

Magnetic Properties from First Principles

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

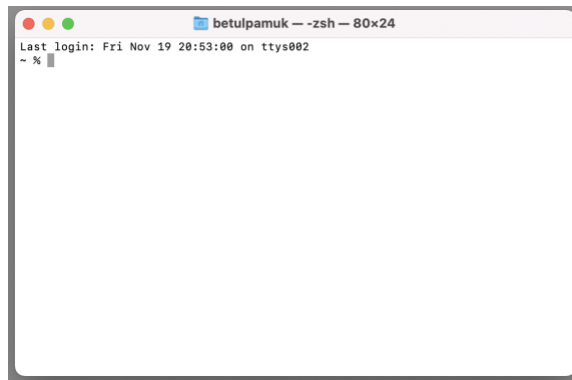
Hands-on tutorial of first-principles calculations

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 (or cluster) 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.

If you are using VPN to connect to TRUBA cluster, use this address:

```
~ % ssh -Y egitim@levrek1.yonetim
```

If you are within the ULAKNET network, to connect to TRUBA cluster, use this address instead:

```
~ % ssh -Y egitim@levrek1.ulakbim.gov.tr
```

You will see a welcome message with recent announcements:

```
egitim@levrek1.yonetim's password:
Last login: Tue Nov 23 21:12:23 2021 from 10.3.9.125
#####
#####
```

DUYURU
2018-09-18

Sayın kullanıcılarımız,

Bugün yaşanan ...

...

Note that `egitim` is my username, which you will need to replace with yours and `levrek1.yonetim` is the IP address of the cluster that I am connecting to.

This lands us in the home directory. We can see the path to where we are by typing:


```
$ pwd
/truba/home/egitim
```

Now we can make a new folder where we will save 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/>



QUANTUM ESPRESSO

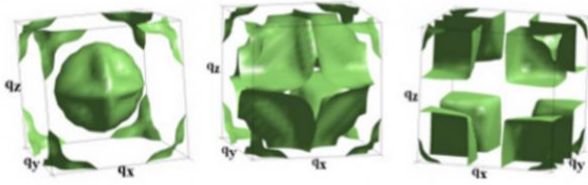
HOME PROJECT DOWNLOAD RESOURCES PSEUDOPOTENTIALS CONTACTS NEWS & EVENTS

NEWS

19.07.21
QUANTUM ESPRESSO V.6.8
The 6.8 version of QUANTUM ESPRESSO is available for download. For more information please see the...

07.04.21
QUANTUM ESPRESSO E-SCHOOL
Applications are open for the MaX e-School on Advanced Materials and Molecular Modelling with Quantu...

30.11.20
QUANTUM ESPRESSO V.6.7
The 6.7 version of QUANTUM ESPRESSO is available for download. For more information please see the...



Phonons of SrTiO₃ under strain, zero-frequency isosurface, Phys. Rev. Lett. 107, 257602 (2011) courtesy of T. Bilro.

QUANTUM ESPRESSO
is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane waves, and pseudopotentials.

The latest version can be found at: <http://www.quantum-espresso.org/download>. To learn how one can compile it even after the summer school, let us 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 192
drwxrwsr-x 6 egitim egitim 4096 Jul 19 12:11 XSpectra
drwxrwsr-x 4 egitim egitim 4096 Jul 19 12:11 XClib
drwxrwsr-x 3 egitim egitim 4096 Jul 19 12:11 UtilXlib
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 upflib
drwxrwsr-x 86 egitim egitim 12288 Jul 19 12:11 test-suite
drwxrwsr-x 7 egitim egitim 4096 Jul 19 12:11 TDDFPT
-rw-rw-r-- 1 egitim egitim 4290 Jul 19 12:11 README.md
-rw-rw-r-- 1 egitim egitim 2481 Jul 19 12:11 README_GPU.md
drwxrwsr-x 5 egitim egitim 4096 Jul 19 12:11 QEHeat
drwxrwsr-x 5 egitim egitim 4096 Jul 19 12:11 PWCOND
drwxrwsr-x 7 egitim egitim 4096 Jul 19 12:11 PW
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 pseudo
drwxrwsr-x 7 egitim egitim 4096 Jul 19 12:11 PP
drwxrwsr-x 7 egitim egitim 4096 Jul 19 12:11 PHonon
drwxrwsr-x 6 egitim egitim 4096 Jul 19 12:11 NEB
drwxrwsr-x 2 egitim egitim 12288 Jul 19 12:11 Modules
-rw-rw-r-- 1 egitim egitim 12055 Jul 19 12:11 Makefile
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 LR_Modules
-rw-rw-r-- 1 egitim egitim 42741 Jul 19 12:11 logo.jpg
-rw-rw-r-- 1 egitim egitim 18009 Jul 19 12:11 License
drwxrwsr-x 3 egitim egitim 4096 Jul 19 12:11 LAXlib
drwxrwsr-x 8 egitim egitim 4096 Jul 19 12:11 KS_Solvers
drwxrwsr-x 3 egitim egitim 4096 Jul 19 12:11 install
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 include
drwxrwsr-x 5 egitim egitim 4096 Jul 19 12:11 HP
drwxrwsr-x 13 egitim egitim 4096 Jul 19 12:11 GWW
drwxrwsr-x 5 egitim egitim 4096 Jul 19 12:11 GUI
drwxrwsr-x 4 egitim egitim 4096 Jul 19 12:11 FFTXlib
drwxrwsr-x 8 egitim egitim 4096 Jul 19 12:11 external
drwxrwsr-x 7 egitim egitim 4096 Jul 19 12:11 EPW
-rw-rw-r-- 1 egitim egitim 5279 Jul 19 12:11 environment_variables
drwxrwsr-x 4 egitim egitim 4096 Jul 19 12:11 Doc
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 dft-d3
drwxrwsr-x 4 egitim egitim 4096 Jul 19 12:11 dev-tools
drwxrwsr-x 5 egitim egitim 4096 Jul 19 12:11 CPV
drwxrwsr-x 6 egitim egitim 4096 Jul 19 12:11 COUPLE
-rw-rw-r-- 1 egitim egitim 1108 Jul 19 12:11 CONTRIBUTING.md
-rwxrwxr-x 1 egitim egitim 2357 Jul 19 12:11 configure
-rw-rw-r-- 1 egitim egitim 28889 Jul 19 12:11 CMakeLists.txt
drwxrwsr-x 3 egitim egitim 4096 Jul 19 12:11 cmake
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 clib
drwxrwsr-x 6 egitim egitim 4096 Jul 19 12:11 atomic
drwxrwsr-x 2 egitim egitim 4096 Jul 19 12:11 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 computer instead of a supercomputer, 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. In MacOS, you can use Homebrew or Macports for this purpose.

Those of us running it on the supercomputer will instead use a module of QE that is already available and compiled with proper optimization flags for us to use from now on.

In a supercomputer, the software is typically available in the form of modules. We can:

see the available modules

```
module avail
```

load the relevant ones, for example:

```
module load centos7.9/lib/openmpi/4.1.1-intel-oneapi-2021.2
```

```
module load centos7.9/app/espresso/6.7.0-openmpi-4.1.1-oneapi-2021.2-GOLD
```

and set up the environment that is already given to us by TRUBA staff:

```
source /truba/sw/centos7.9/comp/intel/oneapi-2021.2/setvars.sh
```

and list the loaded modules:

```
module list
```

Now we all 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, for those of you running your calculations at home, it would be useful to create a shortcut for QE executables:

```

$ cd bin/
$ pwd
/truba/home/egitim/MPFP/q-e-qe-6.8/bin
$ cat >> ~/.bashrc << EOF
export QE="/truba/home/egitim/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>

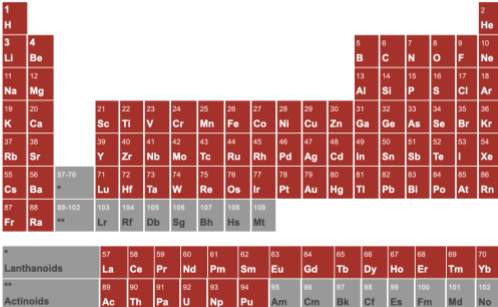
PSEUDOPOTENTIALS

- More about pseudopotentials
- SSSP on Materials Cloud
- Pseudo Dojo
- ONCV Potentials
- SCAN pseudopotentials
- PSLibrary table
- Original QE PP table
- Hartwigesen-Goedecker-Hutter PP table
- Old FHI PP table

PSLIBRARY

Ready-to-use pseudopotentials from **PSLibrary** (recommended). For other ready-to-use tables, follow the links of the menu at the left. For more info, see [here](#).

Please cite the pseudopotentials used and give proper credit to their authors (see [this page](#) for a rather complete list of acknowledgments).



Elements for which at least a pseudopotential is available will appear in red in the periodic table. Click on the element entry and follow the link to access the pseudopotentials and a minimal description of their characteristics.

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

Input File Description

Program: pw.x / PWscf / Quantum Espresso (version: 6.8)

TABLE OF CONTENTS

[INTRODUCTION](#)

&CONTROL

calculation | title | verbosity | restart_mode | wf_collect | nstep | iorint | istress | tornfor | dt | outdir | wfcdir | prefix | lkpoin_dir | max_seconds | etot_conv_thr | forc_conv_thr | disk_io | pseudo_dir | tefield | dfield | leffield | nberrycvs | lorbz | lberry | gdir | nopsr | lfco | gate

&SYSTEM

ibrav | cellldm | A | B | C | cosAB | cosAC | cosBC | nat | ntyp | nbnd | tot_charge | starting_charge | tot_magnetization | starting_magnetization | ecutwfc | ecutrho | ecutfock | nr1 | nr2 | nr3 | nr1s | nr2s | nr3s | nosym | nosym_ewc | noinv | no_l_rev | force_symmorphic | use_all_frac | occupations | one_atom_occupations | starting_spin_angle | degauss | smearing | nspin | noncollin | ecfixed | outz | n2sigma | input_dft | scf | scf_fraction | screening_parameter | exordiv_treatment | x_gamma_extrapolation | ecutvout | nax1 | nax2 | nax3 | localization_thr | lda_plus_u | lda_plus_u_kind | Hubbard_U | Hubbard_U0 | Hubbard_V | Hubbard_aloha | Hubbard_beta | Hubbard_J | starting_nk Eigenvalue | U_projection_type | Hubbard_parameters | ensemble_energies | edir | emaxpos | ecoref | eamp | angles1 | angles2 | fforce | constrained_magnetization | fixed_magnetization | lambda | report | lspinorb | assume_isolated | esm_bc | esm_w | esm_efield | esm_nfl | lgcscf | gcscf_mu | gcscf_conv_thr | gcscf_beta | vdw_corr | london | london_s6 | london_c6 | london_rvdw | london_rcut | effci3_version | effci3_threebody | fs_vdw_econv_thr | fs_vdw_isolated | xdm | xdm_a1 | xdm_a2 | space_group | unique | origin_choice | rhombohedral | zgate | relax | block | block_1 | block_2 | block_height

&ELECTRONS

electron_maxstep | scf_must_converge | conv_thr | adaptive_thr | conv_thr_init | conv_thr_multi | mixing_mode | mixing_beta | mixing_ndim | mixing_fixed_nk | diagonalization | diago_thr_init | diago_cg_maxiter | diago_david_ndim | diago_full_acc | efield | efield_cart | efield_phase | startingpot | startingwfc | tor | real_space

&IONS

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.

It is good practice in supercomputers to run the calculations in a different place than your home directory, typically this is the `scratch` directory. In TRUBA, we will go to:

```
$ cd /truba/scratch/egitim
```

where `egitim` needs to be replaced by your own username.

We will first create a directory to run our test calculation:

```
$ mkdir MPFP; cd MPFP
$ 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,
cellldm(1) = 4.60925103211943136
cellldm(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
```

In a supercomputer, you typically need to run your calculations using a job script. This job script will let the queueing system, SLURM, know how you would like to parallelize your calculations. For TRUBA, it will look like this:

```
$ cat << EOF > job.slurm
#!/bin/bash
#SBATCH -M truba
#SBATCH -p hamsi           #name of the partition
#SBATCH -A egitim         #username
#SBATCH -J graphite_test_n4 #job name - to identify your jobs in the queue
#SBATCH -N 1              #number of nodes
#SBATCH --ntasks=28       #number of processors
#SBATCH --time=0-01:00:00. #time limit to 1 hour
#SBATCH --output=slurm-%j.out
#SBATCH --error=slurm-%j.err

export OMP_NUM_THREADS=1
export OMPI_MCA_btl_openib_allow_ib=1

echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

#This line is commented, because we ran it once when installing QE in the
previous section. If you have skipped that step, uncomment the line below.
#source /truba/sw/centos7.9/comp/intel/oneapi-2021.2/setvars.sh

module load centos7.9/lib/openmpi/4.1.1-intel-oneapi-2021.2
module load centos7.9/app/espresso/6.7.0-openmpi-4.1.1-oneapi-2021.2-GOLD

mpirun -n 4 $ESPRESSO_DIR/bin/pw.x < graphite.scf.in > graphite.scf.out

exit
EOF
```

We enter the file `job.slurm` using the program `vim` and edit the file

```
$ vi job.slurm
```

Once inside the file, you can:

- hit `i` to enter the insert mode
- go with your cursor to the line with the username and change it to your own username
- hit `esc` to enter the command mode
- hit `:wq` to write and quit vim.

You can check the status of your job in the queue with the command:

```
$ squeue -u egitim
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
      7976314      hamsi graphite    egitim R      8:07      1 hamsi60
```

If you want to kill a running job before it finishes, a useful command is:

```
$ scancel JOBID
```

For example, for the job above, it would look like:

```
$ scancel 7976314
```

The QE output file should look like:

```
Program PWSCF v.6.7MaX starts on 24Nov2021 at 0:32: 0

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);
  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
Waiting for input...
Reading input from standard input
...

PWSCF      :      1.30s CPU      1.80s WALL

This run was terminated on: 0:32: 2 24Nov2021

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

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

It is good practice to save the input and output files back into your home directory, since the scratch directory is typically not backed up.

```
$ mkdir ~/MPFP/test_run
$ cp graphite.scf.* ~/MPFP/test_run/
$ cp *.UPF ~/MPFP/test_run/
```

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:

$$\phi_i(\mathbf{r}) = \sum_{\mathbf{G}} c_i(\mathbf{G}) e^{i\mathbf{G}\cdot\mathbf{r}},$$

where \mathbf{G} is the reciprocal lattice vector.

Then Kohn-Sham equation becomes:

$$\frac{|\mathbf{G}|^2}{2} c_i(\mathbf{G}) + \sum_{\mathbf{G}'} V_{tot}(\mathbf{G} - \mathbf{G}') c_i(\mathbf{G}') = \epsilon_i c_i(\mathbf{G}).$$

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:

```
$ cd /truba/scratch/egitim/  
$ 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-005.in  
$ cp ../test_run/job.slurm .
```

We enter the file `graphite-005.in` using the program `vim` and edit the file

```
$ vi graphite-005.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.

Similarly, we need to enter the `job.slurm` file and edit the input and output filenames accordingly.

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

In a local computer, we do this in series by running:

```
$ QE/pw.x < graphite-005.in > graphite-005.out
```

In TRUBA supercomputer, we do this in parallel by submitting to the queue:

```
$ sbatch job.slurm
```

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-005.out
! total energy = -39.74277194 Ry
```

Now you can repeat this exercise for a series of values:

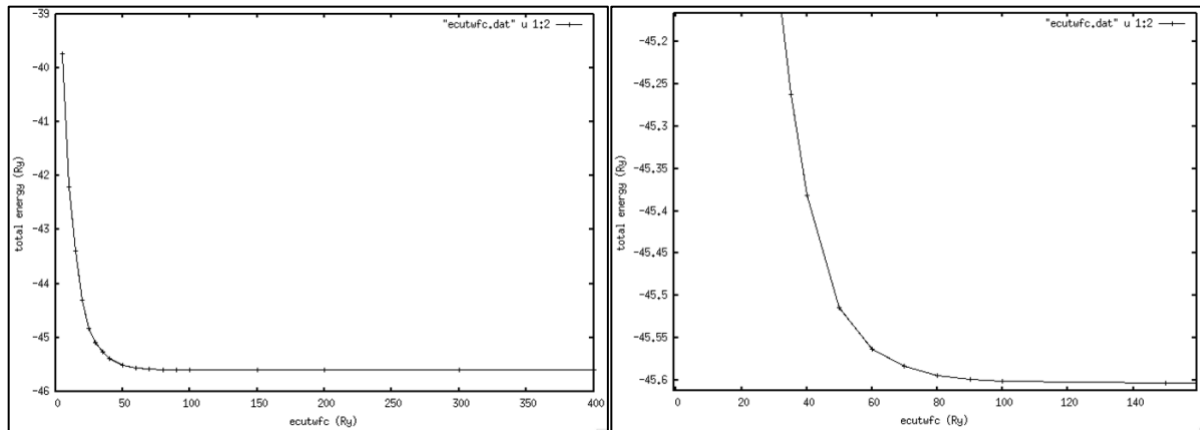
```
ecutwfc = 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 50.0,
60.0, 70.0, 80.0, 90.0, 100.0, 150.0, 200.0, 300.0, 400.0
```

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 plane-wave kinetic energy cutoff.

Finally we can copy all the important files back to our home directory:

```
$ mkdir ~/MPFP/ecut_convergence
$ cp *.in ~/MPFP/ecut_convergence/
$ cp *.out ~/MPFP/ecut_convergence/
$ cp *.UPF ~/MPFP/ecut_convergence/
$ cp ecutwfc.dat ~/MPFP/ecut_convergence/
$ cp job.slurm ~/MPFP/ecut_convergence/
```

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:

$$\begin{aligned} a &= a \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \\ b &= a \begin{pmatrix} -1/2 & \sqrt{3}/2 & 0 \end{pmatrix} \\ c &= c \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

with $a = 2.464 \text{ \AA}$ 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. We copy the same pseudopotential file from the previous directory.

```
$ cd /truba/scratch/egitim/
$ mkdir MPFP/relax_graphite; cd MPFP/relax_graphite
$ cp ../test_run/C.pz-vbc.UPF .
$ cp ../test_run/job.slurm .
```

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

Running in series:

```
$ QE/pw.x < graphite.relax1.in > graphite.relax1.out
```

Running in parallel at TRUBA after appropriately editing the job script file:

```
$ sbatch job.slurm
```

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
 -0.574727823  0.995457789 -0.000000000
  0.000000000  0.000000000  2.739270960

ATOMIC_POSITIONS (crystal)
C          -0.000000000000          -0.000000000000          0.250000000000
C           0.000000000000           0.000000000000          0.750000000000
C           0.333333000000           0.666666000000          0.250000000000
C           0.666666000000           0.333333000000          0.750000000000
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:

$$a = 1.149455645 \times 4.0 = 4.597822580 \text{ Bohr}$$

$$c/a = 2.739270960/1.149455645 = 2.383102795$$

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

```

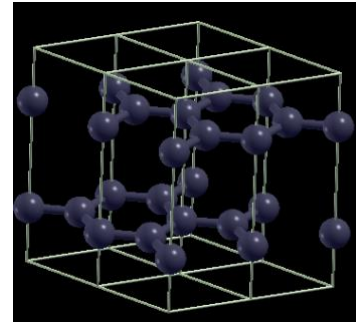
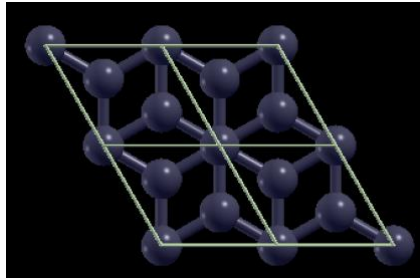
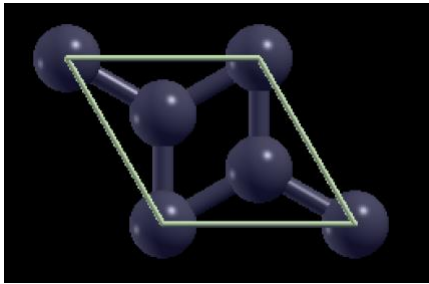
This gives the lattice parameters of $a = 2.439 \text{ \AA}$, $c = 6.524 \text{ \AA}$, which are in overall good agreement with the experimental values of $a = 2.464 \text{ \AA}$, $c = 6.712 \text{ \AA}$. This is an artifact of the LDA approximation in DFT, where LDA tends to overbind the material leading 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

```



Let us copy all the important information back to our home directory:

```
$ cp *.in ~/MPFP/relax_graphite/  
$ cp *.out ~/MPFP/relax_graphite/  
$ cp *.UPF ~/MPFP/relax_graphite/  
$ cp job.slurm ~/MPFP/relax_graphite/
```

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

```
$ cd /truba/scratch/egitim  
$ mkdir MPFP/bands_graphene; cd MPFP/bands_graphene/  
$ cp ../test_run/C.pz-vbc.UPF .  
$ cp ../test_run/job.slurm .
```

```
$ cat << EOF > graphene.scf.in  
&control  
calculation = "scf",  
prefix = "graphene",  
pseudo_dir = "./",  
outdir = "./"  
verbosity = "high"
```

```

/
&system
ibrav = 4,
cellldm(1) = 4.60925103211943136,
cellldm(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"`.

6. 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.

In series you can run:

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

Or in parallel in TRUBA, you can submit the job after appropriately editing the job script file

```
$ sbatch job.slurm
```

We can get the Fermi level of the material from the output of the scf calculation:

```
$ 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,
  cellldm(1) = 4.60925103211943136,
  cellldm(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 0.33333333 0.0 50 !K
0.5 0.0 0.0 50 !M
0.0 0.0 0.0 50 !Gamma
EOF
```

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.

Now we can edit the jobscript file, so it runs the bands calculation

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

and submit it to the queue:

```
$sbatch job.slurm
```

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.

In addition, 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
```

Now we can again edit the jobscript file, so it runs the postprocessing of the bands calculation

```
mpirun -n 4 $ESPRESSO_DIR/bin/bands.x < bands.in > bands.out
```

and submit it to the queue:

```
$sbatch job.slurm
```

Note that we can also edit the job script file to run all three calculations in one submission to the queue:

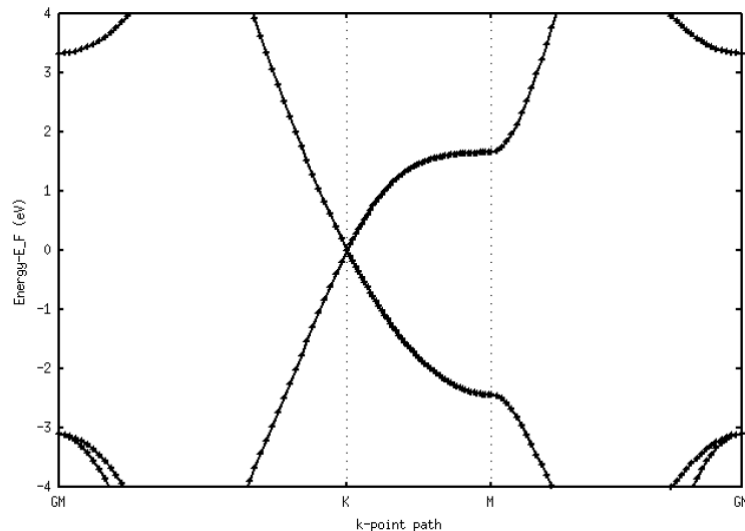
```
mpirun -n 4 $ESPRESSO_DIR/bin/pw.x < graphene.scf.in > graphene.scf.out
mpirun -n 4 $ESPRESSO_DIR/bin/pw.x < graphene.bands.in > graphene.bands.out
mpirun -n 4 $ESPRESSO_DIR/bin/bands.x < bands.in > bands.out
```

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-EF (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'
```



We again copy all the files back to our home directory:

```
$ mkdir ~/MPFP/bands_graphene
$ cp *.in ~/MPFP/bands_graphene/
$ cp *.out ~/MPFP/bands_graphene/
$ cp *.UPF ~/MPFP/bands_graphene/
$ cp bands.gnu ~/MPFP/bands_graphene/
$ cp plot.gnu ~/MPFP/bands_graphene/
$ cp job.slurm ~/MPFP/bands_graphene/
```

If you would like to make a copy of these files to your local computer for your reference in the future, we can compress them using the command:

Go to your home directory by typing:

```
$ cd
```

and compress the files:

```
$ tar czvf MPFP.tar MPFP/
```

Then using the `scp` command with the same IP address that we have used to `ssh`, we can copy them, running the command below in our local computer (not in the cluster)

```
betul-laptop:$ scp egitim@levrek1.yonetim:~/MPFP.tar .
egitim@levrek1.yonetim's password:
```

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.

Note that DFT performs well for metals, and when compared with angle-resolved photoemission spectra (ARPES) measurements. For semiconductors and insulators, standard DFT calculations typically underestimates the band gaps. On the other hand Hartree-Fock approximation typically overestimates the band gaps. Therefore, hybrid functionals that mix the standard DFT functionals with the Fock exchange gives more reasonable band gaps as compared to the experiments.

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/dtak04lw8s2zfn29061osr5pip01lkbk>. Initial test calculations were performed at Cornell University's CAC Tardis3 supercomputer, while the school was run at the Türk Ulusal Bilim e-Altyapısı (TRUBA) supercomputer. BP acknowledges National Science Foundation (Platform for the Accelerated Realization, Analysis, and Discovery of Interface Materials (PARADIM)) under Cooperative Agreement No. DMR-2039380.