More details at
     http://www.quantum-espresso.org/quote

     Parallel version (MPI), running on     4 processors

     MPI processes distributed on     1 nodes
     R & G space division:  proc/nbgrp/npool/nimage =       4
     16570 MiB available memory on the printing compute node when the environment starts

     Waiting for input...
     Reading input from standard input
…
     Parallel routines

     PWSCF        :      0.98s CPU      1.13s WALL


   This run was terminated on:  15:25:52  20Nov2021

=------------------------------------------------------------------------------=
   JOB DONE.
=------------------------------------------------------------------------------=
```

Once we successfully run this job, we are ready to do more serious calculations.

3. Convergence of parameters

QE is a DFT software that is based on pseudopotentials and planewaves. In the planewave-based DFT codes, an important convergence parameter is the "**planewave kinetic energy cutoff**" with the `ecutwfc` flag in the input file.

Kohn-Sham wavefunction in a planewave basis is:
$\phi_i(\boldsymbol{r}) = \sum_{\boldsymbol{G}} c_i(\boldsymbol{G}) e^{i\boldsymbol{G}.\boldsymbol{r}}$,
where **G** is the reciprocal lattice vector.
Then Kohn-Sham equation becomes:
$\frac{|\boldsymbol{G}|^2}{2}\boldsymbol{c_i}(\boldsymbol{G}) + \sum_{\boldsymbol{G}'} \boldsymbol{V_{tot}}(\boldsymbol{G}-\boldsymbol{G}')\boldsymbol{c_i}(\boldsymbol{G}') = \boldsymbol{\epsilon_i}\boldsymbol{c_i}(\boldsymbol{G})$.
Since this expansion with the sum cannot run to infinity, we need to approximate it to include enough planewaves to keep the accuracy of the calculations. This is done via the kinetic energy of the planewaves:
$E_{cut} = \frac{\hbar^2 |G_{max}|^2}{2m_e}$.

For each observable that you want to calculate, you need to check that this observable does not change as you change the planewave kinetic energy cutoff of the DFT calculation.

Let us now check how the total energy changes as we change the `ecutwfc` flag in the input file.

We start by creating a clean folder for this purpose:

```
$ mkdir ~/MPFP/ecut_convergence; cd ~/MPFP/ecut_convergence
```

and we copy the pseudopotential and the input file from the test_run folder of the previous step

```
$ cp ../test_run/C.pz-vbc.UPF .
$ cp ../test_run/graphite.scf.in graphite-5.in
```

We enter the file `graphite-5.in` using the program `vim` and edit the file

```
$ vi graphite-5.in
```

Once inside the file, you can:

- hit `i` to enter the insert mode
- go with your cursor to the line with the `ecutwfc` flag and change it to `ecutwfc = 5.0,`
- hit `esc` to enter the command mode
- hit `:wq` to write and quit vim.

Now we can run this calculation with this planewave kinetic energy cutoff of 5 Ry:

```
$ mpirun -n 4 $QE/pw.x < graphite-5.in > graphite-5.out
```
The total energy of the calculation can be found towards the end of the output file `graphite-5.out` with the line that has "`! total energy`":

We can quickly get this information written to the terminal by using the command:

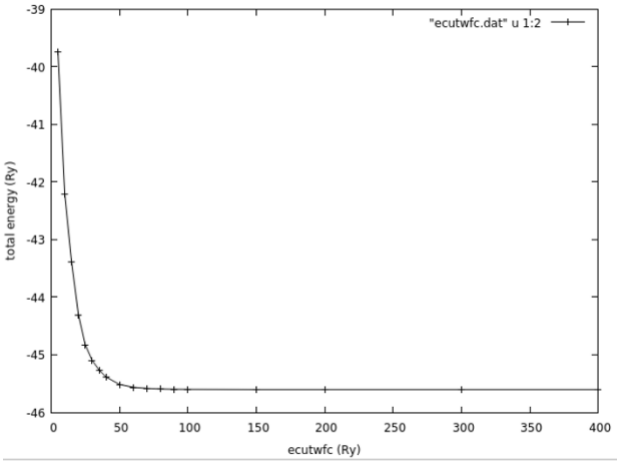
```
$ grep ! graphite-5.out
!    total energy              =     -39.74277194 Ry
```

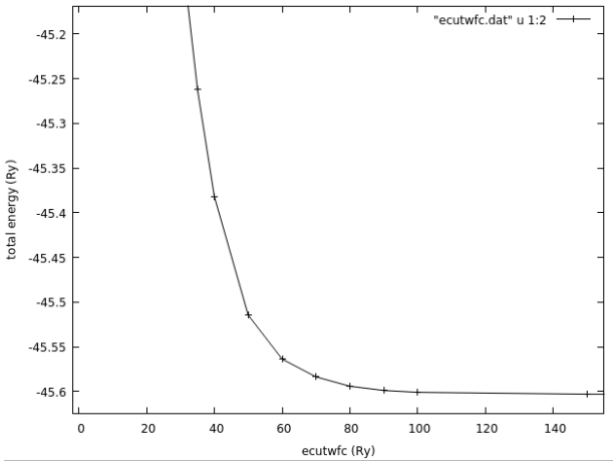
Now you can repeat this exercise for
`ecutwfc = 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0, 150.0, 200.0, 300.0, 400.0`
You can use `grep` to get the relevant information of the total energy and write it into a file

```
$ grep ! graphite-*.out > ecutwfc.dat
```

After editing this file to have two columns of ecutwfc and total energy, you can use `gnuplot` (or any other plotting program that you are familiar with) for plotting how the total energy changes as we change the planewave kinetic energy cutoff.

```
$ gnuplot
gnuplot> set xlabel "ecutwfc (Ry)"
gnuplot> set ylabel "total energy (Ry)"
gnuplot> plot "ecutwfc.dat" u 1:2 w lp lc black
gnuplot> quit
```





If you look carefully at these plots, you will see that the total energy difference between the converged value at `ecutwfc=400.0` Ry and the value of `ecutwfc=100.0` Ry is about 15 meV per C atom and this is a good enough convergence for the purpose of this tutorial.

Note that the absolute value of the total energy of DFT calculations is not meaningful, because we perform these calculations for an infinitely periodic crystal and this value would depend on the choice of the pseudopotential and conventions used in each DFT code. This is different than

solving Schrödinger equation for a hydrogen atom, because there the total energy is calculated with respect to the vacuum, which is not the case for solving DFT for an infinite crystal system. Therefore, what is meaningful is the **difference of the total energies**, as the artificial components of the energies cancel out.

I would like to reiterate that it is very important to check the convergence of the observable (bandstructure, lattice parameters, equilibrium structure, phonons, …) you are aiming to calculate with DFT should be converged with respect to the planewave kinetic energy cutoff.

Homework:
Another important parameter that you need to check is the convergence of the electronic momentum *k*-point mesh. This mesh creates a grid to sample the Brillouin zone (BZ) in reciprocal space. Again, the observable you are aiming to calculate with DFT should be converged with respect to the *k*-point mesh.
The sampling of the BZ can include the Γ-point by setting the input file

```
K_POINTS automatic
6 6 2 0 0 0
```

or can be shifted by half a grid spacing in each direction (Monkhorst-Pack grid) by

```
K_POINTS automatic
6 6 2 1 1 1
```

Depending on the material system, it might be important to include the Γ-point in the calculations, but typically the shifted k-points converges faster in terms of the number of inequivalent k-points than the Γ-centered k-point sampling of the BZ.

Due to timing constraints, I suggest you check, as homework, how the total energy converges as the number of inequivalent k-points changes for each sampling method.

4. Optimization of lattice parameters and atomic coordinates

The equilibrium structure obtained by minimizing the total energy of DFT calculations is the crystal structure at zero pressure and zero temperature.

The equilibrium structure can be obtained by artificially moving the atoms (which are not constrained by symmetry) and changing the lattice parameters and finding the structure with the minimum total energy. QE has internal algorithms to relax the atomic coordinates (relax) and lattice parameters (vc-relax) to optimize the equilibrium structure and we will learn what the key input parameters for this purpose are.

Let us find the equilibrium structure of graphite with Bernal (AB) stacking. Graphite has hexagonal lattice with lattice vectors:

$a = a(\ 1 \quad 0 \quad 0)$
$b = a(-1/2 \ \sqrt{3}/2 \quad 0)$
$c = c(\ 0 \quad 0 \quad 1)$

with a =2.464 Å and c/a = 2.724

Hexagonal Bravais lattice can be selected by setting in the input file:
`ibrav=4`
and this setting requires two parameters to define the lattice vectors:
`celldm(1)` = a in default units of Bohr radius
`celldm(3)` = c/a

Now you can run a series of calculations with different input files with a mesh that covers the two-dimensional (2D) space of `celldm(1)` and `celldm(3)` and find the values that has the minimum total energy. This will be the equilibrium lattice parameters of Bernal-stacked graphene. Today we will rely on the internal algorithms of QE.

In the case of graphene all the atomic coordinates are fixed by symmetry, therefore they will not change as we are performing the relaxation calculation. The atomic positions in crystal coordinates, i.e., in terms of fraction of the lattice vectors are:

```
ATOMIC_POSITIONS crystal
C 0.00 0.00 0.25
C 0.00 0.00 0.75
C 0.333333 0.666666 0.25
C 0.666666 0.333333 0.75
```

Note that if there are any atomic coordinates that are not fixed by symmetry of the material of interest, these coordinates can be relaxed without changing the lattice parameters by setting the input parameter to
`calculation = "relax"`
This is not relevant in this example as all the coordinates are fixed by symmetry here.

We are now ready to create a new directory to start preparing our calculation:

```
$ mkdir ~/MPFP/relax_graphite; cd ~/MPFP/relax_graphite
$ cp ../test_run/C.pz-vbc.UPF .
```

Let us assume that we have no prior information about the lattice parameters of our material and arbitrarily set the lattice parameters:

```
$ cat << EOF > graphite.relax1.in
&control
calculation = "vc-relax",
prefix = "graphite",
pseudo_dir = "./",
```

```
outdir = "./"
/
&system
ibrav = 4,
celldm(1) = 4.0,
celldm(3) = 2.0,
nat = 4,
ntyp = 1,
ecutwfc = 100.0,
/
&electrons
/
&ions
/
&cell
/
ATOMIC_SPECIES
C 12.011 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.00 0.00 0.25
C 0.00 0.00 0.75
C 0.333333 0.666666 0.25
C 0.666666 0.333333 0.75
K_POINTS automatic
6 6 2 1 1 1
EOF
```

The full optimization of the structure is performed by changing the input parameter to `calculation = "vc-relax",` and adding two cards `&ions` and `&cell` that has further controls for more detailed calculations, but the default settings are enough for the purposes of this tutorial.

After running this calculation, we can investigate the output file

```
$ mpirun -n 4 $QE/pw.x < graphite.relax1.in > graphite.relax1.out
```

At each step of the vc-relax (for variable cell relaxation) calculation, a self-consistent field (scf) calculation is performed for a new set of lattice parameters and/or atomic coordinates. Then the stress tensor is evaluated, and lattice parameters are updated to reduce the stress.

We can look inside the output file `graphite.relax1.out` and search for the keyword stress by directly typing

```
/stress
```

and by hitting `n`, you can go to the next instance of this keyword to check the evolution of the stress tensor as the relaxation continues. We see that the total pressure on the system is quite large at first iteration (P = 4044.50 kbar).

Also note that the `/Forces` acting on the atoms are zero by symmetry.

Now you can search for `/final` to get the optimized structure at the end of this calculation:

```
     bfgs converged in   8 scf cycles and   7 bfgs steps
     (criteria: energy <  1.0E-04 Ry, force <  1.0E-03Ry/Bohr, cell <  5.0E-
01kbar)

     End of BFGS Geometry Optimization

     Final enthalpy           =     -45.5685620269 Ry

     File ./graphite.bfgs deleted, as requested
Begin final coordinates
     new unit-cell volume =    200.59959 a.u.^3 (    29.72579 Ang^3 )
     density =      2.68383 g/cm^3

CELL_PARAMETERS (alat=  4.00000000)
   1.149455645   0.000000000   0.000000000
  -0.574727822   0.995457789  -0.000000000
  -0.000000000  -0.000000000   2.739270972

ATOMIC_POSITIONS (crystal)
C             -0.0000000000       -0.0000000000        0.2500000000
C              0.0000000000        0.0000000000        0.7500000000
C              0.3333330000        0.6666660000        0.2500000000
C              0.6666660000        0.3333330000        0.7500000000
End final coordinates
```

Note that the final lattice parameters are given in units of celldm(1) parameter of the input file. Therefore the new lattice parameters are:

$a =$ 1.149455645 x 4.0 = 4.597822580 Bohr
$c/a =$ 2.739270972/1.149455645 = 2.383102805

Note that if one of the lattice parameters changes drastically, it also effectively changes the number of planewaves we have in the calculation. This can cause the loss of accuracy in our calculations and the final parameters of this calculation are not necessarily the optimized structure of our material. To make sure that the optimized structure at the end of this run is the real equilibrium structure, we need to create a new input file with the updated lattice parameters and re-run this calculation until it says

```
     bfgs converged in   1 scf cycles and   0 bfgs steps
```

In this case, I needed to submit it for a total of 4 times until I get the fully relaxed equilibrium structure with the lattice parameters

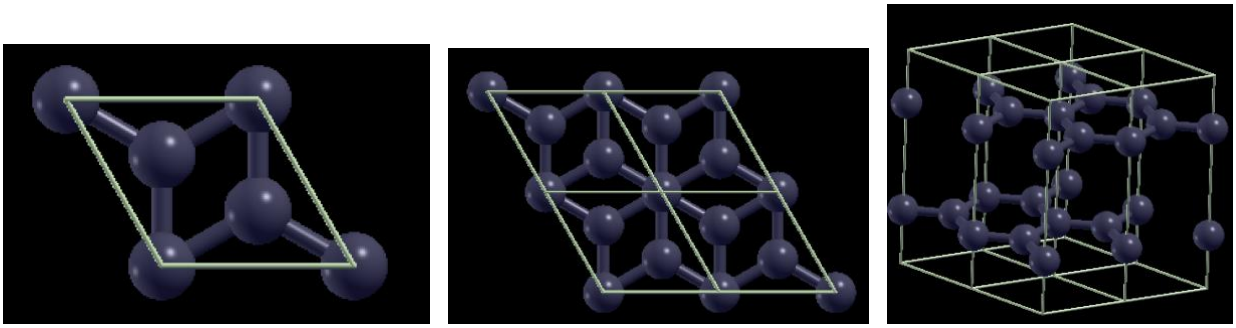
```
celldm(1) = 4.60925103 Bohr
celldm(3) = 2.674851543
```

This gives the lattice parameters of a = 2.439 Å, c = 6.524 Å, which are in good agreement with the experimental values of a = 2.464 Å, c = 6.712 Å. This is an artifact of the LDA approximation in DFT, where LDA tends to overbind the material leading to smaller lattice parameters than the experimental

values. In fact, in general van der Waals corrections are crucial for simulations of layered 2D systems with weak interlayer bonds. Therefore, the good agreement with the experiments in this example is simply due to the cancellation of errors, where overbinding of the LDA functional compensates for the lack of van der Waals interactions. As a side note, another typical functional within the GGA approximation is the PBE functional and it tends to underbind the material leading to larger lattice parameters than the experimental values.

At this point we can visualize the equilibrium structure directly from the QE input or output files by using the software Xcrysden:

```
$ xcrysden --pwi graphite.relax4.in
```



Homework:

Please check that these lattice parameters are converged with respect to the planewave energy cutoff and k-point sampling.

5. Input file for a two-dimensional material

As the focus of this school is 2D materials, we next would like to see how one can simulate a 2D material. For this purpose, we will "peel" one of the layers of graphite and "create" a single layer graphene.

Periodic boundary conditions that are essential for the simulations of infinitely periodic crystals are now needed to be carefully considered for creation of the 2D system. To create the 2D structure, we need to artificially increase the *c* lattice parameter perpendicular to the 2D layer. This artificial *c* lattice parameter is needed to be large enough that the layer does not "see" its periodic image, i.e., the results are independent of the choice of this artificially large *c* lattice parameter.

Let us now make a clean directory to create our graphene

```
$ mkdir ~/MPFP/bands_graphene; cd ~/MPFP/bands_graphene/
$ cp ../test_run/C.pz-vbc.UPF .
```

```
$ cat << EOF > graphene.scf.in
&control
calculation = "scf",
prefix = "graphene",
```

```
pseudo_dir = "./",
outdir = "./"
verbosity = "high"
/
&system
ibrav = 4,
celldm(1) = 4.60925103211943136,
celldm(3) = 20.67485154304358451840,
nat = 2,
ntyp = 1,
ecutwfc = 100.0,
occupations = "smearing",
smearing='mp'
degauss=0.01,
assume_isolated = "2D",
/
&electrons
/
ATOMIC_SPECIES
C 12.011 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.333333 0.666666 0.0
C 0.666666 0.333333 0.0
K_POINTS automatic
6 6 2 1 1 1
EOF
```

For the purposes of this tutorial, we have assumed that the in-plane lattice parameter *a* of graphene is the same as that of graphite and set out-of-plane *c* lattice parameter to 10 times the *c* lattice parameter of graphite. We have also adjusted the number of atoms and the atomic positions of the two atoms in the unit cell. In addition, to avoid the artificial interaction with the periodic images of the material, the Coulomb interaction can be truncated in the *c*-direction by using the flag `assume_isolated = "2D"`.

4. Electronic band structure

An important input parameter that is worth discussing at this point is occupations. For an insulator, `occupations = "fixed"` fills the bands up to the valence band maximum. For a metallic system, `occupations = "smearing"` allows fractional occupations of the Kohn-Sham states with a new distribution set by the flag `smearing='mp'`, which is similar to Fermi-Dirac distribution, with the distribution width (or temperature) set by the flag `degauss` in units of Ry. We now add these keywords to obtain the Fermi level of the material from the scf calculation. Note that Fermi level should be converged with respect to the k-points mesh of the system. For a system with a Fermi point instead of a surface, to capture this behavior in our calculations, we need a very dense k-mesh. This calculation would take longer, so for the purposes of this tutorial we leave it as a homework for you to check.

```
$ mpirun -n 4 $QE/pw.x < graphene.scf.in > graphene.scf.out
$ grep Fermi graphene.scf.out
```

Once we solve the ground state Kohn-Sham equations in the scf calculation, we fix the ground state electron density, Hartree, and XC potentials. In a non-self-consistent calculation, the Kohn-Sham eigenfunctions and eigenvalues are calculation without updating the Hamiltonian. This applies for `calculation = "bands"`(, and `calculation = "nscf"`, which can be useful for density of states calculation.)

```
$ cat << EOF > graphene.bands.in
&control
calculation = "bands",
prefix = "graphene",
pseudo_dir = "./",
outdir = "./"
verbosity = "high"
/
&system
ibrav = 4,
celldm(1) = 4.60925103211943136,
celldm(3) = 20.67485154304358451840,
nat = 2,
ntyp = 1,
ecutwfc = 100.0,
occupations = "smearing",
smearing='mp'
degauss=0.01,
assume_isolated = "2D",
nbnd = 32,
/
&electrons
/
ATOMIC_SPECIES
C 12.011 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.333333 0.666666 0.0
C 0.666666 0.333333 0.0
K_POINTS crystal_b
4
0.0 0.0 0.0 50 !Gamma
0.33333333 0.3333333 0.0 50 !K
0.5 0.0 0.0 50 !M
0.0 0.0 0.0 50 !Gamma
EOF
```

```
$ mpirun -n 4 $QE/pw.x < graphene.bands.in > graphene.bands.out
```

We also increase the number of bands `nbnd = 32`, to make sure that the unoccupied bands are also included in our calculation.
Finally, we need to edit the k-points to reflect the path along which we want to obtain the band structure. In this example, we have 4 high-symmetry points (Γ, K, M, Γ) that we want to include in our path. The coordinates of these points are given in crystal coordinates, meaning they are represented in terms of fractions of reciprocal lattice vectors. Finally, we want 50 k-points in each direction that connects these high symmetry points. The output already includes all the necessary information with the k-point vector and the eigenvalues at that k-point, which are the electronic bands.

Finally, we can use the post-processing tool `bands.x` to easily plot the electronic band structure.

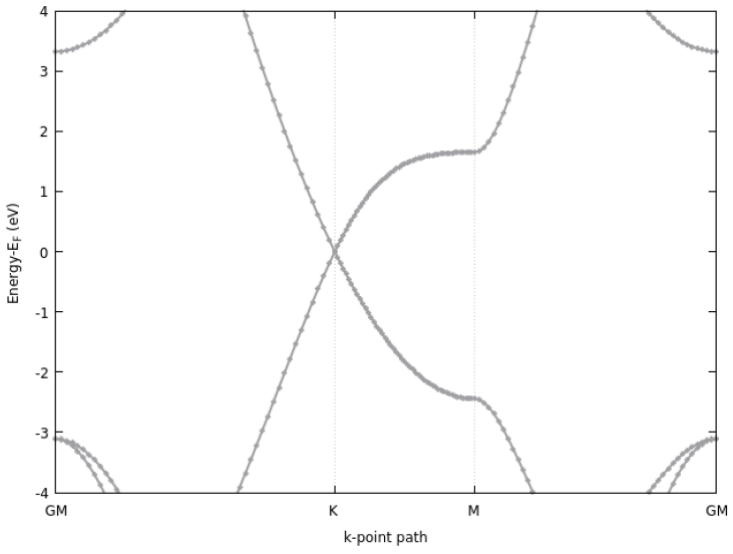
```
$ cat << EOF > bands.in
&bands
prefix = "graphene"
outdir = "./"
filband = "bands"
/
EOF

mpirun -n 4 bands.x < bands.in > bands.out
```

We can now use `gnuplot` to plot the `bands.gnu` file that lists the band energy for each k-point. `ezero` sets the Fermi level to zero of the y-axis of the plot. The location of the tick marks of the x-axis can be taken from `bands.out` file.

```
$ cat << EOF > plot.gnu
ezero = -4.4797
unset key
set ylabel "Energy-E_F (eV)"
set yrange [-4.0:4.0]
set xtics ("GM" 0, "K" 0.6667, "M" 1.0000, "GM" 1.5774)
set grid xtics
set xlabel "k-point path"
plot "bands.gnu" using 1:(column(2)-ezero) w lp lc 0 lw 2 ps 0.5
EOF

$ gnuplot
gnuplot> load 'plot.gnu'
```



Homework:

Check that the electronic band structure is independent of the choice of the artificially large *c* lattice parameter.
Check the assumption that the in-plane lattice parameter *a* of graphene is the same that of graphite, which is the assumption we had in this tutorial.
Check that the band structure is independent of the choice of the k-point mesh in the scf calculation by repeating this calculation for a significantly larger k-point mesh.
Check that the Fermi level is converged with the k-point mesh, e.g. 96x96x2 k-point mesh.
Check that the choice of smearing does not alter the calculated lattice parameters and band structure.
Finally, check that the band structure is still converged within the planewave kinetic energy cutoff.

Acknowledgements

This tutorial is partly based on the tutorial for the **An Introduction to Density Functional Theory (DFT) for Experimentalists** summer school taught by Feliciano Giustino and BP at Cornell University, July 11-18, 2021. The original handouts can be found at this link: https://cornell.app.box.com/s/dtak04lw8s2zfn29061osr5pipo1lkbt. Calculations were performed at Cornell University's CAC Tardis3 supercomputer. BP acknowledges National Science Foundation (Platform for the Accelerated Realization, Analysis, and Discovery of Interface Materials (PARADIM)) under Cooperative Agreement No. DMR-2039380.