Lab 2

First-principles calculations of the electronic properties of materials: The case study of bulk magnesium oxide*

Sebastiaan Huber and Davide Campi
École Polytechnique Fédérale de Lausanne (THEOS)

11 April 2018

Contents

*These notes is the modified version of the notes written by former lecturers of this lab.

1 First-principles codes

There are several popular first-principles codes, and some of them are listed below:

- QUANTUM ESPRESSO (http://www.quantum-espresso.org/). This is a DFT plane-wave pseudopotential code. It is distributed under the GPL license. A brief description about this package is given below.
- ABINIT (http://www.abinit.org/). This is a DFT plane-wave pseudopotential code. It is also distributed under the GPL license.
- CP2K (https://www.cp2k.org/). This is a DFT code using the mixed Gaussian and plane-waves approaches. It is also distributed under the GPL license.
- CPMD (http://www.cpmd.org/). This is a DFT plane-wave pseudopotential code implementing Car-Parrinello molecular dynamics. It is also distributed under the GPL license.
- SIESTA (http://www.uam.es/siesta/). This is a DFT code that uses localized atomic basis sets. It is free for academics.
- VASP (http://www.vasp.at/). This is a DFT plane-wave pseudopotential code. Available with a moderate cost for academics.
- WIEN2k (http://www.wien2k.at/). This is a DFT Full-Potential Linearized Augmented Plane-Wave method (FLAPW). FLAPW is the most accurate implementation of DFT, but the slowest. Available with a small cost for academics.
- Gaussian (http://www.gaussian.com/). This is a quantum chemistry code that includes Hartree-Fock (HF) and higher-order correlated electrons approaches. Moderate cost for academics.
- Crystal (http://www.crystal.unito.it/). This is a HF and DFT code. Small cost for academics.
- Molpro (http://www.molpro.net/). This is a DFT code. Small cost for academics.

An extended list of codes: http://nomad-coe.eu/index.php?page=codes

In this lab we will use the QUANTUM ESPRESSO package [1], which is one of the most popular softwares in the condensed matter community. QUANTUM ESPRESSO is an integrated suite of *open-source* computer codes for electronic structure calculations and materials modeling at the nanoscale. ESPRESSO stands for *opEn Source Package for Research in Electronic Structure, Simulation, and Optimization*. It is based on density functional theory (DFT), plane waves, and pseudopotentials. It allows one to model structural and electronic properties of materials, vibrational properties (phonons), spectroscopic properties (absorption spectra, electron energy loss spectra, etc.), perform molecular dynamics (Born-Oppenheimer, Car-Parrinello, Langevin) and do many other things.

The PWscf program (**pw.x**) is one of the *core* components of QUANTUM ESPRESSO and it is based on DFT for calculations of the *ground state properties* of materials, *e.g.* total energy, forces, stress tensor, etc. The rest of this handout will demonstrate how to compute the total energy of a bulk magnesium oxide (MgO) using the PWscf program.

There are many tutorials about QUANTUM ESPRESSO(especially the hands-on tutorial in Santa Barbara 2009): http://www.quantum-espresso.org/tutorials/

check that is compatible with new enviroment

For the calculations related to Lab 2 you will use the same settings which you used for Lab 1.

On each of the computers of the room, there is a linux (Ubuntu) virtual machine installed. During this lab, we will work at two places: 1) Inside the virtual machine, and 2) by SSH tunneling through the virtual machine to EPFL's supercomputer system: **Deneb**. In both cases you will need to be first logged in to your virtual machine. In the beginning we will stay in the virual machines and get familiar with the concepts. Later we will move our setting to Deneb.

To run the virtual machine, you need to open the Oracle VirtualBox program, and start the THEOS machine from the list of the available virtual machines.

Important reminder: Every file saved inside the virtual machine will remain inside it, and it is thus not easy to obtain. For this reason, we have mapped a shared folder on the disk **/Users/profile/Desktop** of your Mac physical computer onto the folder /EDRIVE of the Linux virtual machine. This means that if you write a file inside /EDRIVE, this will appear inside a folder on `/Users/profile/Desktop` when you switch to Mac. We thus recommend that, inside the Linux Virtual Machine, you save all your files always inside a folder you should create for yourself inside /EDRIVE. Remember also that all files saved on `/Users/profile/Desktop` are available ONLY on the machine on which you are working, and are ACCESSIBLE BY EVERYBODY. Therefore, at the end of each class, we strongly encourage you to **save a copy of all your files in your home folder, in a USB stick or in any other safe location.**

At the first step, launch the Oracle VirtualBox program.

At the second step, from Mac copy the provided LAB2 directory inside the folder shared with the virtual machine, i.e. `/Users/profile/Desktop`. **Note:** this step must be done once the VirtualBox is already running, because when you launch the VirtualBox it deletes all files inside of `/Users/profile/Desktop`.

At the third step, you can click on the virtual machine to start working in Linux with the files in /EDRIVE which is linked to `/Users/profile/Desktop` of Windows. If you like, you can press (RIGHT CTRL) + F to open linux in full screen. When you will need to go back to Mac, press the same keys again.

Finally, in Linux, open a terminal, and go to the LAB2 directory:

```
theos@theosvm:~$ cd EDRIVE
theos@theosvm:~/EDRIVE$ cd LAB2
```

2 Getting started

2.1 Input file for the PWscf program

Once you are in your directory for this lab, you can create another directory inside and do a test there to see how PWscf works. To do that, we first create the directory `Test` and then copy the necessary input file for PWscf in that directory:

```
theos@theosvm:~/EDRIVE/LAB2$ mkdir Test
theos@theosvm:~/EDRIVE/LAB2$ cd Test
theos@theosvm:~/EDRIVE/LAB2/Test$ cp ../Examples/MgO.scf.in .
theos@theosvm:~/EDRIVE/LAB2/Test$ cp -r ../Examples/pseudo .
```

File MgO.scf.in contains the input information needed to compute the total energy per unit cell of magnesium oxide using PWscf, and the directory /pseudo contains the pseudopotential files Mg.pbe.UPF and O.pbe.UPF If you view the file, it will look like this:

```
theos@theosvm:~/EDRIVE/LAB2/Test$ less MgO.scf.in
```

```
&control
    calculation = 'scf'
    restart_mode = 'from_scratch'
    prefix = 'MgO'
    tstress = .true.
    tprnfor = .true.
    pseudo_dir = './pseudo/'
    outdir = './tmp/'
/
&system
    ibrav = 2
    celldm(1) = 7.969
    nat = 2
    ntyp = 2
    ecutwfc = 18.0
/
&electrons
    diagonalization = 'david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-8
/
ATOMIC_SPECIES
    Mg  24.305   Mg.pbe.UPF
    O   15.9994  O.pbe.UPF
ATOMIC_POSITIONS {alat}
```

```
    Mg 0.00 0.00 0.00
    O  0.50 0.50 0.50
K_POINTS {automatic}
    4 4 4 0 0 0
```

All the input parameters are described here:
http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PW.html

The most relevant lines for the calculations related to this lab are the following:

- The line

  ```
  calculation = 'scf'
  ```

  indicates that this is a self-consistent calculation (SCF) to find the total energy of the system. Other options can be found using the link above.

- The line

  ```
  prefix = 'MgO'
  ```

  is a tag the code uses to identify this calculation.

- The line

  ```
  outdir = './tmp/'
  ```

  indicates the path to the directory, where all temporary files of the code (such as wavefunctions, eigenvalues, etc.) will be written during the execution.

  Note: If you give the same path for `outdir` for two different calculations with the same `prefix`, they will overwrite the temporary files and the calculation will crash. When you run several calculations at the same time, remember to give them different `prefix`, or you can use different temporary directories.

- The line

  ```
  pseudo_dir = './pseudo/'
  ```

  indicates the path to the place, where the pseudopotential file is located. We provide the necessary pseudopotential (`Mg.pbe.UPF` and `O.pbe.UPF`) within `LAB2/Examples/pseudo`, just make sure that the path to them is correct in your input file.

- The line

  ```
  tstress = .true.
  ```

  indicates that the stress tensor will be computed.

- The line

  ```
  tprnfor = .true.
  ```

indicates that forces will be computed.

- The line

  ```
  ibrav = 2
  ```

  indicates the Bravais-lattice index. In this case `ibrav = 2` means that we are dealing with the face-centered cubic (fcc) primitive unit cell.

- The line

  ```
  celldm(1) = 7.969
  ```

  is the value of the lattice parameter in Bohr atomic units.

- The line

  ```
  nat = 2
  ```

  indicates the **n**umber of **at**oms in the unit cell.

- The line

  ```
  ntyp = 2
  ```

  indicates the **n**umber of **typ**es of atoms in the unit cell.

- The line

  ```
  ecutwfc = 18.0
  ```

  indicates the value of the kinetic-energy cutoff in Rydberg units. This value determines how many plane waves will be used in the expansion of Kohn-Sham wavefunctions during the iterative solution of the Kohn-Sham equations.

- The lines

  ```
  ATOMIC_SPECIES
  Mg 24.305 Mg.pbe.UPF O 15.9994 O.pbe.UPF
  ```

  indicate the label of the atom, its mass, and the name of the pseudopotential file.

- The lines

  ```
  diagonalization = 'david'
  mixing_mode = 'plain'
  mixing_beta = 0.7
  ```

are the parameters which specify the numerical detailes for the iterative solution of the Kohn-Sham equation. The final result (e.g. the total energy) must not depend on the value of these parameters. These default values were optimized (for all systems, not only MgO) and should not be changed. However, there are cases when one has to change these parameters. In this lab, there will be no need to modify these parameters.

- The line

  ```
  conv_thr = 1.0d-8
  ```

  indicates the convergence threshold in Rydberg units during the iterative solution of the Kohn-Sham equation. When this threshold is reached, the calculation will stop.

- The lines

  ```
  ATOMIC_POSITIONS {alat}
  Mg 0.00 0.00 0.00
  O 0.50 0.50 0.50
  ```

  indicate the atomic positions in the unit cell in units of the the lattice parameter (`celldm(1)`).

- The lines

  ```
  K_POINTS {automatic}
  4 4 4 0 0 0
  ```

  indicate the **k** points sampling of the Brillouin zone. In this case, the automatic $4\times4\times4$ **k** point sampling will be used using the Monkhorst-Pack procedure [4]. The last three numbers indicate that no shift with respect to the center of the Brillouin zone will be made.

2.2 Visualizing the crystal

Before you start your first calculation, let us introduce a very useful visualization tool called XCrySDen (http://www.xcrysden.org/). XCrySDen is a crystalline and molecular structure visualization program that allows us to visualize our structure and measure its certain properties such as inter atomic distances and angles. In the test folder, try typing

```
xcrysden --pwi MgO.scf.in
```

which launches the XCrySDen program, and tells to it that we want to visualize a PWscf input (–pwi) and the name of the file is MgO.scf.in. The program will read the above input file we have seen and visualize it. Some useful features are:

- Dimensions pop-up: The first thing you see when you launch the program is to specify if you would like to visualize a 0D, 1D, or 2D structure. In this exercise we are visualizing a 3D crystal so choose the ‘‘`do not reduce dimensionality`’’ option and click `OK`.

- Display tab on top have many features. Choose to display the crystal cell. You can click and rotate the cell and visualize it from different angles. Experiment with it. Although trivial for this example, visualization tools can be very useful for complicated systems.
- Go to `Modify` tab and further down into `Number of units drawn`. Play with the numbers there to get a feel of your crystal.
- `Atoms Info, Distance, Angle, Dihedral` tabs below allow you to get info on the atoms or measure distances/angles between atoms.

2.3 Running a PWscf example and obtaining the total energy

Now you have seen the crystal structure of bulk MgO, and we are ready to run our first first-principles calculation with PWscf. Close the XCrySDen program and just type

```
pw.x < MgO.scf.in > MgO.scf.out
```

After a few seconds (if everything goes well) you should have in your directory a new file called `MgO.scf.out`. This is the output file of your calculation.

If you view the file `MgO.scf.out`, `(less MgO.scf.out)` or `(vi MgO.scf.out)`, you will find that in the beginning of the file there is a lot of information which summarizes the calculation, such as number of atoms, unit cell size, number of plane waves etc. Then you can follow the output file and see how at each iteration the code finds a solution with a lower energy. And close to the end, a line containing something like

```
!    total energy              =     -74.62919601 Ry
```

will appear. So here you can see the total energy for the unit cell as the final result from the program. Note that the final occurrence of total energy will have an exclamation mark by it, something that makes this line easy to find in a long output file, for example by typing, on the terminal, the following:

```
grep !  *.out
```

which will search all files terminated by .out in the current directory, looking for lines that have an exclamation mark in them.

There is other useful information in the output file. View it again `(less MgO.scf.out)` and go to the end. For example, it is written how much CPU and WALL time the PWscf program spends for various parts of the calculation:

```
init_run     :      0.02s CPU      0.23s WALL (       1 calls)
electrons    :      0.09s CPU      0.14s WALL (       1 calls)
forces       :      0.00s CPU      0.00s WALL (       1 calls)
stress       :      0.00s CPU      0.00s WALL (       1 calls)
```

```
.
.
.
PWSCF        :      0.19s CPU          0.53s WALL
```

As the lectures progress you will get more familiar with the output format and all the information written.

2.4 Convergence of the total energy with respect to various input parameters

The calculation of various quantities for a given system (e.g. the total energy in this case) must be checked for a convergence with respect to:

- Kinetic-energy cutoff `ecutwfc`;
- **k** points sampling of the Brillouin zone;
- Other parameters (e.g. smearing), which are not needed in our case.

It is meaningful to speak about the convergence only when we specify the precision. It is not precise to say: "My calculations of the total energy are converged". Instead, it is correct to say: "My calculations of the total energy are converged with a precision, say, of 10^{-3} Ry with respect to the cutoff and **k** points". This means that if you increase the value of the kinetic-energy cutoff or the density of the **k** points sampling, the value of the total energy will not change more than 10^{-3} Ry.

Let us remind some basic concepts in order to understand better about the convergence. A very good explanation of the concepts is given, e.g., in Ref. [5]. Another excellent book is [2].

2.4.1 Plane-wave expansion

Remember that we are dealing with infinite systems using periodic boundary conditions. This means that we can use the Bloch theorem to help us to solve the (Schrödinger-like) Kohn-Sham equation. The Bloch theorem states that the wavefunction can be written in the form:

$$\psi_{n,\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}} u_{n,\mathbf{k}}(\mathbf{r}), \tag{1}$$

where n is the band index, $\mathbf{k}$ is the point in the Brillouin zone, $e^{i\mathbf{k}\cdot\mathbf{r}}$ is the phase, and $u_{n,\mathbf{k}}(\mathbf{r})$ is the *lattice-periodic* part of the wavefunction which can be written as

$$u_{n,\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} c_{n,\mathbf{k}+\mathbf{G}}\, e^{i\mathbf{G}\cdot\mathbf{r}}, \tag{2}$$

$$u_{n,\mathbf{k}}(\mathbf{r}+\mathbf{R}) \equiv u_{n,\mathbf{k}}(\mathbf{r}), \tag{3}$$

where $\mathbf{G}$ are the reciprocal lattice vectors, $\mathbf{R}$ is the vector of the unit cell in real space, and $c_{n,\mathbf{k}+\mathbf{G}}$ are the coefficients of the expansion. The sum in Eq. (2) runs, in principle, over

infinite number of the reciprocal lattice vectors **G**. In practice, however, one has to truncate the sum (*plane wave cutoff*).

The basis functions, which we use to make an expansion in Eq. (2), are *plane waves*. They are called "plane waves", because they are surfaces of the constant phase and are parallel planes perpendicular to the direction of the propagation. The values of **G** are integer multiples of the three primitive lattice vectors, and hence they are compatible with the periodic boundary conditions of our direct lattice.

In actual calculations, we have to limit the plane-wave expansion by specifying the plane wave cutoff, which must be given in energy units (such as Rydberg or eV):

$$\frac{\hbar^2}{2m}|\mathbf{k}+\mathbf{G}|^2 \leq E_{\text{cut}}, \quad E_{\text{cut}} = \frac{\hbar^2}{2m}G_{\text{cut}}^2, \tag{4}$$

where $\hbar$ is the Planck constant, m is the electron mass, E_{cut} is the cutoff kinetic-energy, and G_{cut} is the cutoff reciprocal lattice vector. Given Eq. (4), Eqs. (1) and (2) read:

$$\psi_{n,\mathbf{k}}(\mathbf{r}) = \sum_{|\mathbf{k}+\mathbf{G}|\leq G_{\text{cut}}} c_{n,\mathbf{k}+\mathbf{G}}\, e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}}. \tag{5}$$

Problem 1 is designed to test cutoff convergence issues. The relevant keyword in the input file is `ecutwfc`. You can always choose a higher cutoff in order to increase the precision, and this will increase the CPU time of your calculations.

One always has to perform tests for convergence of the total energy (and other quantities of interest) with respect to the kinetic-energy cutoff!

2.4.2 k points sampling of the Brillouin Zone

In order to solve the Kohn-Sham equations, one must calculate various integrals in the reciprocal space over **k** vectors ($\int d\mathbf{k}(...)$). In practice, the integrals over **k** vectors are replaced by the summations over descrite **k** points ($\sum_{\mathbf{k}}(...)$). Ideally, the number of **k** points must be infinite, but in practice one has to use a finite number and make tests for convergence.

In the output file of your calculation (`MgO.scf.out`) you will find something like this:

```
 number of k points=     8
                     cart. coord. in units 2pi/alat
     k(    1) = (   0.0000000   0.0000000   0.0000000), wk =   0.0312500
     k(    2) = (  -0.2500000   0.2500000  -0.2500000), wk =   0.2500000
     k(    3) = (   0.5000000  -0.5000000   0.5000000), wk =   0.1250000
     k(    4) = (   0.0000000   0.5000000   0.0000000), wk =   0.1875000
     k(    5) = (   0.7500000  -0.2500000   0.7500000), wk =   0.7500000
     k(    6) = (   0.5000000   0.0000000   0.5000000), wk =   0.3750000
     k(    7) = (   0.0000000  -1.0000000   0.0000000), wk =   0.0937500
     k(    8) = (  -0.5000000  -1.0000000   0.0000000), wk =   0.1875000
```

This is a list of the irreducible (non-equivalent) **k** points that were used to sample the irreducible Brillouin zone. Remember, we specified in the input the uniform Monkhorst-Pack

k points sampling of the Brillouin zone: $4 \times 4 \times 4 = 64$, but we see here only 8 **k** points. This is so, because some of these 64 **k** points are equivalent due to symmetry (give the same Kohn-Sham energy). Consider an example of a cube: it has 8 identical corners, so if the original **k** point mesh includes 8 corner points, it is computationally more convenient (to save CPU time) to calculate the energy using one of these corner points and weighing it with the appropriate multiplicity of that particular **k** point (the corner point of a cube has the multiplicity 8).

Different **k** points in the original 64 point mesh have different multiplicities in the Brillouin zone of the crystal. The **pw.x** code, using the symmetry, finds these multiplicities and lists the non-equivalent **k** points and the corresponding weights (`wk`). The weights add up to 2, because we perform unpolarized calculation, and, hence, every Kohn-Sham state is occupied by two electrons due to the Pauli principle (spin degeneracy).

The list of **k** points will be different, if you use a different **k** point mesh ($6\times6\times6$, $8\times8\times8$, etc.). **One always has to perform tests for convergence of the total energy (and other quantities of interest) with respect to the number of k points!**

3 Using bash scripts to run PWscf multiple times and with many processors

In order to check the numerical convergence of your calculations with respect to kinetic-energy cutoff and the number of **k** points, you will have to run PWscf multiple times with different values for those parameters. Of course you may create a new input file for every set of parameters, but there are faster, automatized ways of doing this. We would like to introduce to you one of them: *bash scripting*.

You can either write a bash script by yourself or use the one from the example provided in `/LAB2/Examples` directory. The script looks like this:

```
#!/bin/bash

# Input data:
LISTA="7.97"            # List of values of lattice parameter to try
LISTECUT="18 36"        # List of plane-wave cutoffs to try
LISTK="2 4"             # List of number of k-points per dimension to try

# Files of interest:
TMP_DIR="./tmp"         # where temporary data will be stored
PSEUDO_DIR="./pseudo"   # where pseudopotentials are stored
OUT_DIR="./Test_script" # where input and output will be
                        # created once the script runs.

PW_LAUNCH='pw.x'        # This is QE executable file.

# Security checks:
if [ ! -d $TMP_DIR ]; then
```

```
   mkdir $TMP_DIR
fi

if [ ! -d $OUT_DIR ]; then
   mkdir $OUT_DIR
fi

# Start loops on plane-wave cutoffs, k-point grids, and lattice constants:
for ecut in $LISTECUT
do
for k in $LISTK
do
for a in $LISTA
do

echo 'Doing ecut, k, a: ' $ecut $k $a
# Create new input file:
cat > $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.in << EOF
      &control
         calculation = 'scf'
         restart_mode = 'from_scratch'
         prefix = 'MgO.$a.$ecut.$k'
         tstress = .true.
         tprnfor = .true.
         pseudo_dir = '$PSEUDO_DIR'
         outdir='$TMP_DIR'
      /
      &system
         ibrav = 2
         celldm(1) = $a
         nat = 2
         ntyp = 2
         ecutwfc = $ecut
      /
      &electrons
         diagonalization = 'david'
         mixing_mode = 'plain'
         mixing_beta = 0.7
         conv_thr = 1.0d-8
      /
      ATOMIC_SPECIES
        Mg  24.305   Mg.UPF
        O   15.9994  O.UPF
      ATOMIC_POSITIONS {alat}
        Mg 0.00 0.00 0.00
```

```
        O  0.50 0.50 0.50
      K_POINTS {automatic}
         $k $k $k  0 0 0
EOF

# Run PWscf to create new output file:
$PW_LAUNCH < $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.in > ...
      ... $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.out

# Finish loops on plane-wave cutoffs, k-point grids, and lattice constants:
done
done
done

rm -r $TMP_DIR/*
```

You can see that there are three lists of input parameters that will be created in order:

```
LISTA="7.97"
LISTECUT="18 36"
LISTK="2 4"
```

The bash script will loop over these values to perform (in this case) four different calculations automatically, without having to change the input files by hand.

You can run the script by doing

```
sh script.sh &
```

or

```
nohup ./script.sh &
```

(the `&` symbol at the end of the line allows you to keep using the terminal without having to wait for the work to finish, and the `nohup` part allows the program to keep running even if you close the terminal in the computer). At the end you will find three more output files in your output directory. Go in the output directory and type `ls -l` to get a list of the produced files.

In the problems that you will have to work, you can modify the above script accordingly to do different calculations you need.

4 Running PWscf on the supercomputer "Deneb"

Above we have shown how to run your calculations on the virtual box. Your virtual box has access to only one processor, but you can speed up your calculations by using the High Performance Computer (HPC) systems, such as the one of EPFL called "Deneb". Here we demonstrate how to prepare a bash script for Deneb, and run the same calculations as above.

In a HPC system there are usually thousands of computing units and tens or hundreds of users, and most of the time there are more calculations to be done than the amount of resources available. Here is how an HPC system solves this problem: The use of the processors is managed by a special scheduler system (on Deneb it is called SLURM). Users (you) prepare job files and submit to this scheduler. Then scheduler places your jobs in the queue according to how many resources your job asks for, and how long your calculation is expected to run for, or sometimes only due to the permissions you are given on that HPC system. Then, in that queue, a priority number is assigned to your job, much like getting a ticket while waiting in line at the post office. When it is your job's turn, SLURM allows it and your job starts running. You can submit several jobs simultaneously (**Important note:** remeber to take care about `prefix` and `outdir` for different jobs in order not to have problems of mixing different temporary files during runs).

Now let us see how we can submit the above calculations, this time on Deneb. To do that first we login to Deneb with our accounts using a standard terminal.

The login command is:

```
ssh your_user_name@deneb1.epfl.ch
```

then type your password:

```
Password: ******
```

Once successfully logged in, you will be at your `HOME` directory on Deneb-HPC, by typing:

```
pwd
```

you will see:

```
/home/your_user_name
```

This `HOME` directory is only able to read from the computing nodes. The `HOME` directory is a good place to store important files, but it is never a good place to run calculations. To run calculations you need to go to place on the HPC system where you are allowed to execute the **pw.x** code, write the PWscf temporary files etc. For the sake of computing time, all this should happen on a place with very fast communication. Each supercomputer has a place like this, and most often they are called the *scratch areas*. Let us go to our scratch area on Deneb:

```
cd /scratch/your_user_name/
```

Let us start by copying the directory Examples_deneb into this scratch area. Type:

`scp -r Lab2 your`$_u ser_n ame@deneb1.epfl.ch : /scratch/your_u ser_n ame??$
On the `/scratch/your_user_name/` directory you will now find the downloaded Examples_deneb directory with the bash script and related directories. This script looks like this:

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 8
#SBATCH --cpus-per-task 1
#SBATCH --time=00:30:00
#SBATCH --mem=14000
#SBATCH --partition=mse-468
#SBATCH --account=mse-468
#SBATCH --reservation=mse-468

#These are the libraries necessary for our code to run
module purge
module load  intel/17.0.2
module load  intel-mpi/2017.2.174
module load  intel-mkl/2017.2.174
module load  fftw/3.3.6-pl2
module load  espresso/6.1.0-mpi


LISTA="7.97"        # List of values of lattice parameter to try
LISTECUT="18 36"    # List of plane-wave cutoffs to try.
LISTK="2 4"         # List of number of k-points per dimension to try.

#
# Files of interest:
TMP_DIR="./tmp"        # where temporary data will be stored.
PSEUDO_DIR="./pseudo"  # where pseudopotentials are stored.
OUT_DIR="./Test"       # where input and output will be
                       # created once the script runs.

# Security checks:
if [ ! -d $TMP_DIR ]; then
   mkdir $TMP_DIR
fi
if [ ! -d $OUT_DIR ]; then
   mkdir $OUT_DIR
fi


# Start loops on plane-wave cutoffs, k-point grids, and lattice constants:
for ecut in $LISTECUT
do
for k in $LISTK
do
for a in $LISTA
```

```
do

#### Choose on how many processors to run depending on the system size
####

PW_LAUNCH="srun pw.x"

# Create new input file:
cat > $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.in << EOF
      &control
         calculation = 'scf'
         restart_mode = 'from_scratch'
         prefix = 'MgO.$a.$ecut.$k'
         tstress = .true.
         tprnfor = .true.
         pseudo_dir = '$PSEUDO_DIR'
         outdir='$TMP_DIR'
      /
      &system
         ibrav = 2
         celldm(1) = 7.97
         nat = 2
         ntyp = 1
         ecutwfc = $ecut
      /
      &electrons
         diagonalization = 'david'
         mixing_mode = 'plain'
         mixing_beta = 0.7
         conv_thr = 1.0d-8
      /
      ATOMIC_SPECIES
        Mg  24.305   Mg.UPF
        O   15.9994  O.UPF
      ATOMIC_POSITIONS {alat}
        Mg 0.00 0.00 0.00
        O  0.50 0.50 0.50
      K_POINTS {automatic}
         $k $k $k  0 0 0
EOF

# Run PWscf to create new output file:
$PW_LAUNCH < $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.in > ...
        ... $OUT_DIR/MgO.scf.a=$a.ecut=$ecut.k=$k.out
```

```
# Finish loops on plane-wave cutoffs, k-point grids, and lattice constants:
done
done
done

# Clean up:
rm -rf $TMP_DIR/*
```

You can see that the main part of the bash script is similar to the one used to run on the single CPU computer. The differences are mostly from the first part:

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 16
#SBATCH --cpus-per-task 1
#SBATCH --time=00:30:00
#SBATCH --mem=14000
#SBATCH --partition=mse-468
#SBATCH --account=mse-468
#SBATCH --reservation=mse-468

#These are the libraries necessary for our code to run
module purge
module load  intel/17.0.2
module load  intel-mpi/2017.2.174
module load  intel-mkl/2017.2.174
module load  fftw/3.3.6-pl2
module load  espresso/6.1.0-mpi
```

The above lines are used to communicate with the schedule, to setup the number of CPUs, maximum time, and the libraries that the Deneb-HPC need to run PWscf jobs. Moreover, the line

```
PW_LAUNCH="srun pw.x"
```

tells you what the executable **pw.x** file is. You have to be sure to have indeed downloaded the executable at the directory from which you are sending the script. Otherwise you can specify the location you downloaded the executable by changing the value of PW_LAUNCH with an expression like "*/scratch/user/.../pw.x*".

Now our job file is ready, we are ready to submit it in the terminal that is connected to Deneb:

```
sbatch script_deneb.sh
```

To see the status of your jobs, you do can type:

```
squeue -u $USER
```

To kill a job before its completion, you can write

```
scancel <job_id_number>
```

When all jobs are finished, you can copy the output files back to your virtual box, by opening a new terminal in the virtual box, and then typing the command:

```
scp -r your_user_name@deneb1.epfl.ch:/scratch/your_user_name/
Examples_deneb/Test ~/EDRIVE/LAB2/Examples_deneb/Test
```

(make sure to create the directory on your hardrive beforehand).
Note that by default, due to your permissions, your jobs are assigned to a fast queue on Deneb, but this is true only during the lab hours. After the lab hours, you maybe not able to login to the system anymore. Therefore try to use your lab hours as efficiently as possible. Outside the lab hours, you should run the calculations on the lab or you own machines. To install QUANTUM ESPRESSO on laptop, you can ask to your teaching assistants.

5 Hints for solving the problems

In order to solve the problems in Lab 2, you will have to run several PWscf calculations. In principle, you can organize directories for saving the output runs in any way you like. One way is to make one directory for each problem you do, in that case if you want to use the above script you would need to change the OUTPUT in the script accordingly. A good organization and good knowledge about bash script may save your time. If you have any questions regarding the bash script please do not hesitate to ask.

Problem 1

In the problem 1 you must check the convergence of the total energy with respect to the kinetic-energy cutoff. In your calculations, you should start from some rather small value of `ecutwfc` (say, 10 Ry), and increase it by 10 Ry. All other input parameters must be kept fixed. By doing so, you have to observe how changes the value of the total energy at various values of `ecutwfc`. You must reach the convergence to a specified precision.

So, you should set `LISTK` to only one value (`'4'` if you are using the provided bash script). Or if you are preparing your input files manually, keep the **k** point mesh always as 4 4 4 0 0 0. In order to try different cutoffs, modify the content of `LISTECUT` so that different numbers are included. For example, you can start with 10 Ry and append the higher cutoffs up to 80 Ry with intervals of 10 Ry. Check the total energies to decide when the calculations are converged.

Problem 2

Problem 2 will test the convergence of the force on atoms with respect to the plane-wave cutoff. In problem 1, the forces on Mg and O are zero in the x, y, and z directions. This is because of symmetry, which cancels out forces. (Forces are written at the end of the output file for each atom in the unit cell, you can check that they are indeed zero.) In problem 2, we will create forces by displacing a O atom +0.05 (fractional coordinates) in the z direction. To do this, we need to edit the z coordinate of the O ion in the input file. After you edit the script, the corresponding lines should look like this:

```
ATOMIC_POSITIONS {alat}
  Mg 0.00 0.00 0.00
  O  0.50 0.50 0.55
```

Now put `'30'` in `LISTECUT` so that the cutoff is 30 Ry, and put `'4'` in `LISTK` so that the **k** point grid is $4 \times 4 \times 4$. Rerun the script, and record the forces. The forces will appear in the output file after the total energies. It will look something like this:

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

atom    1 type  1   force =     0.00000000    0.00000000    0.13771493
atom    2 type  1   force =     0.00000000    0.00000000   -0.13771493

Total force =     0.194758     Total SCF correction =     0.000003
```

Forces are given in units of Ry/bohr. Record this number, and re-test the convergence issues with respect to cutoff.

Problem 3

Problem 3 will test the convergence of the energy with increasing the density of the **k** point grid. To do that, you should set the converged cutoff energy found in Problem 1. In order to increase the size of the **k** point grid, edit the script and change the values in `LISTK`, putting `'6 8'` instead of `'4'`. Submit the calculation using your script, and record the total energies for the two grids $6 \times 6 \times 6$ and $8 \times 8 \times 8$ (or even denser grinds, if you find it necessary). Take a note about the number of the irreducible **k** points in every grid. It looks like this:

```
 number of k points=    8
```

Problem 4

You will have to perform a test for convergence of forces with respect to the number of **k** points. Like in Problem 3, increase the density of the **k** point grid, edit the script and change the values in `LISTK`, putting `'6 8'` (or even denser if needed) instead of `'4'` and record the forces and test the convergence.

Problems 5 and 6

Given the information you learned above, you should be able to do problems 5, 6 and 7. For problem 7, put only one value in both `LISTK` and `LISTECUT` (the converged values), and a range of values in `LISTA`. For problem 8, use the information from the first lab for elastic constants.

Problem 7

You will have to calculate the total energy as a function of the cell volume. By using the appropriate derivatives you can also calculate the pressure for each volume and bulk modulus of the material at hand. Therefore, at the end of this problem, you will have a relationship between some important state variables such as volume and pressure, i.e. which volume corresponds to which pressure, for each data point you have. Such information is

very significant in solid state physics, it helps us identify different phases of matter and phase transitions between them, as each phase of matter has a unique response to compression. These relationships are routinely used in earth sciences, high pressure and shock physics.

But ultimately, if we could recast this relationship into a mathematical function instead of data point-by-data point analysis, we would gain more predictive power about the properties of each phase. Such function that describes the relationship of state variables is called an *Equation of State*. There are several suggestions for the shape of this function. A very commonly used one is the following, what is known as the *third-order Birch-Murnaghan isothermal equation of state*:

$$P(V) = \frac{3B_0}{2}\left[\left(\frac{V_0}{V}\right)^{\frac{7}{3}} - \left(\frac{V_0}{V}\right)^{\frac{5}{3}}\right]\left\{1 + \frac{3}{4}(B_0' - 4)\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]\right\}, \tag{6}$$

where P is the pressure, V_0 is the equilibrium volume, V is the deformed volume, B_0 is the bulk modulus, B_0' is the derivative of the bulk modulus with respect to pressure. Integration of this pressure expression with volume gives us the energy versus volume relationship as below:

$$E(V) = E_0 + \frac{9V_0B_0}{16}\left\{\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^3 B_0' + \left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^2\left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]\right\}. \tag{7}$$

Now that we have a mathematical description of the relationship between state variables, we can make use of it in our calculations. We can make several calculations at different cell volumes and obtain energies. By fitting the above equation to the data we have, we can obtain the values for equilibrium volume, bulk modulus and bulk modulus derivative. Thus, with these few parameters we computed, we can reveal the energy vs. volume or pressure vs. volume relationship for a wide range of volume!

You might assume that such approach is of little use in our age of very powerful computers, as one could obtain the E(V) or P(V) relationship of each material by simply running the calculation at more points and calculating the stress tensor, therefore pressure from first-principles. However, we need to come back to the issue of numerical convergence again: stress tensor often requires many plane waves to converge, for most materials it requries several times the plane wave cutoff required for energy differences. Quantum ESPRESSO CPU time scales as $E_{cut}^{(3/2)}$, therefore you can see that calculating pressure *ab initio* quickly gets too computationally demanding. That is why it is still a common method to obtain the pressure at given volume from an equation of state fit.

Quantum ESPRESSO is shipped with a tool that does the fitting we have mentioned above, given the energy vs. volume data. This little software executable is called **ev.x**. You can find it in the same directory of our main executable **pw.x**. The executable **ev.x** works interactively, it expects that you specify units, the type of Bravais lattice that you used, the type of the equation of state that you want to use for a fit, and a data file where first column is the volume or lattice parameter and second is the energy.

Problem 8

You will have to calculate the elastic constants of MgO. There are several important points to note here:

- The cell symmetry changes upon strain. So the variable in your input that defines the symmetry (`ibrav`) in the cell should change accordingly. For a list of all possible indications of `ibrav` please see the input description at:

  http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PW.html

  `ibrav` section under `system` namelist. If you have any problems with the convention please ask us.

  Note: In this exercise you will have to start from the *conventional* unit cell with 8 atoms. Examine the MgO conventional unit cell in order to specify the atomic positions.

  Let us demonstrate how to construct an orthorhombic cell by using `ibrav=8`. Assume that the unit cell we aim to construct has the following lattice vectors (in cartesian coordinates, in Bohr)

  v1=(a,0,0), v2=(0,b,0), v3=(0,0,c)

  meaning that it is **a** Bohr long in x direction, **b** Bohr long in y, and **c** Bohr long in z. We can specify this cell in the following way:

```
   ibrav = 8
celldm(1) = a
celldm(2) = b/a
celldm(3) = c/a
```

  where instead of `b/a` type of expression, you should put the result of the division, not the expression itself.

  Another example can be the monoclinic cell. Monoclinic cell has two 90° angles and one that is different from 90°. The convention that QUANTUM ESPRESSO adopts by `ibrav=12` is such that the unique, non-90° angle is γ, the one between v1 and v2, where

  v1=(a,0,0), v2=(b$\times cos(\gamma)$,b$\times sin(\gamma)$,0), v3=(0,0,c)

  where our first vector is **a** Bohr long in x direction, our second vector is **b** Bohr long but lays in $x-y$ plane, with components both on x and y with a dependence on angle γ, and our third vector is **c** Bohr long in z axis. We can specify this cell in the following way:

```
   ibrav = 12
celldm(1) = a
celldm(2) = b/a
celldm(3) = c/a
celldm(4) = cos(γ)
```

and again, replace the expressions with the numerical values, including the cosine.

- Upon the application of the strain, it may happen that not all atomic positions in the cell are fixed by symmetry, as there is more than one basis element (i.e. a magnesium and oxygen atom). Therefore, when we apply the strain one of these atoms could move to another relative position than the initial (1/4, 1/4, 1/4) and lower the total energy of the cell. Hence, to calculate the elastic constant of this system, we should allow the cell to relax the atoms to their equilibrium positions instead of fixing them to relative positions. This is done very easily with QUANTUM ESPRESSO. The necessary modification is changing the `calculation` variable in `control` namelist, from `scf` to `relax`

  ```
  calculation="relax"
  ```

  And now since we will move the ions we will need a new namelist called `ions`:

  ```
  &ions
  /
  ```

  Note that you can place this new namelist after the electrons namelist (after the slash) and since we will only use default parameters you do not need to specify anything in this namelist. Therefore we only declare the namelist with the &, and close it with a slash right away.

We suggest to you to read Ref. [3] in order to learn about the first principles calculations of the elastic constants.

6 FAQ

These are some of the questions that have appeared in relation with previous versions of this handout or the related exercises:

- **How precisely do I need to compute the lattice parameter?**

  Lattice parameters are typically listed to within 0.01 Å. There are applications when higher precision is required; this is *not* one of them.

- **Is the total energy higher when you move an atom (the force calculations) with respect to the equilibrium possition?**

  Yes. Remember: at equilibrium the energy is the lowest. Equilibrium for this structure has a Mg atom at (0, 0, 0) and O at (0.25, 0.25, 0.25). As a side note the forces will give you an idea of how far you are from equilibrium (they tell you which direction the atoms "want" to move). The stresses tell you which direction the cell parameters "want" to change to reach equilibrium.

- **My energy versus lattice constant plot is jagged.**

  There are a number of solutions to this; the easiest is to raise the energy cutoff.

- **How do I kill my job?**

  Type `top`. This will tell you which jobs are getting most of the resources of the computer. Look at the number of your process in the PID column. Type `kill` number_of_your_process to kill your job. Another command you might want to use is `ps` (process status) to see what processes you are running.

- **How is "convergence of energy" defined?**

  You say that your energy is converged to X Rydberg when $E_{\text{true}} - E_n = X$ (E_n is the current energy). How do you know E_{true}? In practice you might take your energy at the highest cutoff (or k-grid, or whatever) that you calculated — if that seems converged, you might call that E_{true}. The most important thing is that you do *not* define convergence as $E_{n+1} - E_n$, where n is a step in energy cutoff (or k-grid, or whatever).

  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 am having problems both converting Ry to eV and converting bohr to Angstrom**

  This units page may help you:

  http://www.chemie.fu-berlin.de/chemistry/general/units_en.html

- **Does PWscf use LDA or GGA? DFT or Hartree-Fock?**

  PWscf uses DFT. But if you specify `input_dft = 'HF'`, it will use the Hartree-Fock. It has LDA, GGA and other more complicated functionals. You should look inside of the pseudopotential file in order to understand, which exchange-correlation functional do you use. In this lab, you use LDA, because `C.vbc.UPF` contains "PZ Exchange-Correlation functional", where "PZ" means Perdew-Zunger parametrization of the LDA functional.

- **My energies are really different from lab 1. What is the "correct" scale I should be looking for?**

  Remember that absolute energies do not have any physical meaning. Only the energy differences (other way speaking, the relative positions of the energy levels) are meaninful. Also, the reference energies are different. In lab 1, the reference energy (when atoms are infinitely far apart) is 0. In lab 2, this is not the case. The absolute energies can shift a lot, depending on which reference energies you take.

- **Why do I take k point grids with the same number of points per direction? Can I take other k point grids?**

  For this material, we take the same number of **k** points per direction, because the three lattice directions are equivalent. This is not always true. The most common **k** point

meshes are those that sample the reciprocal space evenly in all directions. Thus, for a tetragonal cell (with $a = b$, $c = 2a$, and all angles 90 degrees) we might take a $8 \times 8 \times 4$ mesh. Try to think why this is so. (Note that if a lattice vector is longer in real space, it is shorter in the reciprocal space).

References

[1] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. De Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. Seitsonen, A. Smogunov, P. Umari, and R. Wentzcovitch. Quantum espresso: A modular and open-source software project for quantum simulations of materials. *J. Phys.: Condens. Matter*, 21:395502, 2009.

[2] R. Martin, editor. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, Cambridge, 2004.

[3] M. J. Mehl, B. M. Klein, and D. A. Papaconstantopoulos, editors. *First principles calculations of elastic properties of metals*. John Wiley and Sons, London, 1994.

[4] H. Monkhorst and J. Pack. Special points for Brillouin zone integrations. *Phys. Rev. B*, 13:5188, 1976.

[5] D. Sholl and J. Steckel. *Density Functional Theory: A Practical Introduction*. Wiley, New Jersey, 2009.