Hands-on Exercises on DFT using Quantum Espresso, 30 Oct. 2025

Hands-on based on QE-v7.5

In this session we will learn how to use `pw.x` and other modules of Quantum Espresso `QE` for studying electronic structure properties of materials using density functional theory (DFT). The tutorial is very basic and it is designed for those with little or no experience in `QE`. If you are using Cyclone login via:

```
ssh -X username@cyclone.hpcf.cyi.ac.cy
```

and you are advised to prepare the following script file, e.g. `script.sh`:

```
#!/bin/bash -l
#
#SBATCH --job-name=DFTtute # Job name
#SBATCH --partition=cpu # Partition
#SBATCH --nodes=1 # Number of nodes
#SBATCH --ntasks-per-node=40 # Number of cores
#SBATCH --output=job.out # Stdout (%j=jobId)
#SBATCH --error=job.err # Stderr (%j=jobId)
#SBATCH -A edu30 # Tutorial code

# Load any necessary modules
module load intel/2023a
module load imkl/2023.1.0
module load impi/2021.9.0-intel-compilers-2023.1.0
QE=/nvme/h/mzacharias/scratch/q-e/bin # path to QE executables
cd $PWD
```

For each calculation, pass the command at the bottom of the script and submit the job, e.g. `sbatch script.sh`. Also set the following environment variable in your shell:

```
$ export QE=/nvme/h/mzacharias/scratch/q-e/bin
```

For the tutorial, the codes (both main ZG and local) are already compiled. If you take the tutorial separately, routines in PW, PH, and PP directories are required to run these exercises (e.g. `make pw ph pp`).

The description of all input flags for `pw.x` is available in the following the link:
https://www.quantum-espresso.org/Doc/INPUT_PW.html

▶ Copy the tutorial tarball from `/nvme/h/mzacharias/scratch` and untar. Then, go to `exercise1`, and create directory `workdir`:

```
$ cd ~/scratch
$ cp /nvme/h/mzacharias/scratch/tutorial_DFT.tar .
$ tar -xvf tutorial_DFT.tar; cd tutorial_DFT/exercise1; mkdir workdir
```

Note: In this tutorial we show how to obtain all input and output files. The directories `inputs` and `outputs` are used as a reference or to speed up the process.

Exercise 1

In this exercise we will perform an SCF calculation for Silicon and perform convergence tests for the total energy.

▶ Run a self-consistent calculation for silicon in the `workdir`. The input file `si.scf.in` is:

```
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.20,
    nat =  2,
    ntyp = 1,
    ecutwfc = 10.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
3 3 3 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

Note: PBEsol ONCV pseudopotential files downloaded from PseudoDojo

```
$  cd workdir; cp ../inputs/si.scf.in .
$  mpirun -np 4 $QE/pw.x -nk 4 < si.scf.in > si.scf.out
```

To find the total energy and individual contributions type:

```
$ grep -10 ! si.scf.out | tail -11
```

The output is:

```
     total energy              =     -16.80271356 Ry
     estimated scf accuracy    <       0.00000002 Ry

     The total energy is the sum of the following terms:
     one-electron contribution =       5.19148467 Ry
     hartree contribution      =       1.08744726 Ry
     xc contribution           =      -6.18188692 Ry
```

```
     ewald contribution        =     -16.89975858 Ry

     convergence has been achieved in   5 iterations
```

Here we see, in the first line, the total DFT energy of the system and its decomposition into various contributions as seen in the lectures. The Ewald contribution is the ion-ion (or nucleus-nucleus) repulsion term, representing the electrostatic interaction between the charged ion cores in the crystal lattice, while the one-electron and Hartree contributions are the electron-ion (+ electrons K.E.) and electron-electron terms, respectively. The total energy is the sum of these an the exchange-correlation (xc) contribution. However, we note that here we use a small cutoff energy of 10 Ry which is not enough for converging the total energy. Furthermore, we can see that convergence has been achieved in 5 iterations.

▶ Add the following lines inside your `script.sh` to run a loop of scf calculations with different cutoffs and submit your script.

```
j=10
f=50
while [ $j -le $f ];do
#
  cp ../inputs/si.scf_Ry.in si.scf_"$j"Ry.in
  sed -i 's/AAA/'$j'/g' si.scf_"$j"Ry.in # replaces AAA with the new cutoff
  mpirun -np 4 $QE/pw.x -nk 4 < si.scf_"$j"Ry.in > si.scf_"$j"Ry.out
  j=$((j+5))
done
```

```
sbatch script.sh
```

To find the total energy from the output files:

```
$ grep ! si.scf_*out
```

The output is:

```
     si.scf_10Ry.out:!    total energy              =     -16.80271356 Ry
     si.scf_15Ry.out:!    total energy              =     -16.86633285 Ry
     si.scf_20Ry.out:!    total energy              =     -16.87580560 Ry
     si.scf_25Ry.out:!    total energy              =     -16.87737474 Ry
     si.scf_30Ry.out:!    total energy              =     -16.87766428 Ry
     si.scf_35Ry.out:!    total energy              =     -16.87782991 Ry
     si.scf_40Ry.out:!    total energy              =     -16.87787150 Ry
     si.scf_45Ry.out:!    total energy              =     -16.87787692 Ry
     si.scf_50Ry.out:!    total energy              =     -16.87788260 Ry
```

An acceptable convergence criterion is $\Delta E \sim 5 \times 10^{-}5$ Ry. Therefore we see that the total energy is converged for 40 Ry. Subsequent calculations should be performed with this converged value. Another typical convergence test is the **k**-point grid.

▶ Add the following lines inside your `script.sh` to run a loop of scf calculations with different **k**-point grids and submit your script.

```
j=3
f=14
```

```
while [ $j -le $f ];do
#
  cp ../inputs/si.scf_kkk.in si.scf_k"$j""$j""$j".in
  sed -i 's/KKK/'$j'/g' si.scf_k"$j""$j""$j".in
  mpirun -np 4 $QE/pw.x -nk 4 < si.scf_k"$j""$j""$j".in > si.scf_k"$j""$j""$j".out
  j=$((j+1))
done
```

```
sbatch script.sh
```

Note: Here we use a uniform grid $N_{k_x} \times N_{k_y} \times N_{k_z}$ with $N_{k_x} = N_{k_y} = N_{k_z}$ because the system is cubic with equal lattice constants along each Cart. direction. In general, the ratio between $N_{k_x}, N_{k_y}, N_{k_z}$ should reflect the ratio of the lattice constants.

The calculation should take 1-2 minutes. Can you find for which uniform grid the DFT total energy is converged, using:

```
$ grep ! si.scf_k*out
```

The answer is for a $10 \times 10 \times 10$ **k**-point grid or higher.

Can you also check the time it took `pw.x` to finish the calculation for each file? Type:

```
$ grep " PWSCF        :" si.scf_k*.out
```

The result is:

```
   si.scf_k101010.out:     PWSCF        :      0.69s CPU      0.77s WALL
   si.scf_k111111.out:     PWSCF        :      0.70s CPU      0.78s WALL
   si.scf_k121212.out:     PWSCF        :      0.94s CPU      1.05s WALL
   si.scf_k131313.out:     PWSCF        :      1.10s CPU      1.20s WALL
   si.scf_k141414.out:     PWSCF        :      1.17s CPU      1.28s WALL
   si.scf_k333.out:     PWSCF        :      0.19s CPU      0.29s WALL
   si.scf_k444.out:     PWSCF        :      0.24s CPU      0.31s WALL
   si.scf_k555.out:     PWSCF        :      0.28s CPU      0.35s WALL
   si.scf_k666.out:     PWSCF        :      0.33s CPU      0.41s WALL
   si.scf_k777.out:     PWSCF        :      0.37s CPU      0.45s WALL
   si.scf_k888.out:     PWSCF        :      0.50s CPU      0.59s WALL
   si.scf_k999.out:     PWSCF        :      0.56s CPU      0.65s WALL
```

The larger the **k**-point grid the slower the calculation as expected.

Now we are ready to perform a geometry optimization for finding the optimum lattice constant (note that fractional positions in silicon are fixed by symmetry).

▶ Before performing a geometry optimization with `vc-relax` we will explore how the DFT total energy varies with lattice constant, by varying the lattice constant "manually". To this aim add the following lines in your script and submit:

```
array=("9.90" "9.95" "10.00" "10.05" "10.10" "10.15" "10.20" "10.25"
         "10.30" "10.35" "10.40" "10.45" "10.50")
for j in ${array[@]}; do
#
  cp ../inputs/si.scf_lattice.in si.scf_"$j"_Bohr.in
```

```
  sed -i 's/AAA/'$j'/g' si.scf_"$j"_Bohr.in
  mpirun -np 4 $QE/pw.x -nk 4 < si.scf_"$j"_Bohr.in > si.scf_"$j"_Bohr.out
done
```

```
sbatch script.sh
```

The calculation should take aroun 1 minute. Can you find for which lattice constant the DFT total energy reaches a minimum:

```
$  grep ! si.scf_*r.out
```

The result is:

```
si.scf_10.00_Bohr.out:!    total energy              =     -16.92051621 Ry
si.scf_10.05_Bohr.out:!    total energy              =     -16.92247336 Ry
si.scf_10.10_Bohr.out:!    total energy              =     -16.92398458 Ry
si.scf_10.15_Bohr.out:!    total energy              =     -16.92506941 Ry
si.scf_10.20_Bohr.out:!    total energy              =     -16.92574679 Ry
si.scf_10.25_Bohr.out:!    total energy              =     -16.92603508 Ry
si.scf_10.30_Bohr.out:!    total energy              =     -16.92595210 Ry
si.scf_10.35_Bohr.out:!    total energy              =     -16.92551500 Ry
si.scf_10.40_Bohr.out:!    total energy              =     -16.92474050 Ry
si.scf_10.45_Bohr.out:!    total energy              =     -16.92364470 Ry
si.scf_10.50_Bohr.out:!    total energy              =     -16.92224304 Ry
si.scf_9.90_Bohr.out:!    total energy              =     -16.91518240 Ry
si.scf_9.95_Bohr.out:!    total energy              =     -16.91809284 Ry
```

Can you plot the results by saving the data in a file `data_lat_energy.txt` on your laptop:

```
9.90      -16.91518240
9.95      -16.91809284
10.00     -16.92051621
10.05     -16.92247336
10.10     -16.92398458
10.15     -16.92506941
10.20     -16.92574679
10.25     -16.92603508
10.30     -16.92595210
10.35     -16.92551500
10.40     -16.92474050
10.45     -16.92364470
10.50     -16.92224304
```

Below I provide the gnuplot commands I used. One can use `python` or any other tool to visualize the result:

```
$ module load gnuplot
$ gnuplot
```

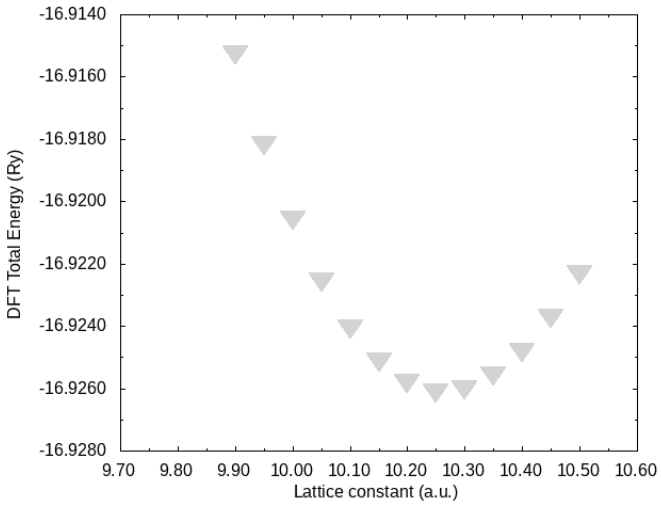
```
set terminal png
set output "figure_ex1.png"
set encoding iso_8859_1
```

```
set format y "%.4f"
set format x "%.2f"
set xlabel "Lattice constant (a.u.)" offset 0,0.5
set ylabel "DFT Total Energy (Ry)" offset 0.95,0
set style data linespoints
set mxtics 2
set mytics 2

set xrange [9.7:10.6]

plot  "data_lat_energy.txt" u 1:($2) not  w p pt 11 ps 2.6 lw 2 lc rgb "light-grey"
exit
```

The result is:



We can see that the minimum is obtained for a lattice constant of 10.25 Bohr (or 5.424 Å).

▶ Run a vc-relax calculation for silicon in the `workdir`. The input file `si.vc-relax.in` is:

```
&control
   calculation = 'vc-relax'
   restart_mode = 'from_scratch',
   prefix = 'si',
   pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
   verbosity = 'high'
   outdir='./'
/
&system
   ibrav = 2,
   celldm(1) = 10.20,
   nat =  2,
   ntyp = 1,
   ecutwfc = 40.0
/
&electrons
```

```
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
&ions
    ion_dynamics = 'bfgs'
 /
&cell
    cell_dynamics='bfgs'
    press_conv_thr = 0.005
/
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
10 10 10 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

Note: We add control flags `&ions` and `&cell` for the geometry optimization. Note flag `press_conv_thr` is for the convergence threshold on the pressure (in Kbar) acting on the system which, in principle, should be reduced to zero.

```
$  cp ../inputs/si.vc-relax.in .
$  mpirun -np 4 $QE/pw.x -nk 4 < si.vc-relax.in > si.vc-relax.out
```

The calculation should take around 4 secs. To find the final lattice constant type:

```
$  grep -4 CELL_PARAM si.vc-relax.out
```

The result is:

```
     new unit-cell volume =    267.75570 a.u.^3 (    39.67730 Ang^3 )
     density =      2.35086 g/cm^3

CELL_PARAMETERS (alat= 10.20000000)
  -0.501536720  -0.000000000   0.501536720
   0.000000000   0.501536720   0.501536720
  -0.501536720   0.501536720  -0.000000000

--

     new unit-cell volume =    270.11762 a.u.^3 (    40.02730 Ang^3 )
     density =      2.33030 g/cm^3

CELL_PARAMETERS (alat= 10.20000000)
  -0.503007125  -0.000000000   0.503007125
   0.000000000   0.503007125   0.503007125
  -0.503007125   0.503007125  -0.000000000

--
```

```
     new unit-cell volume =    270.27278 a.u.^3 (    40.05029 Ang^3 )
     density =      2.32897 g/cm^3

CELL_PARAMETERS (alat= 10.20000000)
  -0.503103416  -0.000000000   0.503103416
   0.000000000   0.503103416   0.503103416
  -0.503103416   0.503103416  -0.000000000

--
Begin final coordinates
     new unit-cell volume =    270.27278 a.u.^3 (    40.05029 Ang^3 )
     density =      2.32897 g/cm^3

CELL_PARAMETERS (alat= 10.20000000)
  -0.503103416  -0.000000000   0.503103416
   0.000000000   0.503103416   0.503103416
  -0.503103416   0.503103416  -0.000000000
```

Based on the final result (i.e. after text "`Begin final coordinates`"), the lattice constant is $10.2 \times 0.503103416/0.5 = 10.263$ Bohr (5.431 Å), in good agreement with the value obtained following the "manual" approach. Our value is in excellent agreement with the experimental value, too. We remark that here we are using the PBEsol functional (i.e. the PBE functional revised for solids) which is known to yield better lattice constants compared to PBE or LDA.

Homework: One could try as a homework to perform the relaxation with different functionals which can be downloaded from PseudoDojo or other libraries as shown in the lectures.

Exercise 2

In this exercise we will calculate the DFT electronic band structure of silicon, starting from the converged settings obtained from `Exercise 1`. We will also perform a hybrid functional calculation for a better treatment to exchange-correlation to improve the band gap agreement with experiment.

▶ Run a self-consistent calculation for silicon in the `workdir`. The input file `si.scf.in` is:

```
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.263,
    nat =  2,
    ntyp = 1,
    ecutwfc = 40.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
10 10 10 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

First go to the directory `exercise2` and run the scf in the `workdir`:

```
$  cd ../../exercise2/; mkdir workdir; cd workdir
$  cp ../inputs/si.scf.in .
$  mpirun -np 40 $QE/pw.x -nk 20 < si.scf.in > si.scf.out
```

Note: Here we parallelize over all CPUs (`-np 40`) and we choose to parallelize over **k**-points using 20 "pools". Can you confirm that the calculation run faster than an scf calculation with the same settings in Exercise 1?

Then copy the following input files in the `workdir`:

```
$  cp ../inputs/si.bands.in .; cp ../inputs/bands.in .
```

Note: in file `si.bands.in` we set `nbnd = 5` to include one empty band. We also sample the L-Γ-X path using 101 **k**-points in crystal coordinates. To better understand the high-symmetry points of the FCC lattice, visit this link. Below we provide the files:

```
--                                                                    si.bands.in
 &control
    calculation = 'bands'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.263,
    nat =  2,
    ntyp = 1,
    ecutwfc = 40.0,
    nbnd = 5
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS  {crystal_b}
3
 0.500000   0.500000   0.500000  50 ! L
 0.000000   0.000000   0.000000  50 ! G
 0.000000   0.500000   0.500000   1 ! X
```

```
--                                                                    bands.in
&BANDS
 prefix  = 'si'
 outdir  = './'
 filband = 'bands.dat'
/
```

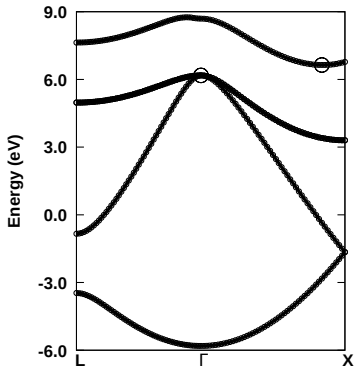
▶ Run a standard band structure calculation in the unit-cell of silicon along L-Γ-X using 101 **k**-points.

```
$  mpirun -np 40 $QE/pw.x -nk 20 < si.bands.in > si.bands.out
$  mpirun -np 40 $QE/bands.x -nk 20 < bands.in > bands.out
```

▶ Plot the band structure using the file **bands.dat.gnu**.

```
$  cp ../inputs/gp_coms.p .
```

```
$  gnuplot gp_coms.p
$  evince band_structure.pdf # if not working, copy file locally
```



Can you find the Cartesian k-coordinates in units of \tpiba (i.e. π/a, where a is the lattice constant) of the valence band maximum (VBM) and conduction band minimum (CBM)? Inspecting file **bands.dat**, those are:

```
            0.000000  0.000000  0.000000
   -5.811    6.179     6.179     6.179     8.692
...
            0.000000  0.840000  0.000000
   -2.824   -0.353     3.404     3.404     6.639
```

Silicon is an indirect band gap material. The VBM lies at the Γ-point and is threefold degenerated with energy $E_{\mathrm{VBM}} = 6.179$ eV. The CBM lies at $\sim$0.84 Γ-X with energy $E_{\mathrm{CBM}} = 6.639$ eV. Therefore, the DFT-PBEsol band gap with these settings is $E_{\mathrm{G}} = E_{\mathrm{CBM}} - E_{\mathrm{VBM}} = 0.46$ eV. This value underestimates the experimental value, $E_{\mathrm{G}}^{\mathrm{expt.}} = 1.17$ eV substantially. The first direct gap is at the Γ point giving: $E_{\mathrm{G}}^{\Gamma} = E_{\mathrm{CBM}}^{\Gamma} - E_{\mathrm{VBM}}^{\Gamma} = 8.692 - 6.179 = 2.513$ eV. The experimental direct band gap is about 3.4 eV. The band gap underestimation is mainly a consequence of the incomplete treatment of exchange-correlation effects within DFT.

▶ Run a self-consistent calculation for silicon using the HSE hybrid functional with one empty band to evaluate the HSE direct gap of silicon at Γ-point. The input file `si.HSE.in` is:

```
&control
   calculation = 'scf'
   restart_mode = 'from_scratch',
   prefix = 'si_HSE',
   pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
   verbosity = 'high'
   outdir='./'
/
&system
   ibrav = 2,
   celldm(1) = 10.263,
   nat =  2,
   ntyp = 1,
   ecutwfc = 40.0
   nbnd = 5,
```

```
    input_dft = 'hse',    ! Specifies the HSE hybrid functional
    exx_fraction = 0.25, ! 25% exact exchange as in PBE0
    screening_parameter = 0.106,  ! Set to 0 for full-range hybrid functional
    nqx1=6, nqx2=6, nqx3=6 ! mesh for q sampling the Fock operator (EXX)
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
10 10 10 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

Note: The screening parameter 0.106 defines the distance beyond which the Coulomb interaction is replaced by screened (semi-local) exchange.

```
$  cp ../inputs/si.HSE.in .
$  mpirun -np 40 $QE/pw.x -nk 4 < si.HSE.in > si.HSE.out
```

To find the direct band gap at Γ type:

```
$ grep -3 "k = 0.0000 0.0000 0.0000 " si.HSE.out |tail -4
```

The output is:

```
          k = 0.0000 0.0000 0.0000 (  1139 PWs)   bands (ev):

    -7.4032   5.8609   5.8609   5.8609   9.1825
```

The HSE direct gap is $E_{\mathrm{G}}^{\Gamma} = E_{\mathrm{CBM}}^{\Gamma} - E_{\mathrm{VBM}}^{\Gamma} = 9.1825 - 5.8609 = 3.3216$ eV which is in excellent agreement with experiment.

▶ Run a self-consistent calculation for silicon using the PBE0 hybrid functional with one empty band to evaluate the PBE0 direct gap of silicon at Γ-point. The input file `si.PBE0.in` is:

```
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si_PBE0',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.263,
```

```
    nat =  2,
    ntyp = 1,
    ecutwfc = 40.0
    nbnd = 5,
    input_dft = 'pbe0', ! Specifies the PBE0 hybrid functional, or 'HSE'
    exx_fraction = 0.25, ! PBE0 uses 25% exact exchange
    screening_parameter = 0.0, ! Set to 0 for full-range hybrid functional
    nqx1=6, nqx2=6, nqx3=6 ! mesh for q sampling the Fock operator (EXX)
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
10 10 10 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

```
$  cp ../inputs/si.PBE0.in .
$  mpirun -np 40 $QE/pw.x -nk 4 < si.PBE0.in > si.PBE0.out
```

To find the direct band gap at Γ type:

```
$ grep -3 "k = 0.0000 0.0000 0.0000 " si.PBE0.out |tail -4
```

The output is:

```
                  k = 0.0000 0.0000 0.0000 (  1139 PWs)   bands (ev):

    -7.8257   5.5224   5.5224   5.5224   9.5170
```

The PBE0 direct gap is $E_{\mathrm{G}}^{\Gamma} = E_{\mathrm{CBM}}^{\Gamma} - E_{\mathrm{VBM}}^{\Gamma} = 9.5170 - 5.5224 = 3.9946$ eV which overestimates experiment by $\sim$0.6 eV.

PBE0 uses unscreened, full-range Fock exchange, treating electron-electron interactions as if they were in vacuum, without accounting for the dielectric screening present in solids. HSE, by damping the long-range part of the exchange, effectively includes this screening. That's why PBE0 tends to overestimate gaps in several solids, while HSE gives more realistic results. PBE0 can be better when the system's dielectric constant is small (i.e. weak screening and electrons localized).

Exercise 3

In this exercise we will calculate the phonon dispersion of silicon, starting from the converged settings obtained from `Exercise 1`.

▶ Run a self-consistent calculation for silicon in the `workdir`. The input file `si.scf.in` is:

```
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'si',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.263,
    nat =  2,
    ntyp = 1,
    ecutwfc = 40.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
 Si  28.086  Si.upf
K_POINTS automatic
10 10 10 0 0 0
ATOMIC_POSITIONS {crystal}
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
```

```
$  cd ../../exercise3/; mkdir workdir; cd workdir
$  cp ../inputs/si.scf.in .
$  mpirun -np 40 $QE/pw.x -nk 20 < si.scf.in > si.scf.out
```

Note: Here we parallelize over all CPUs (`-np 40`) and we choose to parallelize over **k**-points using 20 "pools". Can you confirm that the calculation run faster than an scf calculation with the same settings in Exercise 1?

▶ Run a `ph.x` calculation on a homogeneous $4 \times 4 \times 4$ $\mathbf{q}$-point grid using the input:

```
--                                                          si.ph.in
&inputph
   amass(1) = 28.0855,
   prefix   = 'si'
   outdir   = './'
   ldisp    = .true.
   fildyn   = 'si.dyn'
```

```
   tr2_ph   = 1.0d-12
   nq1      = 4, nq2 = 4, nq3 = 4
/
```

```
$ cp ../inputs/si.ph.in .
$ mpirun -np 40 $QE/ph.x -nk 20 < si.ph.in > si.ph.out
```

This will generate 8 **si.dynX** output files containing the dynamical matrix calculated for each irreducible $\mathbf{q}$-point (i.e. not symmetry-equivalent to any other in the chosen q-point). The list of irreducible $\mathbf{q}$-points is written in the **si.dyn0** file:

```
   4   4   4
   8
   0.000000000000000E+00   0.000000000000000E+00   0.000000000000000E+00
  -0.250000000000000E+00   0.250000000000000E+00  -0.250000000000000E+00
   0.500000000000000E+00  -0.500000000000000E+00   0.500000000000000E+00
   0.000000000000000E+00   0.500000000000000E+00   0.000000000000000E+00
   0.750000000000000E+00  -0.250000000000000E+00   0.750000000000000E+00
   0.500000000000000E+00   0.000000000000000E+00   0.500000000000000E+00
   0.000000000000000E+00  -0.100000000000000E+01   0.000000000000000E+00
  -0.500000000000000E+00  -0.100000000000000E+01   0.000000000000000E+00
```

▶ Run a `q2r.x` calculation to obtain the interatomic force constants (IFC) file using the input:

```
--                                                                    q2r.in
&input
 fildyn='si.dyn', flfrc = 'si.444.fc'
/
```

```
$  cp ../inputs/q2r.in .
$  mpirun -np 1 $QE/q2r.x < q2r.in > q2r.out
```

This will generate **si.444.fc** which contains information of the lattice and atomic positions, the high-frequency dielectric constant (after `T`), Born-Effective charges (which should be zero for a non-polar material like silicon), the grid, and the IFCs, i.e.:

```
$ head -20 si.444.fc
```

```
  1    2   2 10.2630000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
          1  'Si    '    25598.3672898282
    1    1      0.0000000000      0.0000000000      0.0000000000
    2    1     -0.2500000000      0.2500000000      0.2500000000
 T   2.66801901878596
        13.715859204167          0.000000000000          0.000000000000
         0.000000000000         13.715859204167          0.000000000000
         0.000000000000          0.000000000000         13.715859204167
    1
     -0.0284020      0.0000000      0.0000000
      0.0000000     -0.0284020      0.0000000
      0.0000000      0.0000000     -0.0284020
    2
     -0.0284020      0.0000000      0.0000000
      0.0000000     -0.0284020      0.0000000
```

```
      0.0000000      0.0000000     -0.0284020
   4   4   4
   1   1   1   1
   1   1   1   2.71887565744E-01
   2   1   1  -3.59085827318E-03
   ...
```

▶ Run a `matdyn.x` calculation to check the phonon dispersion:

matdyn.in

```
&input
    asr='all', amass(1)=28.0855,
    flfrc='si.444.fc', fldyn='si.dyn.mat', flfrq='si.freq', fleig='si.dyn.eig',
    q_in_cryst_coord = .false., q_in_band_form = .true.
/
 9
 0.00  0.00  0.00  100
 0.75  0.75  0.00  1
 0.25  1.00  0.25  100
 0.00  1.00  0.00  100
 0.00  0.00  0.00  100
 0.50  0.50  0.50  100
 0.00  1.00  0.00  100
 0.50  1.00  0.00  100
 0.50  0.50  0.50  100
```

```
$  cp ../inputs/matdyn.in .
$  $QE/matdyn.x < matdyn.in > matdyn.out
$  $QE/plotband.x
```

```
     Input file > si.freq
Reading    6 bands at    702 k-points
Range:    0.0000  509.4627eV  Emin, Emax, [firstk, lastk] > 0 510
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   0.0000
high-symmetry point:  0.7500 0.7500 0.0000   x coordinate   1.0607
high-symmetry point:  0.2500 1.0000 0.2500   x coordinate   1.0607
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   1.4142
high-symmetry point:  0.0000 0.0000 0.0000   x coordinate   2.4142
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   3.2802
high-symmetry point:  0.0000 1.0000 0.0000   x coordinate   4.1463
high-symmetry point:  0.5000 1.0000 0.0000   x coordinate   4.6463
high-symmetry point:  0.5000 0.5000 0.5000   x coordinate   5.3534
output file (gnuplot/xmgr) > si_ph_dispersion.xmgr
bands in gnuplot/xmgr format written to file si_ph_dispersion.xmgr
output file (ps) >
stopping ...
```

`matdyn.x` will generate **si.freq** which contains the phonon frequencies along the specified path in the Brillouin zone. Then we combine **si.freq** with `plotband.x` to obtain **si_ph_dispersion.xmgr** in gnuplot/xmgr format for the phonon dispersion. Below I provide the gnuplot commands I used. One can use python or any other tool to visualize the result:

```
$ module load gnuplot
$ gnuplot
```

```
set terminal pdf color enhanced size 4,3 font 'Arial-Bold'
set output "ph_dispersion_444.pdf"
set ytics 100
```

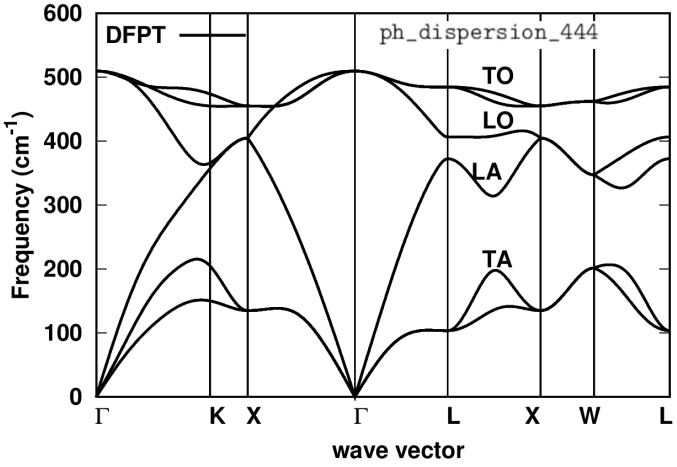
```
set xrange [0:5.3534]
set yrange [0:600]
unset xtics
set key left
set format y "%.0f"
set xlabel "wave vector" offset 1.1,-1.3
set ylabel "Frequency (cm^{-1})"
set arrow from 1.0607,0 to 1.0607,600 nohead
set arrow from 1.4142,0 to 1.4142,600 nohead
set arrow from 2.4142,0 to 2.4142,600 nohead
set arrow from 3.2802,0 to 3.2802,600 nohead
set arrow from 4.1463,0 to 4.1463,600 nohead
set arrow from 4.6463,0 to 4.6463,600 nohead
set arrow from 5.3534,0 to 5.3534,600 nohead
set label "{/Symbol G}" at -0.015,-30
set label "K" at 1.05,-30
set label "X" at 1.40,-30
set label "{/Symbol G}" at 2.4,-30
set label "L" at 3.265,-30
set label "X" at 4.000,-30
set label "W" at 4.500,-30
set label "L" at 5.250,-30
#
set label "TA" at 3.6,215
set label "LA" at 3.5,352
set label "LO" at 3.6,432
set label "TO" at 3.6,502
set style data linespoints

plot "si_ph_dispersion.xmgr" u 1:($2) t "DFPT" w l lw 3 lc rgb "black"

exit
```

```
$  cp ../inputs/gp_coms.p .
$  gnuplot gp_coms.p; evince ph_dispersion_444.pdf
```

Is the phonon dispersion as you expect? One can verify the results with literature data.



Exercise 4

Can you repeat Exercises 1-3 for GaAs? For help we provide the initial scf input file with unconverged settings:

```
 &control
    calculation = 'scf'
    restart_mode = 'from_scratch',
    prefix = 'GaAs',
    pseudo_dir = '/nvme/h/mzacharias/nc-sr-04_pbesol_standard_upf/',
    verbosity = 'high'
    outdir='./'
 /
 &system
    ibrav = 2,
    celldm(1) = 10.40,
    nat =  2,
    ntyp = 2,
    ecutwfc = 10.0
 /
 &electrons
    diagonalization='david'
    mixing_mode = 'plain'
    mixing_beta = 0.7
    conv_thr = 1.0d-7
 /
ATOMIC_SPECIES
Ga 69.723 Ga.upf
As 74.92 As.upf
K_POINTS automatic
3 3 3 0 0 0
ATOMIC_POSITIONS {crystal}
 Ga 0.00 0.00 0.00
 As 0.25 0.25 0.25
```