DFT exercises for Quantum Mobile
First-principles calculations of the electronic properties of materials: The case study of bulk sodium chloride

Davide Grassano, Francisco Ramirez, Norma Rivano, Luca Binci,*
Nicola Marzari

École Polytechnique Fédérale de Lausanne (THEOS)

April 2020

Contents

1 Introduction

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.

You can find an extended list of codes here.

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 several 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 explain how to compute the total energy of a bulk sodium chloride (NaCl) using the PWscf program. You can also find many other tutorials for QUANTUM ESPRESSO in the official webpage of the code and in the corresponding section of the Materials Cloud website: especially recommended is the hands-on tutorial of Santa Barbara 2009. You can also find here a class on atomistic computer modeling of materials (on which this homework is also based) that was taught for 10 years at MIT by Gerd Ceder and Nicola Marzari.

1.1 Setting up the work environment

Computer programs are not always so easy to install and maintain, specially those designed by scientific researchers. One of the easiest ways to deal with this issue in the context of learning the use of a new tool, so that the person learning doesn't need to first start troubleshooting configuration problems before actually testing the tool itself, is to provide the necessary codes pre-installed inside of a self-contained working environment called "Virtual Machine". This virtual machines can be run using different programs and effectively work as a "machine inside the machine", with its own libraries, programs and files and with limited points of contact with the "external device" that is being used to run it (your physical computer).

To set this up you will first need to download and install VirtualBox version 6.1.4 or later: this is the program that we will use to run the virtual machine. Then you also need to download the virtual machine itself, Quantum Mobile version 20.03.1, and import it into Virtualbox (this will occupy 13G of disk space) by going to the "File" menu and choosing "Import Appliance" (use the username "max" and password "moritz", both without the inverted commas). See Figure 1 for an idea on how VirtualBox looks like and where to find the "Import Appliance" option; you can also consult the FAQ of Quantum Mobile for further information or troubleshooting.

Now, since the virtual machine does not see the files of the external device, in order to move files into and out of it we will need to create a "shared folder": a folder in the external machine that will appear in the internal machine as if it was an external device (like a pendrive). To do this, first create a folder on your Desktop called "VM", then open VirtualBox and select the Quantum Mobile virtual machine on the left menu and click "Settings" (make sure you are not currently running the virtual machine). Click on to the "Shared Folders" section and click on the button to add a new shared folder (it has a folder and a plus sign in it) to arrive at the menu showed in Figure 2. Select the field "Folder Path", go to the recently created folder "VM" and click on "Open": the "Folder Name" field should now say "VM" (if it doesn't, write it now), and you should check the "Auto-mount" checkbox (if you have one). Finally, in the "Mount Point" field you should write "/home/max/Desktop/SHARED" and then click the "Ok" button. The next time you log into your virtual machine, you should see the shared folder in your desktop and be able to both read the files inside of it and add new files to it.
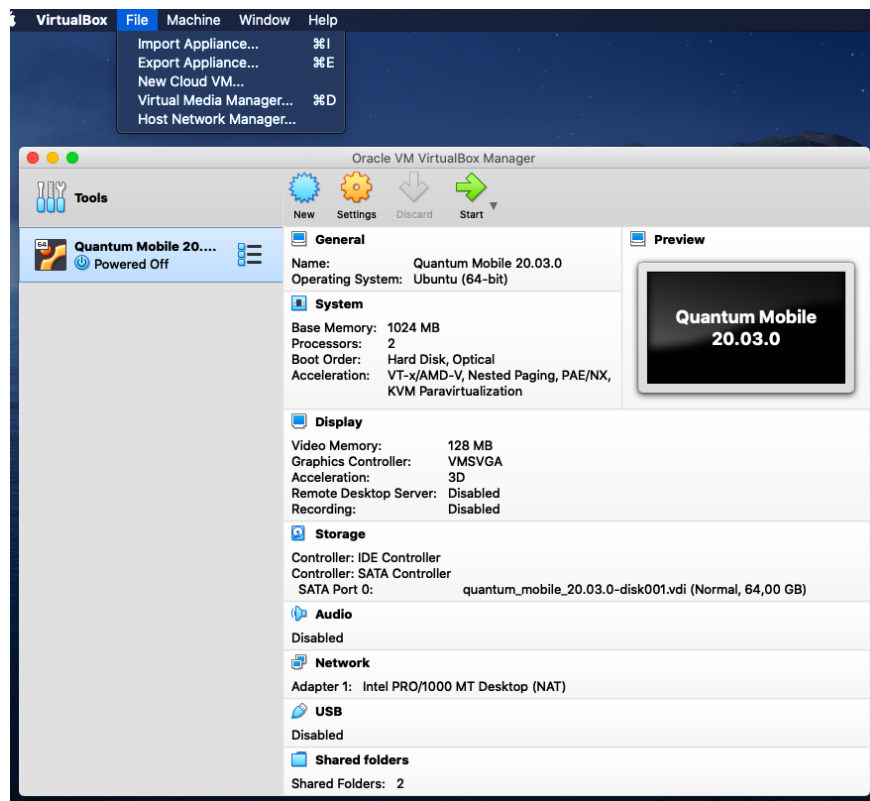
Figure 1: VirtualBox interface and "import" option (in MacOS)
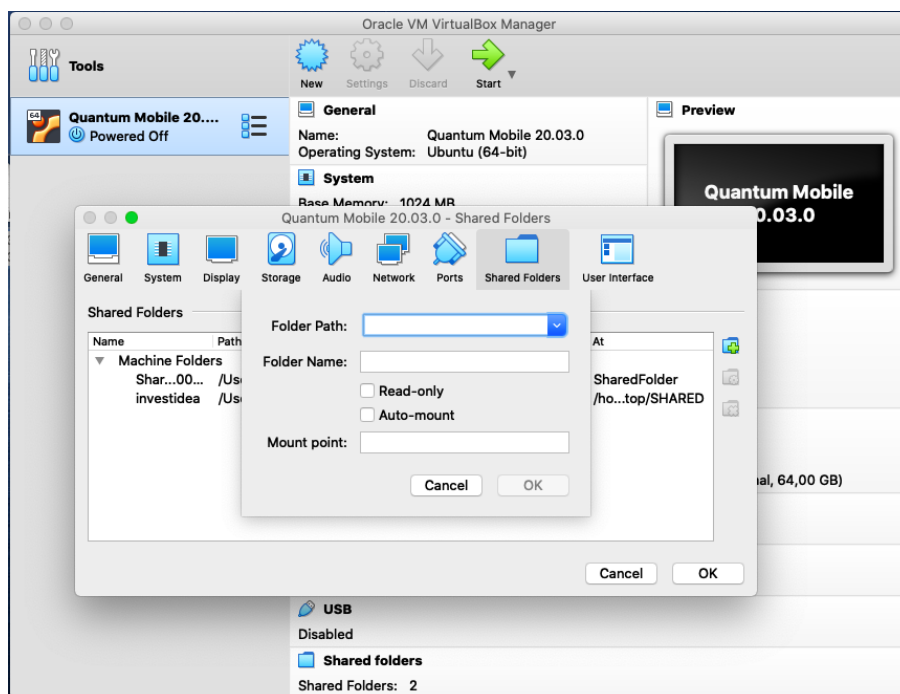

Figure 2: Creating a shared folder

The last configuration you might want to change is the keyboard (in the Virtual Machine the keyboard is set to the US one). For this you need to go to "Settings" and choose the language and layout that is appropriate for your own hardware.

As a first step, if you haven't already, you need to download the material by going to the [github webpagepage of this tutorial](#) and clicking on the green "Clone or download" button and then "Download ZIP". Unzip the learn-fireside-master.zip file and move the generated folder inside of the shared folder "VM" (if you already had the material, you just need to follow this last step to make it accessible to the virtual machine).

You can now go into the virtual machine and open a terminal: we will go to the folder we set up and start working there. We can use the file structure that is already in place to organize our tests and exercises.

```
max@qmobile:~$ cd Desktop/SHARED/learn-fireside-master
max@qmobile:~/Desktop/SHARED/learn-fireside-master$ ls

     (list of folders and files)
```

Note: From now on we will start hiding the full path of the current directory in the commands indications (so we will just use "`max@qmobile:$`" instead of writing the full paths like "`max@qmobile:~/Desktop/SHARED/learn-fireside-master/(...)$`").

1.2 Input file for the PWscf program

First of all we are going to make a basic test to see how PWscf works. To do this, we will first go to the designated directory and then copy the necessary files there: the input file for PWscf and folder with the pseudo-potentials.

```
max@qmobile:$ cd 0_initial_tests
max@qmobile:$ cp ../files/NaCl.scf.in .
max@qmobile:$ cp -r ../files/pseudo .
```

File NaCl.scf.in contains the input information needed to compute the total energy per unit cell of sodium chloride using PWscf, and the directory /pseudo contains the pseudopotential files for sodium (na_pbe_v1.5.uspp.F.UPF) and chlorine (cl_pbe_v1.4.uspp.F.UPF)

If you view the input file, it will look like this:

```
max@qmobile:$ less NaCl.scf.in

&control
    calculation = 'scf'
    restart_mode = 'from_scratch'
    prefix = 'NaCl'
    tstress = .true.
    tprnfor = .true.
```

```
    pseudo_dir = './pseudo/'
    outdir = './tmp/'
/
&system
    ibrav = 2
    celldm(1) = 10.5
    nat = 2
    ntyp = 2
    ecutwfc = 20.0
    ecutrho = 160.0
/
&electrons
    diagonalization = 'david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-8
/
ATOMIC_SPECIES
    Na  22.990    na_pbe_v1.5.uspp.F.UPF
    Cl  35.446    cl_pbe_v1.4.uspp.F.UPF
ATOMIC_POSITIONS {alat}
    Na 0.00 0.00 0.00
    Cl 0.50 0.00 0.00
K_POINTS {automatic}
    2 2 2 0 0 0
```

You can find a more detailed description for all the input parameters of pw.x in the official PW documentation of QE.

The most relevant lines for the calculations related to this tutorial 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 = 'NaCl'

  is a tag the code uses to identify this calculation.

- The line

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

indicates the path to the directory to be created by the code, 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 (`na_pbe_v1.5.uspp.F.UPF` and `cl_pbe_v1.4.uspp.F.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) = 10.5
```

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 = 20.0
```

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
Na 22.990 na_pbe_v1.5.uspp.F.UPF
Cl 35.446 cl_pbe_v1.4.uspp.F.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 details 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 NaCl) 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}
 Na 0.00 0.00 0.00
 Cl 0.50 0.00 0.00
```

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 $4 \times 4 \times 4$ **k** point sampling will be used using the Monkhorst-Pack procedure [2]. The last three numbers indicate that no shift with respect to the center of the Brillouin zone will be made.

These are only some of the possible input settings: the program has many more, for which it usually uses default values when the user doesn't specify them, and the quality of the results obtained will highly depend on the set of parameters used. We will see some examples in this tutorial of how to calibrate the number of K_POINTS and ecutwfc, or how to obtain the relaxed cell dimensions and how to obtain properties of the material by modifying them, but for the rest of the keywords we have already provided a set of values that we know work well for this system and won't be modified (for example, the whole section for electrons).

When doing your own research, it might be difficult to know which set of parameters you need to set up, at least as a starting point, to get some decent results. For this, the Materials Cloud website offers a tool called "Quantum ESPRESSO input generator and structure visualizer" that will both allow you to check the integrity of the structure you are using and will also make a "first guess" at a good set of keyword values and pseudopotentials that you can use to study the system. We will now see how you can also check the structure locally with a different program before you start using it to simulate the material.

1.3 Visualizing the crystal

Before you start your first calculation, let us introduce a very useful visualization tool called XCrySDen. 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:

<p align="center">xcrysden −−pwi NaCl.scf.in</p>

which launches the XCrySDen program, indicating that the file that contains the structure to visualize is named "NaCl.scf.in" and has the format of a PWscf input (−−pwi). 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.
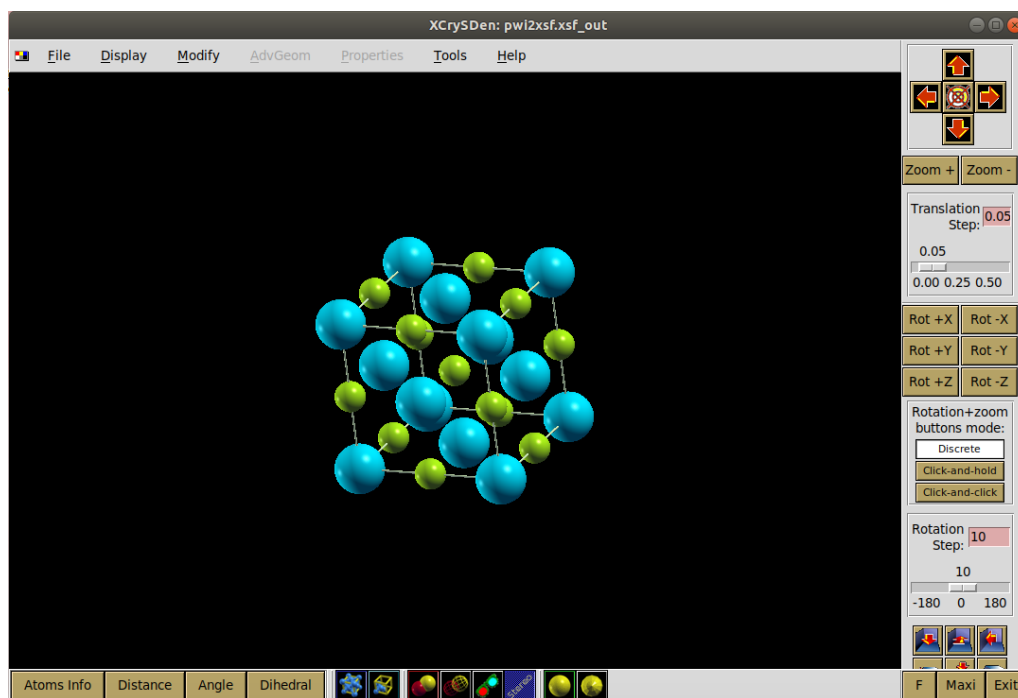
Figure 3: Using XCrySDen to view the NaCl crystal.

1.4 Running a PWscf example and obtaining the total energy

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

pw.x < NaCl.scf.in > NaCl.scf.out

(we are telling the shell to launch the executable pw.x from a location within the `PATH` env variable, pipe the file `NaCl.scf.in` to its `stdin` and pipe its `stdout` to the file `NaCl.scf.out`). After a few seconds (if everything goes well) you should have in your directory a new file called `NaCl.scf.out`. This is the output file of your calculation.

If you view the file `NaCl.scf.out` (you can do su by running `less NaCl.scf.out`, or `vi NaCl.scf.out`, or any other text editor you are comfortable with), 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. Close to the end, a line containing something like:

```
!    total energy              =     -124.49276578 Ry
```

will appear. This is the final result from the program for the total energy of the unit cell. Note that the final occurrence of "total energy" will have an exclamation mark(!) next to it, as something to make 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 NaCl.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     :      0.97s CPU      0.98s WALL (       1 calls)
electrons    :     13.16s CPU     13.21s WALL (       1 calls)
forces       :      0.28s CPU      0.29s WALL (       1 calls)
stress       :      0.76s CPU      0.78s WALL (       1 calls)
.
.
.
PWSCF        :     15.40s CPU     15.50s WALL
```

As you practice with the example exercises, you will get more familiar with the output format and all the information written.

1.5 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. [3]. Another excellent book is [4].

1.5.1 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 $\mathbf{G}$. 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 $\mathbf{G}$ 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:

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

Exercise 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: in the end, you will have to find the right compromise between accuracy and efficiency that is needed for your particular system.

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

1.5.2 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 $\mathbf{k}$ vectors ($\int d\mathbf{k}(...)$). In practice, the integrals over $\mathbf{k}$ 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 (`NaCl.scf.out`) you will find something like this:

```
number of k points=    16
                 cart. coord. in units 2pi/alat
   k(    1) = (    0.0000000    0.0000000    0.0000000), wk =    0.0092593
   k(    2) = (   -0.1666667    0.1666667   -0.1666667), wk =    0.0740741
   k(    3) = (   -0.3333333    0.3333333   -0.3333333), wk =    0.0740741
   k(    4) = (    0.5000000   -0.5000000    0.5000000), wk =    0.0370370
   k(    5) = (    0.0000000    0.3333333    0.0000000), wk =    0.0555556
   k(    6) = (   -0.1666667    0.5000000   -0.1666667), wk =    0.2222222
   k(    7) = (    0.6666667   -0.3333333    0.6666667), wk =    0.2222222
   k(    8) = (    0.5000000   -0.1666667    0.5000000), wk =    0.2222222
   k(    9) = (    0.3333333    0.0000000    0.3333333), wk =    0.1111111
   k(   10) = (    0.0000000    0.6666667    0.0000000), wk =    0.0555556
   k(   11) = (    0.8333333   -0.1666667    0.8333333), wk =    0.2222222
   k(   12) = (    0.6666667   -0.0000000    0.6666667), wk =    0.1111111
   k(   13) = (    0.0000000   -1.0000000    0.0000000), wk =    0.0277778
   k(   14) = (    0.6666667   -0.3333333    1.0000000), wk =    0.2222222
   k(   15) = (    0.5000000   -0.1666667    0.8333333), wk =    0.2222222
   k(   16) = (   -0.3333333   -1.0000000    0.0000000), wk =    0.1111111
```

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 16 **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 ($6 \times 6 \times 6$, $8 \times 8 \times 8$, 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!**

1.6   Using bash scripts to run PWscf multiple times

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 `learn-fireside-master/files/script.sh`. The script looks like this:

```bash
#!/usr/bin/env bash

# Input data:
LISTA="10.5"        # List of values of lattice parameter to try
LISTECUT="20 30"  # 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

SAVE="SAVE.txt"

echo "#   a(bohr)  ecut (Ry)  num_kpt     E_tot (Ry)    F_x Na    F_y Na    F_z Na
# 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

        INPUT="NaCl.scf.a=$a.ecut=$ecut.k=$k.in"
        OUTPUT="NaCl.scf.a=$a.ecut=$ecut.k=$k.out"
        echo 'Doing ecut, k, a: ' $ecut $k $a
        # Create new input file:
```

```
cat > $OUT_DIR/$INPUT << EOF
        &control
            calculation = 'scf'
            restart_mode = 'from_scratch'
            prefix = 'NaCl.$a.$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
            Na  22.990     Na.pbe.oncvpsp.UPF
            Cl  35.446     Cl.pbe.oncvpsp.UPF
        ATOMIC_POSITIONS {alat}
            Na 0.00 0.00 0.00
            Cl 0.50 0.00 0.00
        K_POINTS {automatic}
            $k $k $k  0 0 0
EOF

        # Run PWscf to create new output file:
        $PW_LAUNCH < $OUT_DIR/$INPUT > $OUT_DIR/$OUTPUT

        # Uncomment this lines by removing the leadings # to automatically store rel
        # E_TOT=$(cat $OUT_DIR/$OUTPUT | grep ! | tr -dc '[0-9]\.\-\n')
        # NKPT=$(cat $OUT_DIR/$OUTPUT | grep "number of k points=" |
          tr -dc '[0-9]\.\-\n')
        # F1_x=$(cat $OUT_DIR/$OUTPUT | grep " atom    1 type  1   force" |
          cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f2)
        # F1_y=$(cat $OUT_DIR/$OUTPUT | grep " atom    1 type  1   force" |
          cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f3)
        # F1_z=$(cat $OUT_DIR/$OUTPUT | grep " atom    1 type  1   force" |
```

```
                       cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f4)
          # F2_x=$(cat $OUT_DIR/$OUTPUT | grep " atom    2 type  2   force" |
             cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f2)
          # F2_y=$(cat $OUT_DIR/$OUTPUT | grep " atom    2 type  2   force" |
             cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f3)
          # F2_z=$(cat $OUT_DIR/$OUTPUT | grep " atom    2 type  2   force" |
             cut -d= -f2 | tr -dc '[0-9]\.\-\n ' | tr -s ' ' | cut -d' ' -f4)

          # echo "`printf "%11.4f %10.1f %8d %15.6f %9.6f %9.6f %9.6f %9.6f
             %9.6f %9.6f" $a $ecut $NKPT $E_TOT $F1_z $F1_y $F1_z
             $F2_x $F2_y $F2_z`" >> $SAVE

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

  rm -r $TMP_DIR/*

  echo -e "\n" >> $SAVE
```

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

```
    LISTA="10.5"
    LISTECUT="20 30"
    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:

$$\text{nohup ./script.sh \&}$$

(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 several output files in your output directory. Go in the output directory and type `ls -l` to get a list of the produced files.
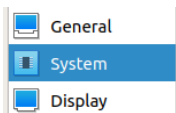
In order to solve the exercises without having to manually create and run all the inputs, we suggest that you try to modify the above script accordingly to do adapt it to the different calculations you need.

## 1.7   Running PWscf on the VM with many processors

For most of the exercises that you will have to do, executing `pw.x` with only one core will be enough to finish them in a matter of a few minutes. Nevertheless, the last exercises will be more demanding than the others, as it requires to run relax calculations on a supercell. Because of this, it is advisable to increase the number of CPUs assigned to the VM (if you have a fast computer). In order to do so you have to:

- Shutdown the VM (full shut-down).

- Select the VM in the VBox interface

- Click on 'Settings'

- Click on 'System' in the left sidebar

- Click on 'Processor' in the tab bar and change the number of CPUs (I would recommend half of the maximum value or before you reach the red part of the slider bar). If you can't change this it means the VM was not totally turned off.
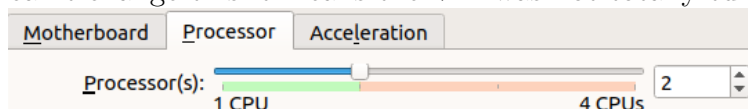
- Press 'Ok' to save the changes

In order to make use of having multiple processors you will need to launch the calculation using 'mpirun':

- From the command line:

$$\text{mpirun pw.x} < \text{input\_name} > \text{output\_name}$$

- From a script: locate the line that sets the executable

$$\text{PW\_LAUNCH='pw.x'}$$

And change it to:

$$\text{PW\_LAUNCH='mpirun pw.x'}$$

Depending on the system, the command 'mpirun' cold be different (e.g. 'srun' on the FIDIS supercomputer).

With a small amount of CPU, the improvement you should experience in run time should be linear (eg. twice the CPUs $\hat{=}$ half the time). Be aware that this is true for long calculations, as the serial part of the code execution should be negligible compared to the parallel one. For shorter calculations the scaling will be different.

2 Exercise examples using QUANTUM ESPRESSO

In order to solve the following exercises, you will have to run several PWscf calculations. We suggest that you use the folder structure that is already in place in the file you downloaded from the github page. Ultimately, you can organize directories for saving the output runs in any way you like, but you might need to do some modifications in the bash scripts provided.

2.1 Exercise 1: Convergence tests for input parameters

A) Convergence of *total (absolute) energies* **with respect to cutoff energies**

In exercise 1.1 we will check the convergence of the total energy with respect to the kinetic-energy cutoff: for this you need to run scf calculations of bulk NaCl for different values of this parameter and obtain the total energy. A good range to try is 10-150 Ry, doing calculations at increments of 10 Ry: in order to do this using the `script.sh`, modify the content of `LISTECUT` so that different numbers are included. When changing the cutoff, make sure to keep the other variables (lattice constant, **k** points mesh, etc.) fixed and to record them: this means that 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.

Record and plot your final results. Check that you reach the level of convergence (difference of energy between a given point and the estimated value of the converged energy) of around 5 meV/atom when you get to the 80 Ry. Note that PWscf calculates energy per unit cell (not per atom) in atomic units (so Rydberg instead of eV).

Note: You should see that as you increase the value of the cutoff energy, and thus include more basis functions, the energy decreases in a monotonical way. This is because the increase in basis functions available can only improve the accuracy of the candidate for the wavefunction of the system (worst case scenario, you just end up using the combination of basis that when you had a lower cutoff) and therefore the energy can only be equal or lower.

B) Convergence of *forces* **with respect to cutoff energies**

In some cases, we are interested in quantities other than energies: in this case, we will be calculating forces acting on atoms. In the previous exercise, the forces on Na and Cl were zero in the x, y, and z directions 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.) Therefore now we will want to create forces by displacing a Cl (or Na) atom by +0.05 (fractional coordinates) in the z direction. To do this, we need to edit the z coordinate of the Cl (or Na) ion in the input file. After you edit the script, the corresponding lines should look like this:

```
ATOMIC_POSITIONS {alat}
Na  0.00 0.00 0.00
Cl  0.50 0.00 0.05
```

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

```
Forces acting on atoms (cartesian axes, Ry/au):

atom    1 type  1   force =     0.00000000    0.00000000    0.02078255
atom    2 type  2   force =    -0.00000000   -0.00000000   -0.02078255

Total force =      0.029391    Total SCF correction =      0.000024
```

(You can also notice that by displacing an atom we are lowering the symmetry of the system, hence more irreducible kpoints will be present with respect with the previous calculations.)

Now re-test the convergence issues with respect to cutoff while keeping other parameters fixed. Reach the convergence on forces to within around 10 meV/A (again, notice that PWscf gives forces in Ry/Bohr). Record relevant parameters. Use the **k** points mesh fixed to $4 \times 4 \times 4$ centered at the Brillouin zone. Plot and record your results.

C) Convergence of the *total (absolute) energies* with respect to the size of the k points mesh

In this exercise you will test the convergence of the energy with increasing the density of the **k** point grid and registering the total energy. For each mesh, record the number of the **k** points in the irreducible wedge of the first Brillouin zone (not the size of the grid): this gives a measure of how long your calculation will take, as calculations scale linearly with the number of **k** points. When changing the size of the **k** points mesh, make sure to keep all other input parameters fixed (lattice constant, energy cutoff, etc.): in particular, you need to use the converged cutoff energy found in exercise 1.

In order to increase the size of the **k** point grid, edit the script and change the values in LISTK, putting '2 4 6 8' instead of '4'. Submit the calculation using your script, and record the total energies for the two grids $2 \times 2 \times 2$, $4 \times 4 \times 4$, $6 \times 6 \times 6$ and $8 \times 8 \times 8$ (or even denser grinds, if you find it necessary). To get the number of irreducible **k** points, you can search in the output file for the following:

```
number of k points=    16
```

Note: Contrary to what happen with the cutoff energy, you may notice that now the energy is not a monotonically decreasing function of the **k** point grid. This is because changing the **k** point grid does not affect the wavefunction directly, but through increasing the precision with which the Hamiltonian is calculated. Since the variational principle applies for a given fixed Hamiltonian, we don't have here the guarantee that the energy has to decrease.

D) Convergence of *forces* **with respect to the size of the k points mesh**

Now you will have to perform a test for convergence of forces with respect to the number of **k** points. Like in exercise 2, in order to calculate the force acting on a Na (or Cl) atom, you will have to displace it by +0.05 in the z direction (in fractional coordinates). But this time, like in exercise 3, you will increase the density of the **k** point grid by modifying the script and change the values in `LISTK`, putting `'6 8'` (or even denser if needed) instead of `'4'` and recording the forces. Remember to keep all other parameters fixed and record all of the relevant ones (lattice parameter, energy cutoff, etc.). Reach the convergence on forces to within around 10 meV/A

E) Convergence of the *total energy differences* **with respect to energy cutoff**

In practice only energy differences have physical meaning, as opposed to absolute energy scales, which can be arbitrarily shifted. Therefore, the final convergence test that we will see here is the one for the total energy difference between two crystals at different lattice parameters, as a function of cutoff. For example, you could calculate the energy of NaCl at the experimental lattice distance of 5.640 Angstrom, and then calculate the energy using another value close to it (5.644 Angstrom, for example), take the difference between the two energies, and repeat this process for many energy cutoffs. As always, make sure to keep all other input parameters (lattice constant, **k** points mesh, etc.) fixed while changing the energy cutoff by putting only one value in the lists of the script that are not relevant for this test. Record all relevant parameters such as the lattice constant, **k** points mesh, etc. Keep increasing the energy cutoff until you reach the convergence to around 5 meV/atom.

2.2 Exercise 2: Determination of the *equilibrium lattice length* **and** *bulk modulus*

We will now use the program to make a series of calculations that will allow us to calculate the properties mentioned for the material we have been testing. For this, you will use the energy cutoff and **k** points mesh which you determined for the force and energy difference calculations. Usually, one is interested in quantities such as forces and energy differences and that is why we made the convergence test with those quantities. But more in general, to be absolutely safe, one should test the convergence of the quantity one is interested in (lattice parameter and bulk modulus in this case) with respect to the energy cutoff and **k** points mesh (and other parameters, which we don't have for our system (e.g. smearing)). We are not going to do it like that in this tutorial but it is something you should take into account when considering your own research.

   This exercise is divided in the following steps:

A) Calculate the equilibrium lattice parameter of NaCl and compare this theoretical (computed) value a_0^{theor} with the experimental equilibrium lattice parameter $a_0^{exp} = 5.640$ (A) one. Make sure to record all the relevant input parameters of the calculations (energy cutoff, **k** points mesh, etc.).

B) Calculate the bulk modulus B of NaCl. The bulk modulus is a measure of the stiffness of a material and it is defined as

$$B = -V_0 \frac{\partial P}{\partial V},$$

where P is the pressure on the material, V is its volume, and V_0 is its equilibrium volume. Knowing that $P = -\partial E/\partial V$, you need first to derive from the previous equations the direct relationship between the magnitude to compute and the results obtainable through the calculations: energies per unit cell for different cell sizes/volumes.

C) Calculate the bulk modulus B of NaCl using the third-order Birch-Murnaghan isothermal equation of state. We suggest to use the provided interactive **ev.x** program provided with the QUANTUM ESPRESSO, which contains the implementation of this equation. In the input file for ev.x you have to provide two columns for the case of an ffc lattice: the first one contains the lattice parameter and the second one the total energy obtained. The fitting function is labeled as "birch2".

The first item of this exercise requires you to calculate the total energy as a function of the cell volume, so at the end of it 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.

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. In first approximation we could approximate the energy vs the volume to the second order of Taylor expansion and use the relation $P = -\partial E/\partial V$ to calculate $P(V)$. A better approach would be to use a more refined *Equation of State* to describe the relationship between these variables, as proposed in items b and c. There are several suggestions for the shape of this function, but the chosen one for this example is 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:

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

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 requires 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 ('Ang' or 'ANG' or 'ang' indicates Angstroms, while any other input will default to atomic 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 the first column is the lattice parameter (or volume in case of 'noncubic'≡'hex') and second is the energy in Ry.

2.3 Exercise 3: Determination of the *elastic constants*

This is the second and last application to material properties that we will show in this tutorial. For this, it will be necessary to make alterations in the cell, resulting from the application of certain "strains" that respect certain symmetries, and for which it will be convenient to use the conventional unit cell containing 8 atoms and the orthorombic (`ibrav=8`) and monoclinic (`ibrav=12`) Bravais lattices (For a list of all possible indications of `ibrav` please click here). We suggest to you to read Ref. [5] in order to learn about the first principles calculations of the elastic constants.

For a demonstration on how to construct an orthorhombic cell, let us assume that the unit cell we aim to construct has the following lattice vectors (in cartesian coordinates, and 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.

On the other hand, the 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.

IMPORTANT: although we are using different lattices and cell, you can use the converged parameters that you estimated in exercise 1. In principle, you should need less k points with the conventional supercell to get to convergence; in your own research you should repeat the convergence tests for these new lattices and cell.

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 sodium and chlorine atom). Therefore, when we apply the strain one of these atoms could move to another relative position than the initial $(1/2, 0, 0)$ 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, which can be 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.

You will compute the energetics of deformation, and fit the resulting energy curves in two different ways [A] and [B]. You can then compare your results with the experimental data.

A) Compute $C_{11} - C_{12}$ using the orthorombic (`ibrav=8`) Bravais lattice, and using the relation $B = \frac{1}{3}(C_{11}+2C_{12})$ determine the elastic constants C_{11} and C_{12}. Then compute C_{44} using the monoclinic (`ibrav=12`) Bravais lattice.

B) Compute $C_{11} - C_{12}$ using `ibrav=0` and specifying the card `CELL_PARAMETERS`, and then use the relation for B (as above) to determine the elastic constants C_{11} and C_{12}. Compute C_{44} using `ibrav=0` too.

3 FAQ

- **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 position?**

  Yes. Remember: at equilibrium the energy is the lowest. Equilibrium for this structure has a Na atom at (0, 0, 0) and Cl at (0.50, 0.0, 0.0). 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 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 don't understand convergence of energy and forces. It seems that, as a percentage of the absolute value, energies converge much faster.**

  Sometimes you are interested in an absolute value, rather than a percentage value. For instance, let's say you can measure the length to within 1 mm. If you measure the length of an ant, in terms of percentage error, you may be off by 50% or more. If you measure the length of an elephant, in terms of absolute error, you may be off by 0.01%. But usually you don't care as long as you are to within 1 mm. Errors on forces are the same. Don't worry about the percentage errors so much. You could always arbitrarily decrease the percentage errors on the forces by taking a bigger displacement. From experience, we know that a good error on energy differences is around 5 meV/atom. From experience, we also know that a good error on forces is 10 meV/Å. These are just values that we know, because we have done many first-principles calculations in the past.

- **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 examples, we are using PBE, as highlighted in the name of the pseudopotential file.

- **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.

References

[1] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, *et al.*, "Quantum espresso: a modular and open-source software project for quantum simulations of materials," *Journal of physics: Condensed matter*, vol. 21, no. 39, p. 395502, 2009.

[2] H. J. Monkhorst and J. D. Pack, "Special points for brillouin-zone integrations," *Physical review B*, vol. 13, no. 12, p. 5188, 1976.

[3] D. Sholl and J. A. Steckel, *Density functional theory: a practical introduction*. John Wiley & Sons, 2011.

[4] R. M. Martin and R. M. Martin, *Electronic structure: basic theory and practical methods*. Cambridge university press, 2004.

[5] M. J. Mehl, B. M. Klein, and D. A. Papaconstantopoulos, "First-principles calculation of elastic properties," *Intermetallic Compounds*, vol. 1, pp. 195–210, 1994.