Chapter 3

Hands-On Tutorials of Quantum ESPRESSO

In this chapter, we show how to run Quantum ESPRESSO. We focus on several topics for obtaining the electronic, optical, transport, and mechanical properties of materials. Each section starts with an explanation of the background and purpose. Then, we show the command to run Quantum ESPRESSO by showing examples of input and output files. Before practicing with these tutorials, please ensure that Quantum ESPRESSO and all the necessary software packages have been installed on your computer (see Chapter 2 for installation). In Chapter 3, we only show how to run Quantum ESPRESSO. In order to understand how Quantum ESPRESSO works, the basic concepts of the density-functional theory are explained in Chapter 4. The concepts of solid-state physics behind these hands-on tutorials are briefly summarized in Chapter 5. Calculation job and plotting scripts are written in Bash and Python, which are explained in Chapter 6. We recommend using XCrySDen and VESTA for the visualization of input and output files.

We adopt two-dimensional graphene as an example of a material for these tutorials. Since graphene consists of two carbon atoms in a hexagonal unit-cell, we can quickly run Quantum ESPRESSO for

Quantum ESPRESSO Course for Solid-State Physics
Nguyen T. Hung, Ahmad R. T. Nugraha, and Riichiro Saito
Copyright © 2022 Jenny Stanford Publishing Pte. Ltd.
ISBN 978-981-4968-37-9 (Hardcover), 978-1-003-29096-4 (eBook)
www.jennystanford.com

graphene. The calculation takes only a few minutes on desktop or notebook computers. Other materials such as MoS_2 and GeTe also appear in GitHub (`https://github.com/nguyen-group/QE-SSP`). It is noted that the format of the input files might be modified from version to version of Quantum ESPRESSO. In this book, we stick to the format of Quantum ESPRESSO version 7.0. However, alternative code lines will be provided in the input files whenever it is necessary to provide backward compatibility. The source files and scripts from this chapter are also available online in GitHub (`https://github.com/nguyen-group/QE-SSP`). The readers can download the whole repository, view them in a terminal, or read them at GitHub, where they are automatically rendered.

3.1 Basic parameters

When starting a new job of simulations with Quantum ESPRESSO, we first need to determine the parameters in the input file for each step of the calculation. In this section, we use graphene as an example material.

3.1.1 Total energy and self-consistent field calculations

❏ **Purpose:** By calculating the total energy with the self-consistent field (SCF) method, we get the ground-state properties, which will be used for electron and phonon dispersion calculations in Secs. 3.2 and 3.3, respectively.
❏ **Background:** The total energy and the SCF calculation are two concepts that we should know for this tutorial. Both concepts are given in Secs. 4.5 and 4.12.
❏ **How to run:** To run the SCF for the input file, the readers should type the following command lines:[1]

```
$ cd ~/QE-SSP/gr/scf/
$ mpirun -np 4 pw.x < scf.in > scf.out &
```

[1] If you do not find 'mpirun' command, you need to install libopenmpi-dev (see Chapter 2).

- Line 1: Go to the `scf` directory that includes the input files.
- Line 2: Run `pw.x` (`pw` = plane-wave, `x` = executable file) by using in parallel with 4 processors (`-np 4`) by the command `mpirun` (running a program with parallel processors) with the input file is `scf.in` and the output file is `scf.out`. The symbol `&` makes the command run in the background. Here, we select 4 processes for parallel calculations, but the readers can run with serial calculations (without `mpirun`, that is `pw.x < scf.in > scf.out &`) or any number of processes (e.g., `-np 8` for Intel Core i7 or Core i9), depending on your computer. For detailed commands for running in parallel, the readers can find on Sec. 6.2.3.

❑ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

It will display a listing of all files in the `scf` directory as

```
scf.in  scf.out
```

If the readers see the `scf.out`, the readers can check the contents of `scf.out` by `tail` command as follows:

```
$ tail scf.out
```

The `tail` command prints the last few lines of the `scf.out` file. If the calculation normally finishes, a message `JOB DONE` is written at the end of this file as

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

❑ **Input file:** Now, let us explain the details of the input file. To see the input file, the readers can use `vi` editor by typing:

```
$ vi scf.in
```

To quit the vi editor without saving any changes, the readers must press : (colon). Then the cursor should reappear in the lower-left corner of the screen beside a colon prompt. After that, the readers need to enter q! to quit the file without saving. Since vi editor is usually available in all Linux distributions, we would like to use vi. However, the readers can use any text editor, such as Emacs or Notepad, to open scf.in directly. When the readers open scf.in file, the readers will see the input variables as

QE-SSP/gr/scf/scf.in

```
 1 &CONTROL
 2 calculation  = 'scf'
 3 pseudo_dir   = '../pseudo/'
 4 outdir       = '../tmp/'
 5 prefix       = 'gr'
 6 /
 7 &SYSTEM
 8 ibrav        = 4
 9 a            = 2.4623
10 c            = 10.0
11 nat          = 2
12 ntyp         = 1
13 occupations  = 'smearing'
14 smearing     = 'mv'
15 degauss      = 0.02
16 ecutwfc      = 60
17 /
18 &ELECTRONS
19 mixing_beta  = 0.7
20 conv_thr     = 1.0D-6
21 /
22 ATOMIC_SPECIES
23 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
24 ATOMIC_POSITIONS (crystal)
25 C  0.333333333  0.666666666  0.500000000
26 C  0.666666666  0.333333333  0.500000000
27 K_POINTS (automatic)
28 12 12 1 0 0 0
```

☞ **Explanation of scf.in:** There are three mandatory namelists (CONTROL, SYSTEM, and ELECTRONS), which start by &, and three mandatory input cards (ATOMIC_SPECIES, ATOMIC_POSITIONS, and K_POINTS in lines 22, 24, and 27, respectively). The SCF calculation is performed by selecting calculation = 'scf' in the namelist

Table 3.1 Meaning of input variables in `scf.in` file.

Line	Syntax	Meaning
1	`&CONTROL`	A *mandatory namelist* includes input variables that control the flux of the calculation and the amount of I/O on disk and on the screen.
2	`calculation`	A string describing the task to be calculated, in which `'scf'` is the SCF calculation.
3	`pseudo_dir`	Directory containing pseudopotential files.
4	`outdir`	Temporary folder to save output data.
5	`prefix`	Filenames of output data in tmp folder.
6	`/`	End of namelist `CONTROL`.
7	`&SYSTEM`	A *mandatory namelist* includes input variables that specify the system for the calculation.
8	`ibrav`	Bravais lattice index (see Table 3.2).
9, 10	`a, c`	Lattice constants in Angstrom unit.
11	`nat`	Number of atoms per unit cell.
12	`ntyp`	Number of types of atoms in unit cell.
13	`ecutwfc`	Cut-off energy (Ry) for pseudopotentials.
14	`occupations`	Gaussian smearing for the case of metal.
15	`smearing`	Smearing method, in which `'mv'` is the Marzari-Vanderbilt-DeVita-Payne cold smearing.
16	`degauss`	Value of the gaussian spreading (Ry) for Brillouin-zone integration in metal.
17	`/`	End of namelist `SYSTEM`.
18	`&ELECTRONS`	A *mandatory namelist* includes input variables that control the algorithms used to reach the SCF solution for the electrons.
19	`mixing_beta`	Mixing factor for self-consistency.
20	`conv_thr`	Convergence threshold for self-consistency.
21	`/`	End of namelist `ELECTRONS`.

Continued

Table 3.1 – *Continued*

Line	Syntax	Meaning
22, 23	`ATOMIC_SPECIES`	A *mandatory input card* includes name, mass and pseudopotential used for each atomic species present in the system.
24–26	`ATOMIC_POSITIONS`	A *mandatory input card* includes type and coordinates of each atom in the unit cell. The option `'crystal'` indicates that atomic positions are in crystal coordinates.
27, 28	`K_POINTS`	A *mandatory input card* includes information of the ***k***-points used for Brillouin-zone integration.

CONTROL in the input file. In the namelist ELECTRONS, we need to set several variables that can be specified to control the SCF calculation. A short description of the input variables is given in Table 3.1. If the readers want to know the original information of input variables, the readers can visit the following web page: `https://www.quantum-espresso.org/Doc/INPUT_PW.html`.
☞ **Visualizing structure from scf.in:** To ensure that structure of the material is correctly represented by the input file, it is crucial to visualize the structure by reading the input file in XCrySDen software. XCrySDen is one of the visualization tools that we can use to check the structure directly. To run XCrySDen, we enter the command:

```
$ xcrysden &
```

One the window of XCrySDen is opened as shown in Fig. 3.1, we select tabs: **File → Open PWscf → Open PWscf Input File** and select `scf.in`. In the box **[PWSCF Input: “scf.in”]**, we click on the **OK** button, XCrySDen will automatically identify the structure of graphene from the input file (see Fig. 3.1).

The structure of graphene in Fig. 3.1 includes two concepts: “bravais lattice” and “atomic basis”. A bravais lattice is specified by an integer number `ibrav` and six lattice constants a, b, c, `cosAB`, `cosAC`, and `cosBC` in the namelist SYSTEM. Here, `cosAB` = cosine of the angles between a and b (γ), `cosAC` = cosine of the angles between

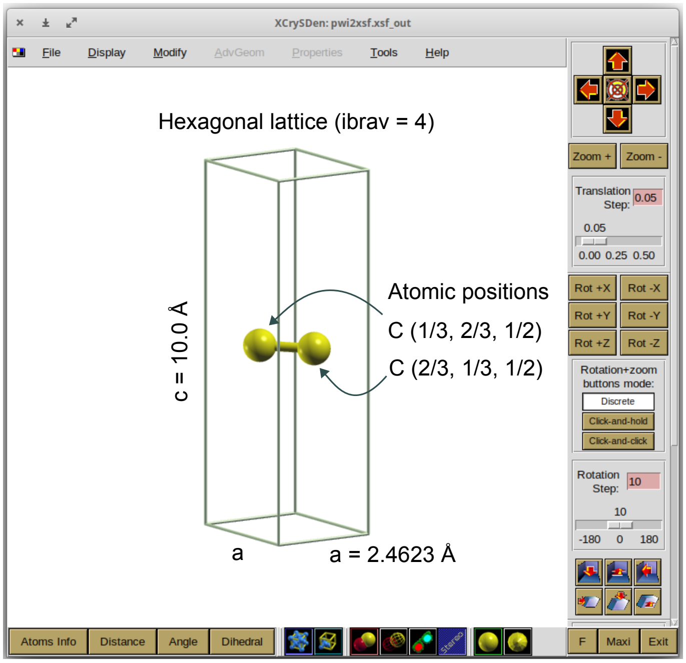


Figure 3.1 Visualizing structure of material from Quantum ESPRESSO input file by using XCrySDen.

a and c (β), and `cosBC` = cosine of the angles between b and c (α). The list of the bravais lattice indexes in Quantum ESPRESSO is shown in Table 3.2. Since graphene has a hexagonal lattice, we select `ibrav = 4` (hexagonal lattice) and the lattice constants $a = b = 2.4623$ Å and $c = 10$ Å as shown in Fig. 3.1. To confirm that our unit cell is hexagonal or not, let us increase the number of unit cells in XCrySDen. From XCrySDen in Fig. 3.1, we select **Modify → Number of Unit Drawn**. In the box **[Modify Number of Unit Drawn]**, we choose 3, 3, and 2 in the x, y, and z directions, respectively. Then we can see a $3 \times 3 \times 2$ supercell, as shown in Fig. 3.2. Since the periodic boundary conditions are always applied for any directions in Quantum ESPRESSO, for the "two-dimensional graphene", we set c `= 10.0` Å as a sufficiently large value compared with distance of two graphene layers in graphite (3.35 Å) to avoid undesirable interactions in the z-direction. Nevertheless, too large c requires computational power.

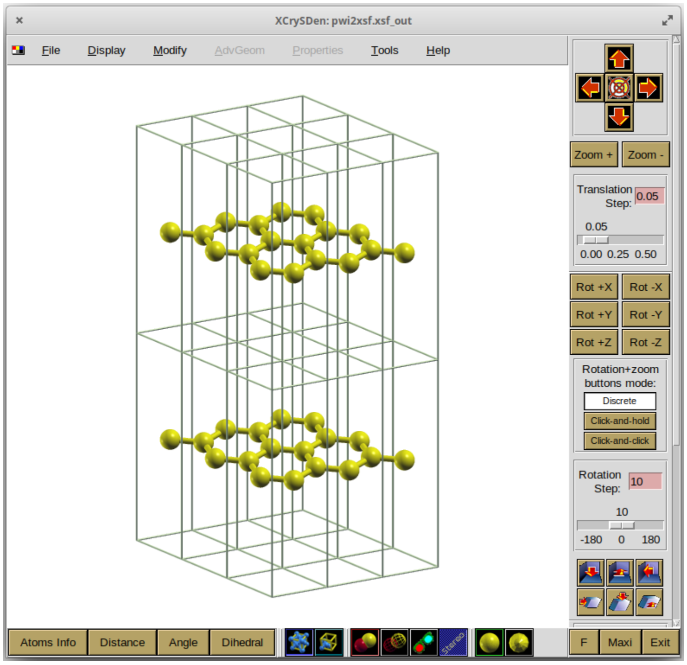


Figure 3.2 Visualizing a supercell $3 \times 3 \times 2$ of graphene by using XCrySDen. We can see hexagonal lattice.

☞ **Note**: For the lattice constants, we can specify either [a, b, c, cosAB, cosAC, and cosBC] OR [celldm(1)- celldm(6)] but NOT both, in which celldm(1) = a (in a.u.), celldm(2) = b/a (in a.u.), celldm(3) = c/a (in a.u.), celldm(4) = cosAB, celldm(5) = cosAC, and celldm(6) = cosBC.

Since the bravais lattice is set, we now check the atomic basis. The atomic basis consists of the number, type, and positions of atoms in the unit cell, in which the number and type of atoms are specified by the integer numbers nat and ntyp in the namelist SYSTEM, respectively, while the positions of atoms are specified in the card ATOMIC_POSITIONS in the crystal coordinates (crystal), i.e., the values of the positions are between 0.0 and 1.0 for each direction. For graphene, there are two C atoms in the unit cell. Thus, we set nat = 2, ntyp = 1, and the atomic positions are (0.333333333 0.666666666 0.500000000)

Table 3.2 Bravais lattice table in Quantum ESPRESSO.

ibrav	Structure	Lattice vectors
0	Free	Lattice vectors are given in card `CELL_PARAMETERS`
1	Cubic P	$v_1 = a(1,0,0), v_2 = a(0,1,0), v_3 = a(0,0,1)$
2	Cubic F	$v_1 = \frac{a}{2}(-1,0,1), v_2 = \frac{a}{2}(0,1,1),$ $v_3 = \frac{a}{2}(-1,1,0)$
3	Cubic I	$v_1 = \frac{a}{2}(1,1,1), v_2 = \frac{a}{2}(-1,1,1),$ $v_3 = \frac{a}{2}(-1,-1,1)$
-3	Cubic I	$v_1 = \frac{a}{2}(-1,1,1), v_2 = \frac{a}{2}(1,-1,1),$ $v_3 = \frac{a}{2}(1,1,-1)$
4	Hexagonal and Trigonal P	$v_1 = a(1,0,0), v_2 = a(-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0),$ $v_3 = a(0,0,\frac{c}{a})$
5	Trigonal R, 3-fold axis c	$v_1 = a(x,-y,z), v_2 = a(0,2y,z),$ $v_3 = a(-x,-y,z)$, where $x = \sqrt{\frac{1-c}{2}}$, $y = \sqrt{\frac{1-c}{6}}, z = \sqrt{\frac{1+2c}{3}}, c = \cos\gamma$
-5	Trigonal R, 3-fold axis $\langle 111 \rangle$	$v_1 = \frac{a}{\sqrt{3}}(u,v,v), v_2 = \frac{a}{\sqrt{3}}(u,v,v),$ $v_3 = \frac{a}{\sqrt{3}}(v,v,u)$, where $u = z - 2\sqrt{2}y$ and $v = z + \sqrt{(2)}y$ with y, z as for case `ibrav = 5`
6	Tetragonal P	$v_1 = a(1,0,0), v_2 = a(0,1,0), v_3 = a(0,0,\frac{c}{a})$
7	Tetragonal I	$v_1 = \frac{a}{2}(1,-1,\frac{c}{a}), v_2 = \frac{a}{2}(1,1,\frac{c}{a}),$ $v_3 = \frac{a}{2}(-1,-1,\frac{c}{a})$
8	Orthorhombic P	$v_1 = (a,0,0), v_2 = (0,b,0), v_3 = (0,0,c)$
9	Orthorhombic	$v_1 = (\frac{a}{2}, \frac{b}{2}, 0), v_2 = (-\frac{a}{2}, \frac{b}{2}, 0), v_3 = (0,0,c)$
-9	Orthorhombic	$v_1 = (\frac{a}{2}, -\frac{b}{2}, 0), v_2 = (\frac{a}{2}, \frac{b}{2}, 0), v_3 = (0,0,c)$
91	Orthorhombic	$v_1 = (a,0,0), v_2 = (0, \frac{b}{2}, -\frac{c}{2}), v_3 = (0, \frac{b}{2}, \frac{c}{2})$
10	Orthorhombic	$v_1 = (\frac{a}{2}, 0, \frac{c}{2}), v_2 = (\frac{a}{2}, \frac{b}{2}, 0), v_3 = (0, \frac{b}{2}, \frac{c}{2})$
11	Orthorhombic	$v_1 = (\frac{a}{2}, \frac{b}{2}, \frac{c}{2}), v_2 = (-\frac{a}{2}, \frac{b}{2}, \frac{c}{2}),$ $v_3 = (-\frac{a}{2}, -\frac{b}{2}, \frac{c}{2})$
12	Monoclinic P (unique axis c)	$v_1 = (a,0,0), v_2 = (b\cos\gamma, b\sin\gamma, 0), v_3 = (0,0,c)$

Continued

Table 3.2 – *Continued*

ibrav	Structure	Lattice vectors
-12	Monoclinic P (unique axis b)	$v_1 = (a, 0, 0), v_2 = (0, b, 0),$ $v_3 = (c\cos\beta, 0, c\sin\beta)$
13	Monoclinic base-centered (unique axis c)	$v_1 = (\frac{a}{2}, 0, -\frac{c}{2}), v_2 = (b\cos\gamma, b\sin\gamma, 0),$ $v_3 = (\frac{a}{2}, 0, \frac{c}{2})$
-13	Monoclinic base-centered (unique axis b)	$v_1 = (\frac{a}{2}, \frac{b}{2}, 0), v_2 = (-\frac{a}{2}, \frac{b}{2}, 0, 0),$ $v_3 = (c\cos\beta, 0, c\sin\beta)$
14	Triclinic	$v_1 = (a, 0, 0), v_2 = (b\cos\gamma, b\sin\gamma, 0),$ $v_3 = (c\cos\beta, c\frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma},$ $c\frac{\sqrt{1+2\cos\alpha\cos\beta\cos\gamma - \cos^2\alpha - \cos^2\beta - \cos^2\gamma}}{\sin\gamma})$

Note: α, β, and γ are angles between axis b and c, a and c, and a and b, respectively.

and `(0.666666666 0.333333333 0.500000000)` in the crystal coordinates.

☞ **Note**: In the card `ATOMIC_POSITIONS`, the positions of atoms can specify *in units of the lattice parameter* (either `a` or `celldm(1)`) with `(alat)` option. It is noted that `(alat)` is default option in the card `ATOMIC_POSITIONS`. For the `(alat)` option, the lines 24–26 in the `scf.in` file will be changed as follows:

```
ATOMIC_POSITIONS (alat)
C   0.0000000   0.5773503   2.0306218
C   0.5000000   0.2886751   2.0306218
```

❏ **Output file:** Now let us see the output file by command: `vi scf.out`. The total energy, as well as its decomposition into several terms, is obtained at the end of the `scf.out` as

QE-SSP/gr/scf/scf.out

```
!    total energy              =     -23.90991271 Ry
     Harris-Foulkes estimate   =     -23.90991328 Ry
     estimated scf accuracy    <       0.00000084 Ry
```

```
        The total energy is the sum of the following terms:

        one-electron contribution =     -90.80734321 Ry
        hartree contribution      =      47.24141117 Ry
        xc contribution           =      -8.30684749 Ry
        ewald contribution        =      27.96304915 Ry
        smearing contrib. (-TS)   =      -0.00018232 Ry

        convergence has been achieved in  13 iterations
```

If the readers want to see only the total energy from `scf.out`, the readers can use the `grep` command to find the lines containing the symbol `!` from `scf.out` as follows:

```
$ grep ! scf.out
```

The total energy is printed in the terminal as

```
$ !    total energy          =     -23.90991271 Ry
```

☞ **Explanation of scf.out:** This output file shows that the total energy of graphene is -23.90991271 Ry (1 Ry $= 13.60569301$ eV), and it is obtained in 13 SCF iterations. The total energy contains one-electron, Hartree, xc, ewald, and smearing contributions. These energy contributions are explained in Sec. 4.12. The smearing contribution is much small compared with other energy contributions. The magnitude of the total energy is not physically meaningful for a given cut-off energy and $\boldsymbol{k}$-points grid, but the convergence of the total energy with the related parameters is essential to determine the correct parameters. In the next tutorials, we discuss the total-energy value depending on the cut-off energy (Sec. 3.1.2) and the $\boldsymbol{k}$-points grid (Sec. 3.1.3).

Try It Yourself

1. Make `scf.in` file for the bulk Si with a face-centered-cubic structure (`ibrav = 2`) and visualize the Si structure by using XCrySDen.
2. Make `scf.in` file and calculate total energy of monolayer MoS_2.

3.1.2 Plane-wave cut-off energy

❑ **Purpose:** The controllable parameter to test the convergence is the cut-off energy related to how many plane waves are used in the calculation. The large numbers of plane waves give a better result through the memory and CPU time increase with increasing the cut-off energy. In this tutorial, we show how to select a suitable value of the cut-off energy.

❑ **Background:** The concept of plane-wave cut-off energy is given in Sec. 5.4. In Quantum ESPRESSO, the electronic wavefunction is represented by the linear combination of plane waves, where the maximum value of $\boldsymbol{G}_{\text{max}}$ is related to the cut-off energy $E_{\text{cut-off}} = \frac{\hbar^2}{2m}\boldsymbol{G}^2_{\text{max}}$ in Ry (1 Ry = 13.6 eV). We will change the value of the cut-off energy from 20 to 80 Ry to check the convergence of total energy in this tutorial.

❑ **How to run:** To run this tutorial, the readers should enter the following command lines:

```
1 $ cd ~/QE-SSP/gr/ecut/
2 $ ./run.sh &
```

- Line 1: Change directory to the `ecut` that includes the input files.
- Line 2: Run a bash script file `run.sh`, which generates and runs many jobs with changing the cut-off energies.

❑ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

The `ls` command displays a listing of the many input and output files including `run.sh` in the `ecut` directory as

```
calc-ecut.dat  ecut.30.in   ecut.50.out  ecut.75.in
ecut.20.in     ecut.30.out  ecut.55.in   ecut.75.out
ecut.20.out    ecut.35.in   ecut.55.out  ecut.80.in
ecut.22.in     ecut.35.out  ecut.60.in   ecut.80.out
ecut.22.out    ecut.40.in   ecut.60.out  plot-ecut.ipynb
ecut.24.in     ecut.40.out  ecut.65.in   run.sh
ecut.24.out    ecut.45.in   ecut.65.out
ecut.26.in     ecut.45.out  ecut.70.in
ecut.26.out    ecut.50.in   ecut.70.out
```

QE-SSP/gr/ecut/run.sh

```
#!/bin/bash
# Convergence test of cut-off energy.
# Set a variable ecut from 20 to 80 Ry.
for ecut in 20 22 24 26 30 35 40 45 50 55 60 65 \
70 75 80; do
# Make input file for the SCF calculation.
# ecutwfc is assigned by variable ecut.
cat > ecut.$ecut.in << EOF
&CONTROL
calculation   = 'scf'
pseudo_dir    = '../pseudo/'
outdir        = '../tmp/'
prefix        = 'gr'
/
&SYSTEM
ibrav         = 4
a             = 2.4623
c             = 10.0
nat           = 2
ntyp          = 1
occupations   = 'smearing'
smearing      = 'mv'
degauss       = 0.02
ecutwfc       = ${ecut}
/
&ELECTRONS
mixing_beta   = 0.7
conv_thr      = 1.0D-6
/
ATOMIC_SPECIES
C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
ATOMIC_POSITIONS (crystal)
C  0.333333333  0.666666666  0.500000000
C  0.666666666  0.333333333  0.500000000
K_POINTS (automatic)
12 12 1 0 0 0
EOF
# Run SCF calculation.
mpirun -np 4 pw.x <ecut.$ecut.in> ecut.$ecut.out
# Write cut-off and total energies in calc-ecut.dat.
awk '/!/ {printf"%d %s\n",'$ecut',$5}' ecut.$ecut.
    out >> calc-ecut.dat
# End of for loop
done
```

❑ **Output file:** The readers can open `calc-ecut.dat` generated by another pick-up-and-edit command `awk` in line 41 to make a plot of the total energy as a function of the cut-off energy.

```
$ vi calc-ecut.dat
```

QE-SSP/gr/ecut/calc-ecut.dat

```
20 -23.58222934
22 -23.74580589
24 -23.82861718
...
70 -23.91018738
75 -23.91023390
80 -23.91024577
```

- Column 1: Cut-off energy in units of Ry.
- Column 2: Total energy of graphene in units of Ry.

☞ **Plotting data from calc-ecut.dat:** To investigate how the total energy decreases with increasing the cut-off energy, we adopt Matplotlib, a plotting library for the Python programming language. Note that the readers can use any plotting software such as Gnuplot or Grace. Here, we would like to recommend the readers to install Matplotlib because it is synchronized for all examples in this book. For running Matplotlib, we make a JupyterLab `plot-ecut.ipynb` as an input file to automatically generate a plot from the extracted data in the output file. There is a detailed description of how to run a JupyterLab in Sec. 6.4. In order to run `plot-ecut.ipynb`, the readers can type as follows:

```
$ jupyter-lab plot-ecut.ipynb
```

QE-SSP/gr/ecut/plot-ecut.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
```

```
import numpy as np

# Open and read the file calc-ecut.dat
ecut, ener = np.loadtxt('calc-ecut.dat', delimiter='
     ', unpack=True)

# Create figure object
plt.figure()
# Plot the data, using scatter plot
plt.scatter(ecut, ener, s=150)
# Plot a dashed line at 80 Ry
plt.axhline(ener[14], c='gray', ls='--')
# Add the x and y-axis labels
plt.xlabel('Cut-off energy (Ry)')
plt.ylabel('Total energy (Ry)')
# Set the axis limits
plt.xlim(20, 80)
plt.ylim(-23.94, -23.56)
# Save the figure
plt.savefig('plot-ecut.pdf')
# Show the figure
plt.show()
```

By running `jupyter-lab`, we obtain the total-energy plot for the plane-wave cut-off energy as shown in Fig. 3.3. The total energy rapidly converges around 30 Ry, and it shows a best-converged value at 40 Ry. In principle, the result of the lowest total energy is obtained for infinity cut-off energy by a variational principle. However, the higher the cut-off energy leads to more time-consuming the calculation. Therefore, in practice, 40 Ry is good enough for this tutorial. The total energy difference between 40 Ry and 80 Ry is only 1.15 meV, which is much smaller than $k_B T = 25$ meV (at $T = 300$ K) or optical phonon energy (>100 meV). Since most quantities that can be computed using Quantum ESPRESSO critically depend on the cut-off energy, it is essential to perform this test when running Quantum ESPRESSO calculations. Note that, sometimes, the pseudopotential files will suggest cut-off energy. By referring to this value, we can select the cut-off energy. The suggested minimum cut-off energy is usually 40–60 Ry for ultrasoft pseudopotentials, while it is 80–120 Ry for norm-conserving pseudopotentials. The cut-off energy dependence of the type of pseudopotential is explained in Sec. 5.4.

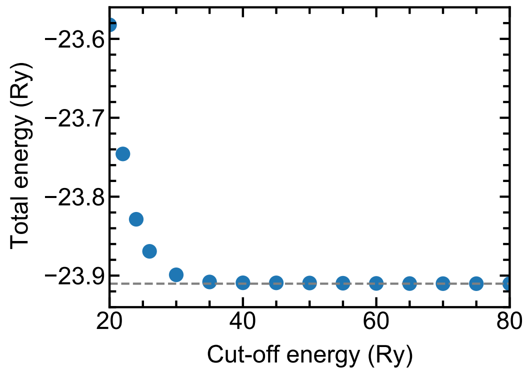


Figure 3.3 Total energy as a function of plane-wave cut-off energy. The dashed line is the total energy at 80 Ry. The cut-off energy shows a best-converged value at 40 Ry.

Try It Yourself

Check convergence value of cut-off energy for bulk Si and monolayer MoS_2.

3.1.3 *k*-points for Brillouin-zone integration

❑ **Purpose:** Other controllable parameter to test for convergence of the total energy is ***k***-points grid. The integration on ***k*** in the Brillouin zone (BZ) is approximated by summation of finite numbers of ***k***-points that is called ***k***-points grid. In this tutorial, we show how to select the ***k***-points grid.

❑ **Background:** The concept of ***k***-points in the BZ is given in Sec. 5.5. In Quantum ESPRESSO, there are six values to determine a ***k***-points grid, in which the first three values are set as the number of ***k***-points mesh of each axis, and the last three values are a parameter to move the lattice of each axis. The total number of ***k***-points is obtained by multiplying the first three values. For example, k_1, k_2, and k_3 are set to 10, 10, and 1, respectively, then the total number of ***k***-points is $10 \times 10 \times 1 = 100$. Table 3.3 shows the rule for selecting the ***k***-points grid based on the dimensions of the system.

Table 3.3 Selecting rule for ***k***-points grid based on dimension of system.

System	*k*-points grid
Three-dimensional (3D) system	$k_1 \times k_2 \times k_3$
Two-dimensional (2D) system in *xy*-plane	$k_1 \times k_2 \times 1$
One-dimensional (1D) system along *z*-direction	$1 \times 1 \times k_3$
Zero-dimensional (0D) system	$1 \times 1 \times 1$

Table 3.4 Selecting a rule for shifting (0 or 1) based on symmetry of system.

System	Shifting
Cubic, tetragonal, orthorhombic, and monoclinic systems	1
Hexagonal and trigonal systems	0
Rhombohedral and triclinic systems	1 or 0

The last three values should be set to either 0 or 1, where 0 indicates the default Gamma (Γ) position and 1 means that the ***k***-points grid is moved parallel, as shown in Fig. 3.4. Even though the ***k***-points are set with the same density by selecting the shift to move the ***k***-point mesh by selecting either 0 or 1, the number of inequivalent ***k***-points (red points in Fig. 3.4) can be reduced. The number of inequivalent ***k***-points depends on the crystal system of a cell. In Fig. 3.4 (a) and (b), we show a schematic diagram of the ***k***-point sampling in a cubic cell and hexagonal cell, respectively. In the cubic-cell case, the shifting decreases the number of inequivalent ***k***-points from 6 to 3 in the first BZ (see Fig. 3.4 (a)). However, in the case of a hexagonal cell, setting the shift breaks the hexagonal symmetry in the BZ. After the shift = 1 is set, symmetry operation is performed, and it needs more ***k***-points than the mesh with the case of shift = 0. Thus, the number of inequivalent ***k***-points increases due to the three-fold symmetry of the hexagonal cell, as shown in Fig. 3.4 (b). We thus should carefully select a shifting (0 or 1) depending on the symmetry of the system to calculate. Table 3.4 shows the selecting rule for shift based on the symmetry of the system.

Based on Tables 3.3 and 3.4, the ***k***-points grid for graphene is selected as "*k k* 1 0 0 0" with *k* is an integer from 4 to 14 in this tutorial.

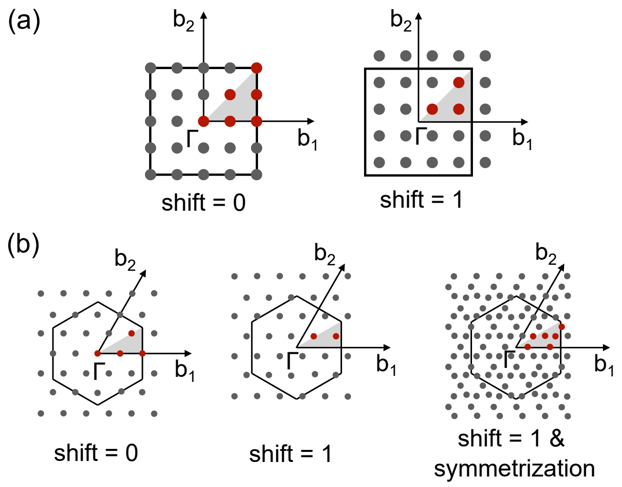


Figure 3.4 Schematic diagram of the ***k***-point sampling in a cubic cell (a) or a hexagonal cell (b). We also illustrate how to shift the ***k***-points by specifying "shift=1" and "symmetrization". Red points denote the inequivalent ***k***-points in the first Brillouin zone.

Next, we will learn how to perform the convergence test for this ***k***-points grid.

❏ **How to run:** To run this tutorial, the readers should type the following command lines:

```
$ cd ~/QE-SSP/gr/k-point/
$ ./run.sh &
```

- Line 1: Go to directory `k-point` that includes input files.
- Line 2: Run a bash script file `run.sh`, which consists of many jobs with changing the ***k***-points grid.

❏ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

The `ls` command displays a listing of the many output files in the `k-point` directory as

```
calc-kpoint.dat   kpoint.14.in    kpoint.7.out
kpoint.10.in      kpoint.14.out   kpoint.8.in
kpoint.10.out     kpoint.4.in     kpoint.8.out
kpoint.11.in      kpoint.4.out    kpoint.9.in
kpoint.11.out     kpoint.5.in     kpoint.9.out
kpoint.12.in      kpoint.5.out    plot-kpoint.ipynb
kpoint.12.out     kpoint.6.in     run.sh
kpoint.13.in      kpoint.6.out
kpoint.13.out     kpoint.7.in
```

❏ **Input file:** All input files are generated automatically by a bash script file `run.sh` as

QE-SSP/gr/k-point/run.sh

```bash
#!/bin/bash
# Convergence test of k-points grid.
# Set a variable k-point from 4 to 14.
for k in 4 5 6 7 8 9 10 11 12 13 14; do

# Make input file for the SCF calculation.
# k-points grid is assigned by variable n.
cat > kpoint.$k.in << EOF
&CONTROL
calculation  = 'scf'
pseudo_dir   = '../pseudo/'
outdir       = '../tmp/'
prefix       = 'gr'
/
&SYSTEM
ibrav        = 4
a            = 2.4623
c            = 10.0
nat          = 2
ntyp         = 1
occupations  = 'smearing'
smearing     = 'mv'
degauss      = 0.02
ecutwfc      = 40
/
&ELECTRONS
mixing_beta  = 0.7
conv_thr     = 1.0D-6
/
ATOMIC_SPECIES
C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
ATOMIC_POSITIONS (crystal)
C   0.333333333   0.666666666   0.500000000
C   0.666666666   0.333333333   0.500000000
```

```
35 K_POINTS (automatic)
36 ${k} ${k} 1 0 0 0
37 EOF
38
39 # Run pw.x for SCF calculation.
40 mpirun -np 4 pw.x <kpoint.$k.in>kpoint.$k.out
41 # Write the number of k-points (= k*k*1) and
42 # the total energy in calc-kpoint.dat
43 awk '/!/ {printf"%d %s\n",'$k*$k',$5}' kpoint.$k.out
         >> calc-kpoint.dat
44 # End of for loop.
45 done
```

☞ **Explanation of run.sh:** The ***k***-points grid is controlled with the card `K_POINTS` in line 35, wherein a ***k***-points grid needs to be set to determine how much density is necessary for the BZ integration. By selecting the `automatic` option in the card `K_POINTS`, it allows the ***k***-points grid through the Monkhorst-Pack method [Monkhorst and Pack (1976)].

❑ **Output file:** The number of ***k***-points ($= k \times k \times 1$) and the total energy are written in `calc-kpoint.dat`. The readers can open the `calc-kpoint.dat` file in the terminal by typing:

```
$ vi calc-kpoint.dat
```

QE-SSP/gr/ecut/calc-kpoint.dat

```
16 -23.91186965
25 -23.90766485
36 -23.90599780
...
144 -23.90913362
169 -23.90963897
196 -23.90953385
```

- Column 1: The number of ***k***-points.
- Column 2: Total energy of graphene in units of Ry.

☞ **Plotting data from calc-kpoint.dat:** The total energy as a function of the ***k***-point is plotted by running JupyterLab `plot-kpoint.ipynb`:

```
$ jupyter-lab plot-kpoint.ipynb
```

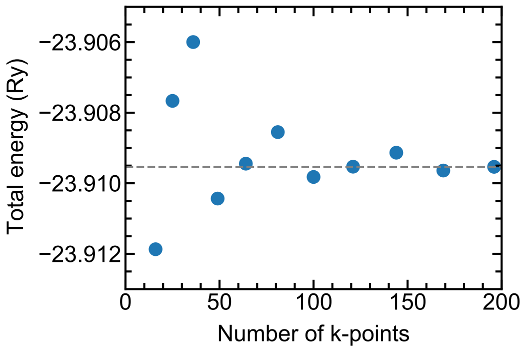


Figure 3.5 Total energy as a function of number of $\boldsymbol{k}$-points. It shows significant oscillations around the converged value. The dashed line is the total energy at $14 \times 14 \times 1 = 196$ $\boldsymbol{k}$-points. A $\boldsymbol{k}$-points grid of $10 \times 10 \times 1 = 100$ shows a good convergence.

QE-SSP/gr/k-point/plot-kpoint.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np

# Open and read the file calc-kpoint.dat
kp, ener = np.loadtxt('calc-kpoint.dat', delimiter='
     ', unpack=True)

# Create figure object
plt.figure()
# Plot the data, using black color
plt.scatter(kp, ener, s=150)
# Plot a dashed line at 14x14x1
plt.axhline(ener[10], c='gray', ls='--')
# Add the x and y-axis labels
plt.xlabel('Number of $\bm k$-points')
plt.ylabel('Total energy (Ry)')
# Set the axis limits
plt.xlim(0, 200)
plt.ylim(-23.913, -23.905)
# Save the figure
plt.savefig('plot-kpoint.pdf')
# Show the figure
plt.show()
```

In Fig. 3.5, we plot the total energy as a function of the total number of ***k***-points. Compared with the calculation of the cut-off energy convergence, the ***k***-points convergence does not occur monotonically, but it may show some oscillations around the converged value. Since a convergence of 1 meV or less for the entire system is our goal here, as shown in Sec. 3.1.2, a ***k***-points grid $k \times k \times 1 = 10 \times 10 \times 1 = 100$ is a reasonable choice.

Try It Yourself

1. Plot running-time, which can be found at the end of `scf.out` file, as a function of number of k-points for graphene.
2. Check convergence value of ***k***-points grid for bulk Si and monolayer MoS_2.

3.1.4 Optimizing atomic positions

❑ **Purpose:** In this tutorial, we are going to learn how to optimize atomic positions of the system.
❑ **Background:** In Quantum ESPRESSO, the default algorithm for structural optimization is the **BFGS algorithm**, which was proposed by Broyden [Broyden (1970)], Fletcher [Fletcher (1991)], Goldfarb [Goldfarb (1970)], and Shanno [Shanno (1970)], independently. The BFGS algorithm is one of the most powerful methods to solve unconstrained optimization problem. Let us consider the unconstrained optimization problem as

$$\min f(x), \quad x \in \mathbb{R}^n, \tag{3.1}$$

where $f(x)$ is a general function that has continuous second derivatives. The goal of the optimization problem is to find a stationary point x^* such that $\nabla f(x^*) = 0$. Given a starting point x_0 and an initial estimate of $H_0^{-1} \approx \nabla^2 f(x_0)$, the $(k+1)$-th iteration of

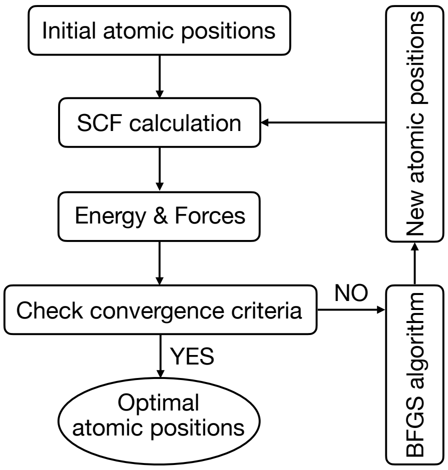


Figure 3.6 Flowchart of the optimizing atomic positions in Quantum ESPRESSO.

the BFGS algorithm is

$$\begin{aligned} x_{k+1} &= x_k - H_k^{-1}\nabla f(x_k), \\ s_k &= x_{k+1} - x_k, \\ y_k &= \nabla f(x_{k+1}) - \nabla f(x_k), \\ H_{k+1} &= H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \text{ with } k = 0, 1, 2, \ldots \end{aligned} \tag{3.2}$$

The initial matrix H_0^{-1} is usually taken to be a positive multiple of the identity matrix, which means that the initial search direction will be the steepest descent direction. The advantage of the BFGS algorithm is that it does not need to compute $\nabla^2 f(x_k)$ for each iteration, which can be time-consuming.

In Fig. 3.6, we show the flowchart of the optimizing atomic positions in Quantum ESPRESSO. At each step of the BFGS algorithm, the new positions of the atoms are obtained. Then the total energy and the forces are calculated by using the SCF method and the Hellmann-Feynman theorem, respectively. When the change of the total energy and all components of all forces are smaller than the convergence criteria, which are given in the input file, the optimized atomic positions are obtained. In Fig. 3.6, we show the flowchart of the optimizing atomic positions in Quantum ESPRESSO. At each step of the BFGS algorithm, the new positions of the atoms are obtained. Then the total energy and the forces are calculated by using the SCF method and the Hellmann-Feynman theorem, respectively. When the change of the total energy and all components of all forces are smaller than the convergence criteria, which are given in the input file, the optimized atomic positions are obtained.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/relax/
2 $ mpirun -np 4 pw.x <relax.in> relax.out &
```

- Line 1: Go to `relax` directory that includes input files.
- Line 2: Run `pw.x` by using in parallel with 4 processors (`-np 4`).

❑ **How to check:** The calculation will finish when `JOB DONE` is written at the end of the output file `relax.out`.

❑ **Input file:** The input file is showed in terminal by typing:

```
$ vi relax.in
```

QE-SSP/gr/relax/relax.in

```
&CONTROL
calculation   = 'relax'
pseudo_dir    = '../pseudo/'
outdir        = '../tmp/'
prefix        = 'gr'
etot_conv_thr = 1.0D-5
forc_conv_thr = 1.0D-4
/
&SYSTEM
ibrav         = 4
a             = 2.4623
c             = 10.0
nat           = 2
ntyp          = 1
occupations   = 'smearing'
smearing      = 'mv'
degauss       = 0.02
ecutwfc       = 60
/
```

```
20 &ELECTRONS
21 mixing_beta    = 0.7
22 conv_thr       = 1.0D-9
23 /
24 &IONS
25 ion_dynamics   = 'bfgs'
26 /
27 ATOMIC_SPECIES
28 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
29 ATOMIC_POSITIONS (crystal)
30 C  0.333333333  0.666666666  0.500000000  0  0  0
31 C  0.666666666  0.333333333  0.400000000
32 K_POINTS (automatic)
33 12 12 1 0 0 0
```

☞ **Explanation of relax.in:** The calculation of optimizing atomic position can be performed by selecting `calculation = 'relax'` (line 2) in the namelist `CONTROL`. The convergence criteria for the change of total energy and the forces are set by `etot_conv_thr` and `forc_conv_thr` (lines 6 and 7) in the namelist `CONTROL`, respectively. The BFGS algorithm is set in the namelist `IONS` as `ion_dynamics = 'bfgs'` (line 25). It is noted that the namelist `IONS` is the only mandatory addition. There are several variables that can be specified within this namelist, which control the algorithm used to find the optimized atomic positions. However, the readers can leave it empty if the readers are happy with selecting all defaults. The description of input variables is given in Tables 3.1, and 3.5. Here a.u. refers to the atomic unit (1 a.u. = 1 Hartree = 27.2 eV).

In order to observe the displacement of the carbon atoms in a unit cell of graphene, one carbon atom is fixed at a position `(0.333333333 0.666666666 0.500000000)` by adding `(0 0 0)`, and other carbon atom is set at position `(0.666666666 0.333333333 0.400000000)`. We expect that this carbon atom will move to the optimal position at `(0.333333333 0.666666666 0.500000000)`.

☞ **Note**: We can select the coordinates that will be fixed or not in relaxation by adding either `0` or `1`, respectively, in the atomic positions. For example, `(0 0 1)` means that the *x* and *y* coordinates are fixed, while *z* coordinate can be changed in structural optimization. If no option is specified, `(1 1 1)` is assumed.

Table 3.5 Meaning of input variables in `relax.in` file.

Line	Syntax	Meaning
6	`etot_conv_thr`	Convergence threshold on total energy (a.u.) for atomic minimization between two consecutive BFGS steps.
7	`forc_conv_thr`	Convergence threshold on forces (a.u.) for atomic minimization: the convergence criterion is satisfied when all components of all forces are smaller than this value.
24	`&IONS`	A namelist *must* be specified in the case of structural relaxation or molecular dynamics calculations.
25	`ion_dynamics`	Specify the type of atomic dynamics, in which `bfgs` (default) use BFGS quasi-newton algorithm.
26	`/`	End of namelist `IONS`.

❑ **Output file:** The optimized atomic positions are given in the output file `relax.out`. The readers can open output file in terminal by typing:

```
$ vi relax.out
```

To search forward for final coordinates, press `Esc` and then enter `/final`, vi editor will search the word `'final'` in `relax.out`.

QE-SSP/gr/relax/relax.out

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

  atom    1 type  1   force =     0.00000000    0.00000000   -0.00001508
  atom    2 type  1   force =     0.00000000    0.00000000    0.00001508

  Total force =     0.000015     Total SCF correction =     0.000001

  bfgs converged in   8 scf cycles and   7 bfgs steps
  (criteria: energy <  1.0E-05 Ry, force <  1.0E-04 Ry/Bohr)

  End of BFGS Geometry Optimization

  Final energy   =     -23.9099132232 Ry
Begin final coordinates

ATOMIC_POSITIONS (crystal)
C         0.3333333330       0.6666666660       0.5000000000    0   0   0
C         0.6666666660       0.3333333330       0.4999977356
```

```
   End final coordinates
```

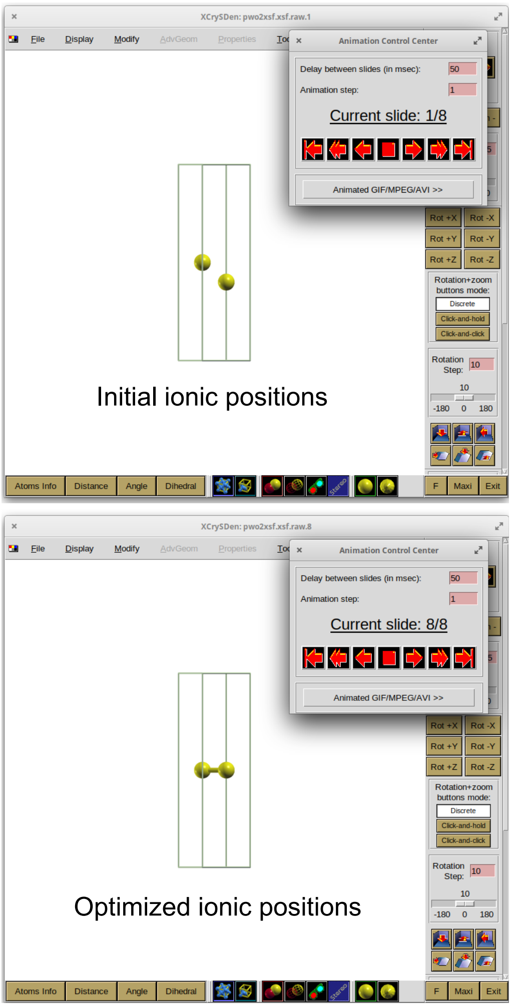


Figure 3.7 Visualizing optimized atomic positions from Quantum ESPRESSO output file by using XCrySDen. Top figure is the initial atomic positions (1/8 steps) and bottom figure is the optimized atomic positions (8/8 steps).

☞ **Explanation of relax.out:** This output file shows that the BFGS optimization is converged in 7 steps with the energy difference between two consecutive steps

$< 1.0 \times 10^{-5}$ Ry and the force $< 1.0 \times 10^{-4}$ Ry/Bohr. The second carbon atoms is moved from the initial position (`0.6666666660    0.3333333330    0.4000000000`) to the final position (`0.6666666660    0.3333333330    0.4999977356`).

☞ **Note**: The `Total force` in the output file is the square root of the sum of all the squared force components rather than the sum of the magnitudes of the individual forces on the atoms. If the `Total SCF correction` is large or comparable to the `Total force`, the output file will show a message "`SCF correction compared to forces is large: reduce conv_thr to get better values`". It usually means the readers need to try with a relatively smaller `conv_thr`.

☞ **Visualizing optimized ionic positions:** The readers can visualize the optimized ionic positions by using XCrySDen as

```
$ xcrysden &
```

Then the readers can go through the following steps: **File → Open PWscf → Open PWscf Output File** and select `relax.out`. In the box **[PWSCF Input: "relax.out"]**, we click on the **OK** button, then in the box **[Question]**, we select **Display All Coordinates as Animation** and click on the **Continue** button. The initial or optimized ionic positions can be showed by choosing **Current slide** in the box **[Animation Control Center]**, as shown in Fig. 3.7.

Try It Yourself

Run optimizing atomic positions for monolayer MoS_2 by changing position of only Mo atom.

3.1.5 Optimizing unit cell

❑ **Purpose:** In this tutorial, we show how to optimize the lattice vectors of the unit cell (or the variable-cell relaxation).

❑ **Background:** Similar to optimization the atomic positions, we can optimize the unit cell by using the BFGS algorithm.

❑ **How to run:** To run this tutorial, the readers should type as

```
1 $ cd ~/QE-SSP/gr/vc-relax/
2 $ mpirun -np 4 pw.x <vc-relax.in> vc-relax.out &
```

- Line 1: Go to `vc-relax` directory that includes input files.
- Line 2: Run `pw.x` by using in parallel calculation.

❑ **How to check:** The calculation will finish when `JOB DONE` is written at the end of the output file `vc-relax.out`.

❑ **Input file:** The readers can open the input file in terminal by typing:

```
$ vi vc-relax.in
```

QE-SSP/gr/vc-relax/vc-relax.in

```
&CONTROL
calculation   = 'vc-relax'
pseudo_dir    = '../pseudo/'
outdir        = '../tmp/'
prefix        = 'gr'
etot_conv_thr = 1.0D-5
forc_conv_thr = 1.0D-4
/
&SYSTEM
ibrav         = 4
a             = 2.5
c             = 15.0
nat           = 2
ntyp          = 1
occupations   = 'smearing'
smearing      = 'mv'
degauss       = 0.02
ecutwfc       = 60
/
&ELECTRONS
mixing_beta   = 0.7
conv_thr      = 1.0D-9
/
&IONS
ion_dynamics  = 'bfgs'
/
&CELL
cell_dynamics = 'bfgs'
press_conv_thr= 0.05
cell_dofree   = '2Dxy'
```

Table 3.6 Meaning of input variables in `vc-relax.in` file.

Line	Syntax	Meaning
27	`&CELL`	A namelist *must* be specified in the case of `calculation = 'vc-relax'`.
28	`cell_dynamics`	Specify the type of cell dynamics, in which `bfgs` (default) use BFGS quasi-newton algorithm.
29	`press_conv_thr`	Convergence threshold on the pressure for variable cell relaxation. Default value is 0.5 Kbar.
30	`cell_dofree`	Select the cell parameters to change, in which `'2Dxy'` is only x and y components are allowed to change.
31	`/`	End of namelist `CELL`.

```
31 /
32 ATOMIC_SPECIES
33 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
34 ATOMIC_POSITIONS (crystal)
35 C  0.333333333  0.666666666  0.500000000
36 C  0.666666666  0.333333333  0.500000000
37 K_POINTS (automatic)
38 12 12 1 0 0 0
```

☞ **Explanation of vc-relax.in:** The calculation of the optimizing unit cell can be performed by setting `calculation = 'vc-relax'` (line 2) in the namelist `CONTROL`. We need to specify both the namelist `IONS`, as shown in Sec. 3.1.4, and `CELL`. Note that the convergence criterion for optimizing unit cell is based on the pressure, which is set by `press_conv_thr` (line 29) in the namelist `CELL`. For the two-dimensional material as graphene, we also need to set `cell_dofree = '2Dxy'` (line 30) in the namelist `CELL`, that means that only *x* and *y* components are optimized. The description of input variables is given in Tables 3.1, 3.5, and 3.6.

❏ **Output file:** Both the optimized ionic positions and the lattice vectors are given in the output file `vc-relax.out`. The readers can open output file in terminal by typing:

```
$ vi vc-relax.out
```

To search forward for final coordinates, press `Esc` and then enter `/final`, vi editor will display as

QE-SSP/gr/vc-relax/vc-relax.out

```
     Computing stress (Cartesian axis) and pressure


     negative rho (up, down):  1.211E-04 0.000E+00
          total   stress  (Ry/bohr3)                (kbar)     P=       -0.13
   0.00000002   0.00000000  -0.00000000           0.00       0.00      -0.00
   0.00000000   0.00000002  -0.00000000           0.00       0.00      -0.00
   0.00000000   0.00000000  -0.00000259           0.00       0.00      -0.38


     bfgs converged in   5 scf cycles and   4 bfgs steps
     (criteria: energy<1.0E-05 Ry, force<1.0E-04Ry/Bohr, cell<5.0E-02kbar)

     End of BFGS Geometry Optimization

     Final enthalpy =     -23.9099689524 Ry
Begin final coordinates
     new unit-cell volume =    532.18989 a.u.^3 (    78.86240 Ang^3 )
     density =      0.50580 g/cm^3

CELL_PARAMETERS (alat=  4.72431533)
   0.985562233  -0.000000000   0.000000000
  -0.492781117   0.853521931   0.000000000
   0.000000000   0.000000000   6.000000000

ATOMIC_POSITIONS (crystal)
C             0.3333333330        0.6666666660        0.5000000000
C             0.6666666660        0.3333333330        0.5000000000
End final coordinates
```

☞ **Explanation of vc-relax.out:** This output file shows that the optimized structure (i.e., both ionic positions and lattice constants are optimized) with the convergence criteria of the energy difference $< 1.0 \times 10^{-5}$ Ry, the total forces of each atom $< 1.0 \times 10^{-4}$ Ry/Bohr and the in-plane pressures of unit cell $< 5.0 \times 10^{-2}$ kbar. Since we optimize the two-dimensional structure, the absolute value of the pressure in the z direction can be larger than the convergence threshold (5.0×10^{-2} kbar). To reduce the absolute value of the pressure in the z direction, the lattice constant in the z direction should be large enough. Here, we choose $c = 15.0$ Å as shown in the input file.

The optimized lattice vectors are given in `CELL_PARAMETERS` in the block of the `final coordinates`. Note that the lattice vectors are given explicitly in Bohr (1 Bohr = 0.529177211 Å) along with a scaling factor `alat = 4.72431533`, which is converted from a = `2.5` Å in the input file. The optimized lattice constant is a = `2.5×0.985562233 = 2.4639055825` Å, which is consistent with experimental value ($a \sim 2.46$ Å) [Ohta *et al.* (2006)]. Note that the $c = 15$ Å constant does not change because of setting `cell_dofree = '2Dxy'`.

Try It Yourself

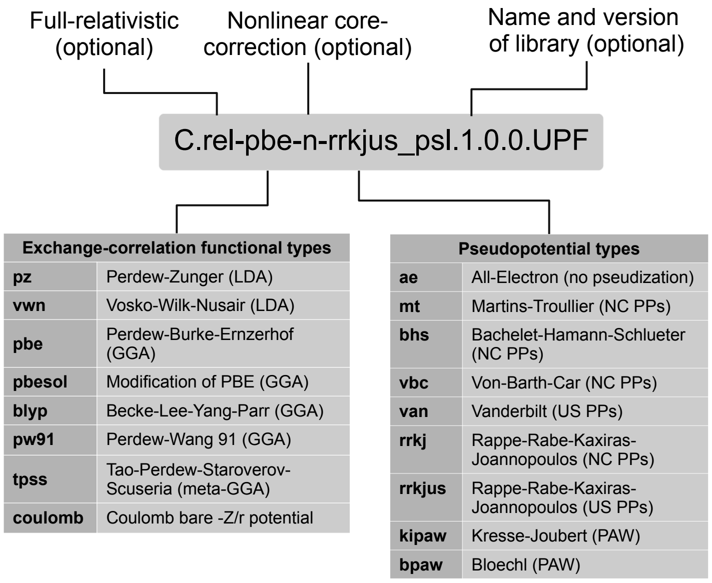
Run optimizing unit cell for bulk Si and monolayer MoS_2. It is noted that `cell_dofree` is set to `all` for the bulk Si.

3.1.6 Selecting pseudopotential

❑ **Purpose:** In this tutorial, we learn how to select the pseudopotential in the input file of Quantum ESPRESSO.
❑ **Background:** As discussed in Sec. 5.4, the pseudopotential (PP) is an approximation of atomic potentials to minimize the number of plane wave basis, and PP contains various information for simulating a system. The PP in Quantum ESPRESSO uses a unified PP format (UPF).[2] Three types of PPs, including norm-conserving (NC) PPs, ultra-soft (US) PPs, and projector-augmented wave (PAW), are supported. Some calculations (e.g., Γ-point phonon, third-order energy derivatives, non-resonance Raman, anharmonic force constants) work only with the NC PPs because the NC PPs provide wavefunctions that we can use in the integration for calculating some physical properties. Thus, the readers should check what type of PP is supported and select a proper PP for each atom before calculating.

The main library of the PPs is PSlibrary, which includes the scalar relativistic and fully relativistic US PPs and PAW data sets for many elements. The readers can download the PPs from PSlibrary in this link: `http://pseudopotentials.quantum-espresso.org/legacy_tables/ps-library`. In Fig. 3.8, we show the naming convention for the PP file. A PP file includes not only the type of pseudopotential (NC, US, or PAW) but also the type of the exchange-correlation functional (LDA, GGA, or meta-GGA). The detail

[2] Unified pseudopotential format is the Extensible Markup Language (XML)-like structure to store pseudopotentials.


Full-relativistic (optional)
Nonlinear core-correction (optional)
Name and version of library (optional)
C.rel-pbe-n-rrkjus_psl.1.0.0.UPF

Exchange-correlation functional types	
pz	Perdew-Zunger (LDA)
vwn	Vosko-Wilk-Nusair (LDA)
pbe	Perdew-Burke-Ernzerhof (GGA)
pbesol	Modification of PBE (GGA)
blyp	Becke-Lee-Yang-Parr (GGA)
pw91	Perdew-Wang 91 (GGA)
tpss	Tao-Perdew-Staroverov-Scuseria (meta-GGA)
coulomb	Coulomb bare -Z/r potential

Pseudopotential types	
ae	All-Electron (no pseudization)
mt	Martins-Troullier (NC PPs)
bhs	Bachelet-Hamann-Schlueter (NC PPs)
vbc	Von-Barth-Car (NC PPs)
van	Vanderbilt (US PPs)
rrkj	Rappe-Rabe-Kaxiras-Joannopoulos (NC PPs)
rrkjus	Rappe-Rabe-Kaxiras-Joannopoulos (US PPs)
kipaw	Kresse-Joubert (PAW)
bpaw	Bloechl (PAW)

Figure 3.8 Naming convention for the pseudopotential from PSlibrary in unified pseudopotentials format (UPF).

of the exchange-correlation functional is given in Sec. 4.7. Besides PSlibrary, other libraries are also available as

- **Standard Solid-State PPs (SSSP)**: a collection of the best verified PPs, maintained by THEOS and MARVEL. Available on the Materials Cloud web site is `https://www.materialscloud.org/discover/sssp/table/efficiency`.
- **Pseudo Dojo**: a complete set of PPs and PAW sets. Available on the web site is `http://www.pseudo-dojo.org/index.html`.
- **GBRV high-throughput PPs**: a highly accurate and computationally inexpensive open-source US PP library. Available on the web site is `http://www.physics.rutgers.edu/gbrv`.
- **Optimized NC Vanderbilt PP (ONCVPSP)**: an accurate and inexpensive NC PPs by D. R. Hamann. Available on the web site is `http://www.quantum-simulation.org/potentials/sg15_oncv/upf/`.
- **Hartwigesen-Goedecker-Hutter PPs**: a collection of NC PPs from CPMD, by Matthias Krack. Available on the web site is

`http://pseudopotentials.quantum-espresso.org/legacy_tables/hartwigesen-goedecker-hutter-pp`.

A recommended way of selecting a good PP is based on testing and comparing the preliminary results to experimental data. Note that when comparing Quantum ESPRESSO calculation results, you should not directly compare the energy values by using different PPs. Since each PP has a different option or condition, using a different type of PP results in different energy values, which does not have any meaning. It is recommended to compare the properties such as lattice constant, energy band gap, etc., with experiments for observed materials. Since we showed how to obtain the lattice constant of graphene in Sec. 3.1.5, in this tutorial, we will compare the lattice constant for several PPs.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 | $ cd ~/QE-SSP/gr/pps/
2 | $ ./run.sh &
```

- Line 1: Go to pps directory that includes input files.
- Line 2: Run a bash script file `run.sh`.

❑ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

The `ls` displays a listing of the all files in the pps folder as

```
run.sh
vc-relax.C.pbe-hgh.UPF.in
vc-relax.C.pbe-hgh.UPF.out
vc-relax.C.pbe-n-kjpaw_psl.1.0.0.UPF.in
vc-relax.C.pbe-n-kjpaw_psl.1.0.0.UPF.out
vc-relax.C.pbe-n-rrkjus_psl.0.1.UPF.in
vc-relax.C.pbe-n-rrkjus_psl.0.1.UPF.out
vc-relax.C.pz-hgh.UPF.in
vc-relax.C.pz-hgh.UPF.out
vc-relax.C.pz-n-kjpaw_psl.0.1.UPF.in
vc-relax.C.pz-n-kjpaw_psl.0.1.UPF.out
vc-relax.C.pz-n-rrkjus_psl.0.1.UPF.in
vc-relax.C.pz-n-rrkjus_psl.0.1.UPF.out
```

❏ **Input file:** All input files are generated automatically by a bash script file `run.sh` as

QE-SSP/gr/pps/run.sh

```
#!/bin/bash
# Set a variable pp for 6 PPs.
for pp in C.pbe-hgh.UPF \
C.pz-hgh.UPF \
C.pbe-n-rrkjus_psl.0.1.UPF \
C.pz-n-rrkjus_psl.0.1.UPF \
C.pbe-n-kjpaw_psl.1.0.0.UPF \
C.pz-n-kjpaw_psl.0.1.UPF; do
# Make input file for the vc-relax calculation.
cat > vc-relax.$pp.in << EOF
&CONTROL
calculation    = 'vc-relax'
pseudo_dir     = '../pseudo/'
outdir         = '../tmp/'
prefix         = 'gr'
etot_conv_thr = 1.0D-5
forc_conv_thr = 1.0D-4
/
&SYSTEM
ibrav          = 4
a              = 2.4639055825
c              = 15.0
nat            = 2
ntyp           = 1
occupations    = 'smearing'
smearing       = 'mv'
degauss        = 0.02
ecutwfc        = 80
/
&ELECTRONS
mixing_beta    = 0.7
conv_thr       = 1.0D-9
/
&IONS
ion_dynamics   = 'bfgs'
/
&CELL
cell_dynamics = 'bfgs'
press_conv_thr= 0.05
cell_dofree    = '2Dxy'
/
ATOMIC_SPECIES
C 12.0107 $pp
ATOMIC_POSITIONS (crystal)
```

Table 3.7 Calculated lattice constants (a) by 6 pseudopotentials.

Pseudopotential	Type	a (Å)
`C.pbe-hgh.UPF`	NC + GGA	2.4529
`C.pz-hgh.UPF`	NC + LDA	2.4321
`C.pbe-n-rrkjus_psl.0.1.UPF`	US + GGA	2.4643
`C.pz-n-rrkjus_psl.0.1.UPF`	US + LDA	2.4456
`C.pbe-n-kjpaw_psl.1.0.0.UPF`	PAW + GGA	2.4673
`C.pz-n-kjpaw_psl.0.1.UPF`	PAW + LDA	2.4456

```
45 C   0.333333333  0.666666666  0.500000000
46 C   0.666666666  0.333333333  0.500000000
47 K_POINTS (automatic)
48 12 12 1 0 0 0
49 EOF
50 # Run pw.x for vc-relax calculation.
51 mpirun -np 4 pw.x <vc-relax.$pp.in> vc-relax.$pp.out
52 # End of for loop.
53 done
```

☞ **Explanation of run.sh:** Six PP files are adopted in this tutorial. The folder path containing these PP files is set in syntax `pseudo_dir` (line 13) the namelist `CONTROL`, and the filename of PP is set as `$pp` in the namelist `ATOMIC_SPECIES` (line 43).

❏ **Output file:** The optimized lattice constants are obtained from the output files `vc-relax.*.out` (see Sec. 3.1.5) and listed in Table 3.7. Compared with the measured lattice constant of graphene ($\sim$ 2.46 Å [Ohta *et al.* (2006)]), the calculated lattice constant of `C.pbe-n-rrkjus_psl.0.1.UPF` (US PP + GGA) is in good agreement. The PP with GGA often overestimates the lattice constants of solids, while LDA almost always underestimates the lattice constants. In both cases, the typical errors amount to 1–2% of the lattice parameters.

Try It Yourself

Calculate lattice constants of bulk Si for several PP files, and compare the obtained results with the experimental value (5.431 Å).

3.1.7 Selecting smearing function and energy

❑ **Purpose:** Smearing is a concept for suppressing unstable electron density during the iteration in the calculation of metal. In this tutorial, we learn how to select the smearing function and energy.

❑ **Background:** Let us briefly explain the concept of smearing. In the DFT, the electron density $n(\boldsymbol{r})$ is calculated from the Kohn-Sham wavefuntion $\phi(\boldsymbol{r})$ as

$$n(\boldsymbol{r}) = \sum_i f_i |\phi_i(\boldsymbol{r})|^2, \tag{3.3}$$

where the index i runs over all states and f_i are the occupation numbers, which can be either 1 or 0 depending on that the i-th state is occupied or not, respectively, as shown in Fig. 3.9 (a). In Quantum ESPRESSO, the sum of Eq. (3.3) is replaced by an integration over the Brillouin zone (BZ) of the system as

$$n(\boldsymbol{r}) = \sum_i \int_{\mathrm{BZ}} f_{i\boldsymbol{k}} |\phi_{i\boldsymbol{k}}(\boldsymbol{r})|^2 d\boldsymbol{k}, \tag{3.4}$$

where $\boldsymbol{k}$ is the wavevector in the BZ. In practice, this integration is carried out numerically by summing over a finite set of $\boldsymbol{k}$-points. For the semiconductor or insulator, electrons in the valence band can not occupy the conduction band during the calculation since the energies of the valence and conduction bands are sufficiently separated. Therefore, the electron density and derived quantities (e.g., the total energy) converge quickly with the finite number of $\boldsymbol{k}$-points used in the integration. However, in the case of the metal or semimetal, the valence bands that cross the Fermi level are partially occupied. Thus, the electrons may occupy the unoccupied states during one iteration, making the algorithm unstable. In this case, it

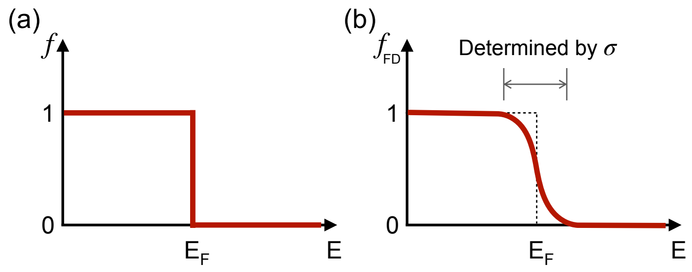


Figure 3.9 (a) The step function f and (b) the smooth Fermi-Dirac function f_{FD} are plotted as a function of the energy E. The width of the Fermi-Dirac function is determined by σ.

often needs a large number of $\boldsymbol{k}$-points in order to make calculations converge. To stabilize the algorithm and reduce the number of $\boldsymbol{k}$-points, we employ a smearing function, in which $f_{i\boldsymbol{k}}$ is replaced by a function that varies smoothly from 1 to 0 at near the Fermi level. By smoothly changing f_i as a function of energy, we can suppress the unstable change of electron density for each SCF iteration.

The Fermi-Dirac smearing: A simple smearing function is the Fermi-Dirac-like distribution function, which is defined by [Mermin (1965)]

$$f_{\mathrm{FD}} = \frac{1}{\exp(x)+1}, \text{ with } x = \frac{E_{i\boldsymbol{k}} - E_{\mathrm{F}}}{\sigma}, \tag{3.5}$$

where E_{F} is the Fermi energy, and σ is the smearing energy, analogous to $k_{\mathrm{B}}T$ in the Fermi-Dirac statistics. Note that when $\sigma \to 0$, f approaches the step function. As shown in Fig. 3.9 (b), f is a smooth function. Thus, Eq. (3.4) does not cause problems during calculation. However, this approach has a problem. In order to calculate with a relatively small number of $\boldsymbol{k}$-points sampling, we need to use a very large σ value about $0.1 - 0.5$ eV, corresponding to thousands of degrees of the temperature, and the distribution has long tails. Thus, we need to calculate a relatively large number of unoccupied states compared with slowly decay to zero occupation.

The Gaussian smearing: Another smearing function is the Gaussian smearing, in which the step function is approximated by the (Gaussian) complementary error function (erfc), which is defined by [Fu and Ho (1983)]

$$f_{\mathrm{G}} = \frac{1}{2}\mathrm{erfc}(x), \tag{3.6}$$

where erfc(x) is defined as $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-t^2)dt$. As shown in Fig. 3.10, the width of the Gaussian smearing is smaller than that of the Fermi-Dirac smearing. This means that in order to get similar $\boldsymbol{k}$-points convergence as for the Fermi-Dirac smearing, the Gaussian smearing can use a smaller value of σ.

The Methfessel-Paxton smearing: Methfessel and Paxton proposed a more sophisticated approximation to the step function based on the Gaussian smearing [Methfessel and Paxton (1989)]. The first-order Methfessel-Paxton approximation is expressed as

$$f_{\mathrm{MP}} = \frac{1}{2}\mathrm{erfc}(x) - \frac{1}{2\sqrt{\pi}} x \exp\left(-x^2\right), \tag{3.7}$$

where the first term on the right-hand side corresponds to the Gaussian smearing, while the second term serves to correct the error introduced by the Gaussian smearing. Although only fewer $\boldsymbol{k}$-points are needed for the Methfessel-Paxton smearing, the occupation numbers become lower than zero, or larger than 1, as shown in Fig. 3.10. Note that while occupation numbers that are larger than 1 are less worrisome, negative occupation numbers can be conceptually problematic in the Methfessel-Paxton smearing.

The Marzari-Vanderbilt smearing: The Marzari-Vanderbilt approach has been proposed in order to avoid the negative occupancies introduced by the Methfessel-Paxton smearing. In this case, the step function is approximated by a Gaussian function multiplied by a first-order polynomial as [Marzari *et al.* (1999)]

$$f_{\mathrm{MV}} = \frac{1}{2}\left[\sqrt{\frac{2}{\pi}} \exp(-x^2 - \sqrt{2}x - 1/2) + \mathrm{erfc}\left(x + \frac{1}{\sqrt{2}}\right)\right]. \tag{3.8}$$

As shown in Fig. 3.10, the Marzari-Vanderbilt smearing is asymmetric but does not have negative values and thus the problem with negative electron density is safety avoided.

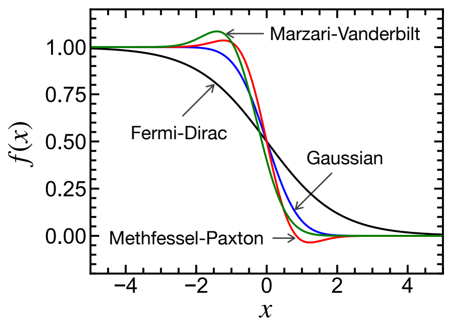


Figure 3.10 The several smearing functions $f(x)$ are plotted as a function of $x = (E_{ik} - E_F)/\sigma$.

In this tutorial, we plot the total energy as function of the smearing energy σ by using the Fermi-Dirac, Gaussian, Methfessel-Paxton, and Marzari-Vanderbilt smearing functions.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
$ cd ~/QE-SSP/gr/smearing/
$ ./run.sh &
```

- Line 1: Go to `smearing` directory that includes input files.
- Line 2: Run a bash script file `run.sh`, which consists of many jobs with changing both the smearing function and smearing energy.

❑ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

The `ls` command displays a listing of the all files in the `smearing` folder as

```
calc-smearing.dat   gauss.0.020.in    mp.0.035.out
fd.0.005.in         gauss.0.020.out   mp.0.040.in
fd.0.005.out        gauss.0.025.in    mp.0.040.out
fd.0.010.in         gauss.0.025.out   mv.0.005.in
fd.0.010.out        gauss.0.030.in    mv.0.005.out
fd.0.015.in         gauss.0.030.out   mv.0.010.in
```

```
   fd.0.015.out        gauss.0.035.in    mv.0.010.out
   fd.0.020.in         gauss.0.035.out   mv.0.015.in
   fd.0.020.out        gauss.0.040.in    mv.0.015.out
   fd.0.025.in         gauss.0.040.out   mv.0.020.in
   fd.0.025.out        mp.0.005.in       mv.0.020.out
   fd.0.030.in         mp.0.005.out      mv.0.025.in
   fd.0.030.out        mp.0.010.in       mv.0.025.out
   fd.0.035.in         mp.0.010.out      mv.0.030.in
   fd.0.035.out        mp.0.015.in       mv.0.030.out
   fd.0.040.in         mp.0.015.out      mv.0.035.in
   fd.0.040.out        mp.0.020.in       mv.0.035.out
   gauss.0.005.in      mp.0.020.out      mv.0.040.in
   gauss.0.005.out     mp.0.025.in       mv.0.040.out
   gauss.0.010.in      mp.0.025.out      plot-smearing.ipynb
   gauss.0.010.out     mp.0.030.in       run.sh
   gauss.0.015.in      mp.0.030.out
   gauss.0.015.out     mp.0.035.in
```

❏ **Input file:** All input files are generated automatically by a bash script file `run.sh` as

QE-SSP/gr/smearing/run.sh

```
#!/bin/bash
# Set a variable sf for the smearing functions.
for sf in fd gauss mp mv; do

# Set a variable se for the smearing energies.
for se in 0.005 0.010 0.015 0.020 0.025 0.030 \
0.035 0.040; do

# Make input file for the scf calculation.
cat > $sf.$se.in << EOF
&CONTROL
calculation    = 'scf'
pseudo_dir     = '../pseudo/'
outdir         = '../tmp/'
prefix         = 'gr'
/
&SYSTEM
ibrav          = 4
a              = 2.4639055825
c              = 15.0
nat            = 2
ntyp           = 1
occupations    = 'smearing'
smearing       = '$sf'
degauss        = $se
ecutwfc        = 40
/
&ELECTRONS
```

```
29 mixing_beta    = 0.7
30 conv_thr       = 1.0D-6
31 /
32 ATOMIC_SPECIES
33 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
34 ATOMIC_POSITIONS (crystal)
35 C  0.333333333  0.666666666  0.500000000
36 C  0.666666666  0.333333333  0.500000000
37 K_POINTS (automatic)
38 12 12 1 0 0 0
39 EOF
40
41 # Run pw.x for SCF calculation.
42 mpirun -np 4 pw.x <$sf.$se.in> $sf.$se.out
43
44 # Write the name of the smearing functions, the
       smearing energies, and total energies
45 awk -v var="$sf" '/!/ {printf"%-6s %1.3f %s\n",var,'
       $se',$5}' $sf.$se.out >> calc-smearing.dat
46 # End of for sf loop.
47 done
48 # End of for se loop.
49 done
```

☞ **Explanation of run.sh:** The smearing option is set by `occupations = 'smearing'` (line 23) in the namelist SYSTEM. The smearing function and energy σ in Ry are set in the syntaxes `occupations` (line 24) and `degauss` (line 25), respectively. The Fermi-Dirac, Gaussian, Methfessel-Paxton, and Marzari-Vanderbilt smearing can selected by setting `smearing = 'fd'`, `smearing = 'gauss'`, `smearing = 'mp'`, and `smearing = 'mv'`, respectively.

☞ **Note:** For the metal and semimetal materials such as graphene, the readers should select the smearing option, otherwise the program might be stopped with the error as

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Error in routine electrons (1):
    charge is wrong: smearing is needed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

❑ **Output file:** The smearing functions, the smearing energies, and the total energy are written in `calc-smearing.dat`. The readers can open this file in terminal by typing:

```
$ vi calc-smearing.dat
```

QE-SSP/gr/smearing/calc-smearing.dat

```
fd      0.005  -23.90915383
fd      0.010  -23.90934651
fd      0.015  -23.90954805
...
mv      0.030  -23.90923457
mv      0.035  -23.90927836
mv      0.040  -23.90931451
```

- Column 1: The abbreviation of the smearing function.
- Column 2: The smearing energy in units of Ry.
- Column 3: The total energy in units of Ry.

☞ **Plotting data from calc-smearing.dat:** The total energy as a function of the smearing energy is plotted by running JupyterLab `plot-smearing.ipynb`:

```
$ jupyter-lab plot-smearing.ipynb
```

QE-SSP/gr/smearing/plot-smearing.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')

# Open and read the file calc-smearing.dat
f = open('calc-smearing.dat', 'r')
f_smearing = [line for line in f.readlines() if line
    .strip()]
f.close()
nsf = 4 # Number of the smearing functions
nse = 8 # Number of the smearing energies
# Read the smearing energy (se) and total energy (
    ener) for each smearing function
se = []
ener = []
for i in range(nsf):
    se.append([])
    ener.append([])
    for j in range(nse):
        tmp1 = f_smearing[i*nse+j].split()[1]
        tmp2 = f_smearing[i*nse+j].split()[2]
```

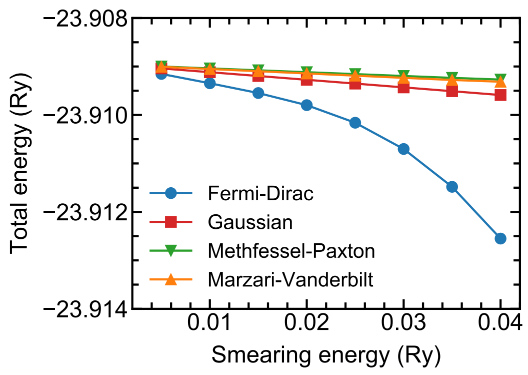


Figure 3.11 Total energy as a function of smearing energy by using the Fermi-Dirac, Gaussian, Methfessel-Paxton, and Marzari-Vanderbilt smearing functions.

```
        se[i].append(float(tmp1))
        ener[i].append(float(tmp2))

# Create figure object
plt.figure()
# Plot the data
plt.plot(se[0],ener[0],'o-', label='Fermi-Dirac')
plt.plot(se[1],ener[1],'s-', label='Gaussian')
plt.plot(se[2],ener[2],'v-', label='Methfessel-Paxton')
plt.plot(se[3],ener[3],'^-', label='Marzari-Vanderbilt')
# Add the legend
plt.legend(loc='lower left')
# Add the x and y-axis labels
plt.xlabel('Smearing energy (Ry)')
plt.ylabel('Total energy (Ry)')
# Set the axis limits
plt.xlim(0.003, 0.042)
plt.ylim(-23.914, -23.908)
# Save the figure.
plt.savefig('plot-smearing.pdf')
# Show the figure
plt.show()
```

In Fig. 3.11, we show the total energy of graphene as a function of smearing energy by using the Fermi-Dirac, Gaussian, Methfessel-Paxton, and Marzari-Vanderbilt smearing functions. We can see that the total energy decreases significantly as the smearing energy increases for the Fermi-Dirac function. In contrast, for the Methfessel-Paxton (MP) or Marzari-Vanderbilt (MV) functions, the decrease is not significant, and therefore it is possible to use relatively large smearing energy for the MP or MV functions. Note that the smearing energy is to treat the Fermi surface efficiently, and we should use small smearing energy from 0.01 to 0.02 Ry for the MP and MV functions. Thus the selection of the MV smearing is reasonable for graphene, and we will use this smearing for the next tutorials of graphene.

Try It Yourself

Plot total energy as a function of smearing energy with different smearing functions for monolayer MoS_2.

3.2 Electronic properties

In Sec. 3.1, we show how to converge our calculations with respect to the cut-off energy and the sampled $\boldsymbol{k}$-points grid. By using these parameters, the unit cell is optimized with a specific pseudopotential and smearing function. This section shows how to visualize and calculate charge density, band structure, and density of state of an optimized unit cell.

3.2.1 Charge density

❑ **Purpose:** In this tutorial we visually check the electron charge density in space.
❑ **Background:** By running `pw.x` in the SCF calculation as shown in Sec. 3.1.1, you can find `tmp/gr.save` directory containing the charge density file (`charge-density.dat`) and other output files. Now in order to do a post-process for converting the file format of `charge-density.dat` into another file format that we can visualize, we use the executable `pp.x` inside the Quantum ESPRESSO distribution. This code can read the output files produced by `pw.x`

and convert them into files compatible with various visualization programs such as XCrySDen or VESTA.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/rho/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pp.x < pp.in > pp.out &
```

- Line 1: Go to `rho` directory.
- Line 2: Run `pw.x` with the input file `scf.in` to obtain `scf.out`.
- Line 3: Run `pp.x` with the input file `pp.in` to obtain `pp.out`.

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of both output files `scf.out` and `pp.out`.

❑ **Input file:** The detail of the `scf.in` file is given in Sec. 3.1.1. For other input file `pp.in`, the readers can use `vi` editor by typing:

```
$ vi pp.in
```

QE-SSP/gr/rho/pp.in

```
&INPUTPP
outdir        = '../tmp/'
prefix        = 'gr'
plot_num      = 0
/
&PLOT
iflag         = 3
output_format= 6
fileout       = 'gr_rho.cube'
nx            = 64
ny            = 64
nz            = 12
/
```

The description of input variables of `pp.in` is given in Table 3.8.

☞ **Note:** `outdir` and `prefix` must be the same as for `scf.in` for `pw.x` calculation.

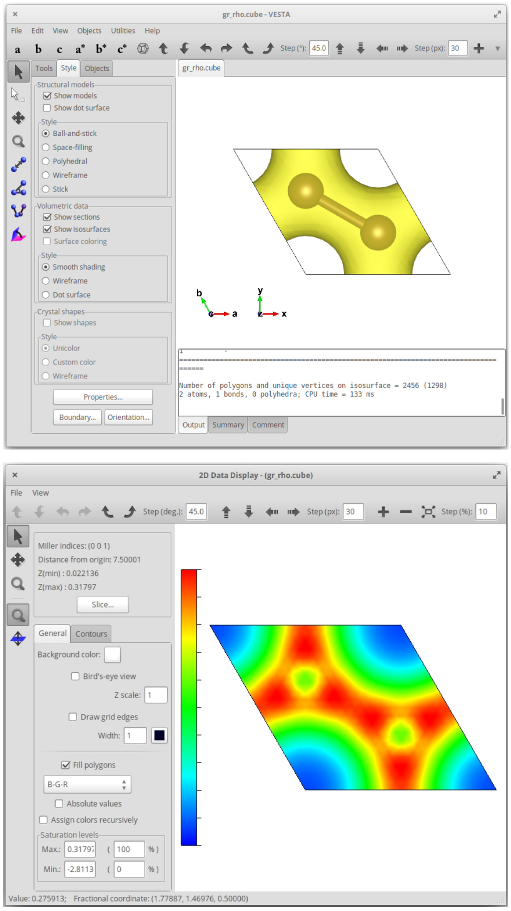
☞ **Visualizing charge density:** To see the electron charge density, we launch VESTA and load the file `gr_rho.cube` by typing:

Table 3.8 Meaning of input variables in `pp.in` file.

Line	Syntax	Meaning
1	`&INPUTPP`	A *mandatory namelist* includes input variables, which read the output produced by `pw.x` and extract or calculate the desired quantities (rho, V, ...).
2	`outdir`	Directory includes input data, i.e. the same as in `pw.x`.
3	`prefix`	Prefix of files saved by program `pw.x`.
4	`plot_num`	Select the output quantities, in which 0 = electron charge density.
5	`/`	End of namelist `INPUTPP`.
6	`&PLOT`	A *mandatory namelist* includes input variables, which provide the desired quantity for outputting file in a suitable format for various plotting programs.
7	`iflag`	Type of plot, in which 3 = 3D plot.
8	`output_format`	Type of format, in which 6 = Gaussian cube file, which can be read by VESTA software.
9	`fileout`	Name of the file to which the plot is written.
10-12	`nx,ny,nz`	To determine the 3D grid for `iflag = 3`.
13	`/`	End of namelist `PLOT`.

```
$ VESTA gr_rho.cube
```

VESTA will automatically identify the 3D charge density of graphene as shown in Fig. 3.12 top. The readers might want to look at the charge density as a 2D contour plot or heat map. From VESTA, we go to **Utilities → 2D Data Display...**, then in the box **[2D Data Display - (gr_rho.cube)]**, we click on the **Slice** button to put the values for Miller indices, then click to **Ok** button. We will obtain the 2D plot of the charge density (see Fig. 3.12 bottom).



Structural models: [✓ Show models], Show dot surface
Style: [✓ Ball-and-stick], Space-filling, Polyhedral, Wireframe, Stick
Volumetric data: [✓ Show sections], [✓ Show isosurfaces], Surface coloring
Style: [✓ Smooth shading], Wireframe, Dot surface
Crystal shapes: Show shapes
Style: [✓ Unicolor], Custom color, Wireframe

```
================================================================================
======

Number of polygons and unique vertices on isosurface = 2456 (1298)
2 atoms, 1 bonds, 0 polyhedra; CPU time = 133 ms
```

Miller indices: (0 0 1)
Distance from origin: 7.50001
Z(min) : 0.022136
Z(max) : 0.31797

Background color: ☐
Bird's-eye view ☐
Z scale: 1
Draw grid edges ☐
Width: 1
Fill polygons ✓
B-G-R
Absolute values ☐
Assign colors recursively ☐
Saturation levels
Max.: 0.31797 (100 %)
Min.: -2.8113 (0 %)

Value: 0.275913; Fractional coordinate: (1.77887, 1.46976, 0.50000)

Figure 3.12 3D (top) and 2D (bottom) visualizing charge density of graphene by using VESTA.

Try It Yourself

Calculate the visualize the wavefunction $|\psi|^2$ of graphene. The readers need to set `plot_num = 7` and add two lines with `kpoint(1) = 1` and `kband(1) = 4` in the namelist INPUTPP of `pp.in`. This setting will calculate the wavefunction of the 4th band at the Gamma point.

3.2.2 Electronic energy dispersion

❑ **Purpose:** As discussed in Sec. 5.5, the electronic energy dispersion gives the electronic energy as a function of $\boldsymbol{k}$ wavevector, which we call energy band structure. The band structure can tell us useful information, such as a material is metallic or semiconductor. In this tutorial, we show how to calculate the energy dispersion of graphene.

❑ **Background:** In order to calculate the energy dispersion of graphene, we need three-step process as follows:

1. Calculate the Kohn-Sham states with a standard `scf` calculation. As discussed in Sec. 4.10.2, although the Kohn-Sham states are not strictly the real electronic states of the systems, they are often good first-approximations for the electronic states. Therefore, the band structure, which is plotted by the Kohn-Sham eigenvalues, is meaningful to understand the electronic property of the material.
2. Use the output data of the `scf` calculation to perform a non-self-consistent (non-SCF) calculation for many $\boldsymbol{k}$-points along the selected high-symmetry lines. The non-SCF calculates the Kohn-Sham eigenfunctions and eigenvalues without upgrading the Kohn-Sham Hamiltonian at every step.
3. Extract the energies from the non-SCF calculation and convert it to a dataset we can plot with Gnuplot or Python. For this step, we use the executable `bands.x` from Quantum ESPRESSO.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/bands/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
4 $ mpirun -np 4 bands.x < bands.in > bands.out &
```

- Line 1: Go to `bands` directory.
- Line 2: Run `pw.x` with the input file `scf.in` for step (1).
- Line 3: Run `pw.x` with the input file `nscf.in` for step (2).
- Line 4: Run `bands.x` with the input file `bands.in` for step (3).

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of all output files `scf.out`, `nscf.out`, and `bands.out`.

❑ **Input file:** The detail of the `scf.in` file is given in Sec. 3.1.1. For input file `nscf.in`, the readers can use `vi` editor by typing:

```
$ vi nscf.in
```

QE-SSP/gr/bands/nscf.in

```
 1 &CONTROL
 2 calculation    = 'bands'
 3 pseudo_dir     = '../pseudo/'
 4 outdir         = '../tmp/'
 5 prefix         = 'gr'
 6 /
 7 &SYSTEM
 8 ibrav          = 4
 9 a              = 2.4639055825
10 c              = 15.0
11 nat            = 2
12 ntyp           = 1
13 nbnd           = 16
14 occupations    = 'smearing'
15 smearing       = 'mv'
16 degauss        = 0.020
17 ecutwfc        = 40
18 /
19 &ELECTRONS
20 mixing_beta    = 0.7
21 conv_thr       = 1.0D-6
22 /
23 ATOMIC_SPECIES
24 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
25 ATOMIC_POSITIONS (crystal)
26 C  0.333333333  0.666666666  0.500000000
27 C  0.666666666  0.333333333  0.500000000
28 K_POINTS (crystal_b)
29 4
30 gG 40
31 K  20
32 M  30
33 gG 0
```

☞ **Explanation of nscf.in:** The non-SCF for the band structure calculation is performed by using the keyword `calculation = 'bands'` (line 2) in the namelist `CONTROL`. Here, we select the number of Kohn-Sham states with the keyword `nbnd = 16` (line 13) in the namelist

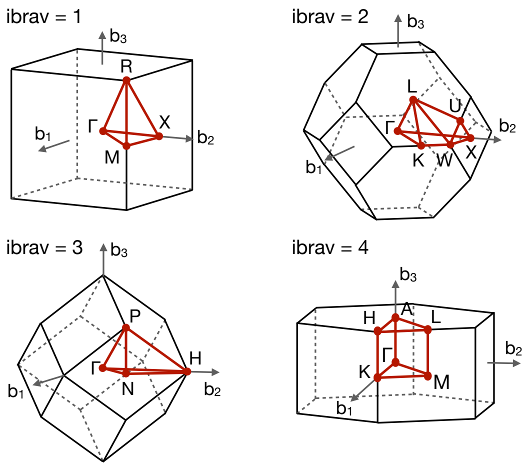


Figure 3.13 The labels of high-symmetry points in the Brillouin-zone of several structures. The Γ-point is denoted by gG in the input file.

SYSTEM, which means that 16 bands are calculated for the case of graphene.

We also need to specify the ***k***-points in the card K_POINTS (crystal_b) (line 28), in which crystal_b allows us to use the labels of high-symmetry points in the Brillouin-zone. In Fig. 3.13, we show the labels for several structures with ibrav = 1, 2, 3, and 4. Note that the Γ-point is denoted by gG (line 30) in the input file. The labels of all structures can be found in Ref. [Setyawan and Curtarolo (2010)]. It is noted that the ***k***-point path must be continuous in the Brillouin-zone.

To see the input file bands.in, the readers can type:

```
$ vi bands.in
```

QE-SSP/gr/bands/bands.in

```
&BANDS
outdir  = '../tmp/'
prefix  = 'gr'
filband = 'gr.bands'
/
```

☞ **Note:** `outdir` and `prefix` must be the same as for `scf.in` for `pw.x` calculation. By setting `lp = .true.` in the namelist `&BANDS`, the readers can obtain the square of the absolute value of the matrix elements of the momentum operator between valence and conduction bands in the output file `p_avg.dat`.

❑ **Output file:** The band structure of graphene is given by `gr.bands`, which is output of `bands.x`. The readers can use `vi` editor to see the `gr.bands` file as follows:

```
$ vi gr.bands
```

QE-SSP/gr/bands/gr.bands

```
 1 &plot nbnd=   16, nks=     91 /
 2           0.000000  0.000000  0.000000
 3 -21.257 -9.357 -4.798 -4.798  1.157  1.900  2.126  4.139
 4   4.937  6.590  6.590  8.097  9.243  9.721 10.839 13.470
 5           0.016667  0.000000  0.000000
 6 -21.252 -9.351 -4.819 -4.809  1.164  1.907  2.133  4.146
 7   4.943  6.593  6.612  8.103  9.250  9.703 10.848 13.476
 8           0.033333  0.000000  0.000000
 9 -21.237 -9.334 -4.880 -4.842  1.184  1.927  2.153  4.166
10   4.963  6.603  6.676  8.123  9.270  9.651 10.874 13.494
11 ...
```

- Line 1: `nbnd` is the number of bands and `nks` is the number of ***k***-points.
- Line 2: The coordinates (k_x, k_y, k_z) of the first ***k***-point in crystal coordinates.
- Line 3 and 4: The list of 16 energies at the first ***k***-point in units of eV.
- Next lines: This format from line 2 to line 4 is repeated until the final ***k***-point.

Other output file is `gr.bands.gnu`, which has a simpler format than `gr.bands`. The `gr.bands.gnu` file contains two columns, the first column is the ***k***-points, and the second column is the energies (eV). We will use this file for plotting as below.

❑ **Plotting data from gr.bands.gnu:** The band structure is plotted by running JupyterLab `plot-bands.ipynb`, which reads the data from the `gr.bands.gnu` file:

```
$ jupyter-lab plot-bands.ipynb
```

QE-SSP/gr/bands/plot-bands.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np

# The Fermi energy, find it in scf.out
efermi = -1.6790

# Load data from gr.bands.gnu
data = np.loadtxt('gr.bands.gnu')
k = np.unique(data[:, 0])
bands = np.reshape(data[:, 1], (-1, len(k)))

# Set high-symmetry points from nscf.in
gG1 = k[0]; K = k[40]; M = k[60]; gG2 = k[90]

# Create figure object
plt.figure()
# Plot dotted line at Fermi energy
plt.axhline(0, c='gray', ls=':')
# Plot dotted lines at high-symmetry points
plt.axvline(K, c='gray')
plt.axvline(M, c='gray')

# Plot band structure
for band in range(len(bands)):
    plt.plot(k, bands[band, :]-efermi, c='b')

# Add the x and y-axis labels
plt.xlabel('')
plt.ylabel('Energy (eV)')
# Set the axis limits
plt.xlim(gG1, gG2)
plt.ylim (-20, 20)
# Add labels for high-symmetry points
plt.xticks([gG1, K, M, gG2], ['$\Gamma$', 'K', 'M',
    '$\Gamma$'])
# Hide x-axis minor ticks
plt.tick_params(axis='x', which='minor', bottom=
    False, top=False)
# Save the figure
plt.savefig('plot-bands.pdf')
# Show the figure
```

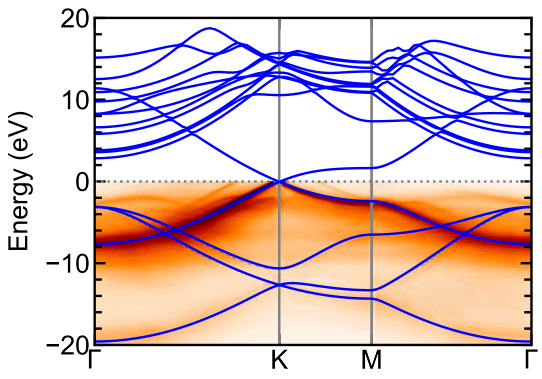


Figure 3.14 Electronic band structure of graphene. The blue lines plot the calculated electron dispersion. Dirac point is the transition between the valence and conduction bands at the K point and zero energy. The ARPES intensity of graphene appears on the electron dispersion of the valence band, which is reproduced with permission [Ohta *et al.* (2007)].

```
plt.show()
```

By running `plot-bands.ipynb`, we obtain the band structure, as shown in Fig. 3.14. It shows that the graphene is a semimetal with a zero band gap at $E = 0$ eV at the K point, which is called the Dirac point since the linear dispersion relation near the K point [Geim and Novoselov (2007)]. In order to compare with the experiment, the angle-resolved photoemission spectroscopy (ARPES) intensity of the electron dispersion of the valence band is plotted below the Dirac point (Fig. 3.14) [Ohta *et al.* (2007)]. The readers can see that the ARPES intensity is reproduced by the calculated electron dispersion (blue lines).

Try It Yourself

1. Calculate electronic energy dispersion of graphene by selecting a LDA functional such as `C.pz-hgh.UPF` and compare it with the GGA functional in Fig. 3.14.
2. Calculate electronic energy dispersion of bulk Si and monolayer MoS_2, and compare calculated energy band gaps with the experiment values (1.12 eV and 2.15 eV for bulk Si and monolayer MoS_2, respectively).

3.2.3 Electronic density of states

❑ **Purpose:** As discussed in Sec. 5.5, the electronic density of states (DOS) is a property on how many electronic states exist per volume per energy. In this tutorial, we will plot the DOS of graphene.

❑ **Background:** In a similar way to calculate the band structure in Sec. 3.2.2, we produce the DOS calculation in the following three steps:

1. Perform the SCF calculation as shown in Sec. 3.2.2.
2. Take the data in the previous step and use it to perform a non-SCF calculation on a more dense grid of ***k***-points. Since the DOS needs an integration over the Brillouin zone, we use a much denser grid for the non-SCF than that for the SCF. In this case, we use the **tetrahedron method** (see Sec. 5.5), in which the Brillouin zone is divided into non-overlapping small tetrahedra, and the integrated quantity is assumed to take a linear interpolation of energy for each tetrahedron. The obtained DOS is equivalent to a very large number of the ***k***-points grid.
3. Convert the state energies calculated on the dense grid of the ***k***-points to the DOS by using the executable `dos.x` from Quantum ESPRESSO. This step will produce `gr.dos` file, which can be plotted by JupyterLab.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/edos/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
4 $ mpirun -np 4 dos.x < dos.in > dos.out &
```

- Line 1: Go to `edos` directory.
- Line 2: Run `pw.x` with the input file `scf.in` for step (1).
- Line 3: Run `pw.x` with the input file `nscf.in` for step (2).
- Line 4: Run `dos.x` with the input file `dos.in` for step (3).

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of all output files `scf.out`, `nscf.out`, and `dos.out`.

❑ **Input file:** The detail of the `scf.in` file is given in Sec. 3.1.1. For input file `nscf.in`, the readers can use `vi` editor by typing:

```
$ vi nscf.in
```

QE-SSP/gr/edos/nscf.in

```
 1 &CONTROL
 2 calculation    = 'nscf'
 3 pseudo_dir     = '../pseudo/'
 4 outdir         = '../tmp/'
 5 prefix         = 'gr'
 6 /
 7 &SYSTEM
 8 ibrav          = 4
 9 a              = 2.4639055825
10 c              = 15.0
11 nat            = 2
12 ntyp           = 1
13 nbnd           = 16
14 occupations    = 'tetrahedra'
15 ecutwfc        = 40
16 /
17 &ELECTRONS
18 mixing_beta    = 0.7
19 conv_thr       = 1.0D-6
20 /
21 ATOMIC_SPECIES
22 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
23 ATOMIC_POSITIONS (crystal)
24 C   0.333333333  0.666666666  0.500000000
25 C   0.666666666  0.333333333  0.500000000
26 K_POINTS (automatic)
27 48 48 1 0 0 0
```

☞ **Explanation of nscf.in:** The non-SCF for the DOS is performed by using the keyword `calculation = 'nscf'` (line 2) in the namelist `CONTROL`, while the tetrahedron method is applied by using the keyword `occupations = 'tetrahedra'` (line 14) in the namelist `SYSTEM`. Here, a ***k***-points grid 48 × 48 × 1 in the namelist `K_POINTS` is selected for the non-SCF calculation.

To see the input file `dos.in`, the readers can type:

```
$ vi dos.in
```

QE-SSP/gr/edos/dos.in

```
&DOS
outdir = '../tmp/'
prefix = 'gr'
fildos = 'gr.dos'
/
```

☞ **Note:** `outdir` and `prefix` must be the same as for `scf.in` for `pw.x` calculation.

❑ **Output file:** The output file for the DOS of graphene is given by `gr.dos`, which is output of step (3). The readers can use `vi` editor to see the `gr.dos` file as follows:

```
$ vi gr.dos
```

QE-SSP/gr/edos/gr.dos

```
  #  E (eV)   dos(E)     Int dos(E) EFermi =   -1.722 eV
  -21.257  0.0000E+00  0.0000E+00
  -21.247  0.4960E+00  0.2480E-02
  -21.237  0.2942E+00  0.5277E-02
...
```

- Column 1: The energies in units of eV.
- Column 2: The DOS in units of state/eV/unit-cell.
- Column 3: The integrated DOS in units of state/unit-cell.

☞ **Note:** The readers can find a difference between the Fermi energy value ($E_F = -1.6708$ eV) of the SCF calculation and $E_F = -1.722$ eV of the non-SCF calculation. $E_F = -1.6708$ eV can be found in `scf.out`, while $E_F = -1.722$ eV is given in the header row of `gr.dos`. This inconsistency might come from a small underestimation in the tetrahedron method.

☞ **Plotting data from gr.dos:** The DOS is plotted by running JupyterLab `plot-dos.ipynb` as follows:

```
$ jupyter-lab plot-dos.ipynb
```

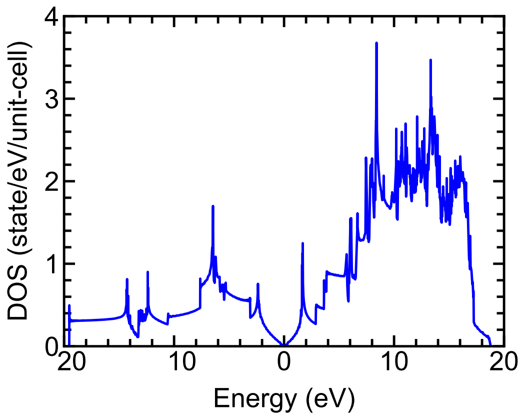


Figure 3.15 Electronic density of states (DOS) of graphene.

QE-SSP/gr/edos/plot-dos.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np

# The Fermi energy in header row of gr.dos
efermi = -1.724

# Open and read the file gr.dos
ener, dos, idos = np.loadtxt('gr.dos', unpack=True)

# Create figure object
plt.figure()
# Plot the DOS with the Fermi energy shifting
plt.plot(ener-efermi, dos, c='k')
# Add the x and y-axis labels
plt.xlabel('Energy (eV)')
plt.ylabel('DOS (state/eV/unit-cell)')
# Set the axis limits
plt.xlim(-20, 20)
plt.ylim(0, 2.5)
# Save the figure
plt.savefig('plot-dos.pdf')
# Show the figure
plt.show()
```

By running `plot-pdos.ipynb`, we obtain the DOS of graphene, as shown in Fig. 3.15. Since graphene is a zero-band-gap semimetal with bands intersecting at the Dirac point at $E = 0$ (see Fig. 3.14), the DOS is zero at the Dirac point.

Try It Yourself

1. Calculate the DOS of bulk Si and monolayer MoS_2.
2. Check if $\mathrm{DOS}(E) \propto \sqrt{E}$ for the 3D DOS, where E is measured from the bottom of the conduction band.

3.2.4 Partial density of states

❑ **Purpose:** Partial density of states (PDOS) is the relative contribution of a particular atom/orbital to the total DOS. In this tutorial, we will plot the PDOS of graphene.

❑ **Background:** For the SCF calculation in Sec. 3.2.2, we selected the pseudopotential `C.pbe-n-rrkjus_psl.0.1.UPF`, which gives us the information of the particular orbital for the C atom. The readers can find this information in `C.pbe-n-rrkjus_psl.0.1.UPF` as

```
Valence configuration:
nl pn  l   occ     Rcut   Rcut US    E pseu
2S  1  0  2.00    1.000    1.300   -1.010676
2P  2  1  2.00    1.000    1.450   -0.388489
```

The pseudopotential shows that the C atom contains the 2s and 2p orbitals. We can calculate the PDOS of the 2s and 2p orbitals for each C atom in the unit cell by using the executable `projwfc.x` from Quantum ESPRESSO.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
$ cd ~/QE-SSP/gr/pdos/
$ mpirun -np 4 pw.x < scf.in > scf.out &
$ mpirun -np 4 pw.x < nscf.in > nscf.out &
$ mpirun -np 4 projwfc.x < projwfc.in > projwfc.out
    &
```

- Line 1: Go to `pdos` directory.
- Line 2: Run SCF calculation by using `pw.x`.
- Line 3: Run non-SCF calculation by using `pw.x`.
- Line 4: Run `projwfc.x` with the input file `projwfc.in`.

☞ **Note:** The SCF and non-SCF calculations can be omitted if the readers have successfully finished the DOS calculation in Sec. 3.2.2.

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of the all output files `scf.out`, `nscf.out`, and `projwfc.out`.

❑ **Input file:** The detail of the input files `scf.in` and `nscf.in` are given in Sec. 3.1.2. For input file `projwfc.in`, the readers can use `vi` editor by typing:

```
$ vi projwfc.in
```

QE-SSP/gr/pdos/projwfc.in

```
&projwfc
prefix  = 'gr'
outdir  = '../tmp/'
/
```

☞ **Note:** `outdir` and `prefix` in `projwfc.in` must be the same as for `scf.in` and `nscf.in`.

❑ **Output file:** The PDOS is written to output files `gr.pdos_atm#N(X)_wfc#M(l)`, where X is atom symbol (C), N is atom number (1, 2) in the unit cell, M is wavefunction number, and l is orbital (s, p, d, etc.). We can open the file `gr.pdos_atm#1(C)_wfc#1(s)` for the PDOS of 2*s* orbital of the first C atom in the unit cell as follows:

```
$ vi gr.pdos_atm#1\(C\)_wfc#1\(s\)
```

QE-SSP/gr/pdos/gr.pdos_atm#1(C)_wfc#1(s)

```
# E (eV)   ldos(E)   pdos(E)
 -21.259  0.000E+00  0.000E+00
 -21.249  0.243E+00  0.243E+00
 -21.239  0.144E+00  0.144E+00
  ...
```

```
   17.041   0.126E-03   0.126E-03
   17.051   0.557E-04   0.557E-04
   17.061   0.000E+00   0.000E+00
```

- Column 1: The energies (in eV).
- Column 2: The "total" PDOS of the 2*s* orbital (LDOS) in state/eV/unit-cell. It is noted that since we consider only wavefunction for *s* orbital, the LDOS is equal to the PDOS. In general, LDOS = PDOS_1 + PDOS_2 + ...
- Column 3: The PDOS of the 2*s* orbital with the first wavefunction of the first C atom (in state/eV/unit-cell).

☞ **Plotting data from gr.dos:** The PDOS of graphene is plotted by running JupyterLab `plot-pdos.ipynb` as follows:

```
$ jupyter-lab plot-dos.ipynb
```

QE-SSP/gr/edos/plot-dos.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np

# The Fermi energy, find it in nscf.out
efermi = -1.7241

# Define the function to read the data file
def r_dos(name):
    ener, dos = np.loadtxt(name, usecols=(0,1),
        unpack=True)
    return ener, dos

# Read the total PDOS
ener, dos = r_dos('gr.pdos_tot')
# Read the PDOS of C atom number 1
ener1, p1C1s = r_dos('gr.pdos_atm#1(C)_wfc#1(s)')
ener1, p1C2p = r_dos('gr.pdos_atm#1(C)_wfc#2(p)')
# Read the PDOS of C atom number 2
ener2, p2C1s = r_dos('gr.pdos_atm#2(C)_wfc#1(s)')
ener2, p2C2p = r_dos('gr.pdos_atm#2(C)_wfc#2(p)')

# Create figure object
plt.figure()
# Plot the DOS
```

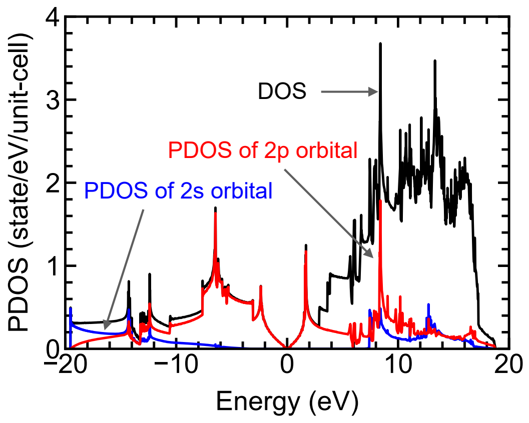


Figure 3.16 The DOS and PDOS of 2s and 2p orbital of graphene.

```python
plt.plot(ener-efermi, dos, c='k')
# Plot the PDOS of s-orbital
plt.plot(ener1-efermi, p1C1s+p2C1s, c='b')
# Plot the PDOS of p-orbital
plt.plot(ener2-efermi, p1C2p+p2C2p, c='r')
# Add the x and y-axis labels
plt.xlabel('Energy (eV)')
plt.ylabel('PDOS (state/eV/unit-cell)')
# Set the axis limits
plt.xlim(-20, 20)
plt.ylim(0, 4)
# Save the figure
plt.savefig('plot-pdos.pdf')
# Show the figure
plt.show()
```

In Fig. 3.16, we show the DOS and the PDOS of the 2s and 2p orbitals as functions of energy E. For the valence bands ($E < 0$), the DOS is equal to the total PDOS since all valence bands mainly come from the 2s and 2p orbitals. For the conduction bands, with $E > 2.5$ eV, the DOS is not equal to the sum of PDOS because the high energy bands include not only the 2s and 2p orbitals but also more delocalized wavefunctions far from the atoms, which are called interlayer-bands [Fretigny *et al.* (1989)].

Try It Yourself

Calculate the contribution of the PDOS of the S and Mo atoms to the total DOS of monolayer MoS_2.

3.3 Lattice oscillations

In this section, we learn how to calculate the frequencies of oscillations of solid. The oscillations of atoms in a crystal with a finite amplitude around the equilibrium positions, known as phonon modes (see Sec. 5.7), are essential to understand the dynamical properties of the material, such as thermal conductance or the Raman spectra.

3.3.1 Phonon dispersion

❑ **Purpose:** As discussed in Sec. 5.7, the phonon dispersion is the phonon frequency as a function of phonon wavevector.[3] The phonon dispersion, which is observed by neutron scattering, can tell us useful information of the material such as elastic constants or thermal conductivity. In this tutorial, we show how to plot the phonon dispersion of graphene.
❑ **Background:** The phonon angular frequencies $\omega_{\boldsymbol{q}}$ are determined by solving the simultaneous equation of motion for each $\boldsymbol{q}$ as

$$\sum_{J\beta} \tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})\tilde{u}_{J\beta}(\boldsymbol{q}) = \omega_{\boldsymbol{q}}^2 \tilde{u}_{I\alpha}(\boldsymbol{q}) \text{ with } (I\alpha = 1, ..., 3N), \tag{3.9}$$

where N is the number of atoms in the unit cell and α (β) corresponds to x, y, z components of oscillation. $\tilde{u}_{I\alpha}$ ($\tilde{u}_{J\beta}$) is a phonon eigenvector (or the amplitude of the phonon) of the I-th (J-th) atom. Summation on $J\beta$ is taken for neighbor atoms from a given I-th atom up to several nearest neighbors. $\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})$ is called the dynamical matrix, which is

[3] We use $\boldsymbol{k}$-points to refer to electron wavevectors and $\boldsymbol{q}$-points to refer to phonon wavevectors

defined by

$$\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q}) = \frac{1}{\sqrt{M_I M_J}} \tilde{F}_{I\alpha,J\beta}(\boldsymbol{q}), \tag{3.10}$$

where M_I (M_J) is mass of the I-th (J-th) atom. $\tilde{F}_{I\alpha,J\beta}$ is second derivatives of the total energy E_{tot} with respect to the displacements ($\tilde{u}_{I\alpha}, \tilde{u}_{J\beta}$) of atomic positions, which is called the inter-atomic force constant (IFC), which is given by

$$\tilde{F}_{I\alpha,J\beta}(\boldsymbol{q}) = \frac{\partial^2 E_{\text{tot}}}{\partial \tilde{u}^*_{I\alpha}(\boldsymbol{q}) \partial \tilde{u}_{J\beta}(\boldsymbol{q})}. \tag{3.11}$$

$\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})$ in Eq. (3.10) is calculated by the executable `ph.x` in Quantum ESPRESSO within the density-functional perturbation theory[4] (DFPT) [Baroni *et al.* (2001)]. Then, $\omega_{\boldsymbol{q}}$ is obtained from Eq. (3.9) by diagonalizing $\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})$. By repeating this procedure for several $\boldsymbol{q}$, we obtain $\omega_{\boldsymbol{q}}$ as a function of $\boldsymbol{q}$ and plot the phonon dispersions. However, the DFPT calculation is time-consuming compared with the SCF calculation. Thus, we will calculate $\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})$ for a small set of $\boldsymbol{q}$ vectors and take an interpolation for $\omega_{\boldsymbol{q}}$ over the Brillouin zone.

In order to plot the $\omega_{\boldsymbol{q}}$ by interpolation, we use the executables `q2r.x` and `matdyn.x` in Quantum ESPRESSO. `q2r.x` reads $\tilde{D}_{I\alpha,J\beta}(\boldsymbol{q})$, which is calculated by `ph.x` for a uniform mesh of (q_1, q_2, q_3) vectors. Then, the IFC of a $q_1 \times q_2 \times q_3$ supercell in the real space, $F_{I\alpha,J\beta}(\boldsymbol{R})$, is calculated from $\tilde{F}_{I\alpha,J\beta}(\boldsymbol{q})$ by using the inverse Fourier transform as

$$F_{I\alpha,J\beta}(\boldsymbol{R}) = \frac{1}{N_q} \sum_{i=1}^{N_q} \tilde{F}_{I\alpha,J\beta}(\boldsymbol{q}_{ijk}) e^{i\boldsymbol{q}_{ijk}\cdot\boldsymbol{R}}, \tag{3.12}$$

where $N_q = q_1 \times q_2 \times q_3$, $\boldsymbol{R}$ is an atomic position in the $q_1 \times q_2 \times q_3$ supercell, and $\boldsymbol{q}_{ijk}$ is defined by

$$\boldsymbol{q}_{ijk} = \frac{i-1}{q_1}\boldsymbol{G}_1 + \frac{j-1}{q_2}\boldsymbol{G}_2 + \frac{k-1}{q_3}\boldsymbol{G}_3, \tag{3.13}$$

[4] The DFPT is a method that can directly calculate the second-order derivatives of the energy in Eq. (3.11) by a self-consistent procedure.

where $i = 1, ..., q_1$, $j = 1, ..., q_2$, $k = 1, ..., q_3$, and $\boldsymbol{G}_1$, $\boldsymbol{G}_2$, and $\boldsymbol{G}_3$ are the reciprocal lattice vectors. Then, `matdyn.x` reads the IFC that is calculated by `q2r.x` and calculates the IFC at an interpolated $\boldsymbol{q}'$ by using the Fourier transform as

$$\tilde{F}_{I\alpha,J\beta}(\boldsymbol{q}') = \sum_{\boldsymbol{R}} F_{I\alpha,J\beta}(\boldsymbol{R}) e^{i\boldsymbol{q}'\cdot\boldsymbol{R}}. \tag{3.14}$$

Eqs. (3.12) and (3.14) allow the interpolation of the dynamical matrix at arbitrary $\boldsymbol{q}'$, by a few IFCs. It is noted that the Fourier interpolation works well if the IFCs decay rapidly in the real space. Therefore, the Fourier interpolation might fail in some special cases when IFCs do not decay. For example, when the Kohn anomaly occurs in a metal, the dynamical matrix is not a smooth function of $\boldsymbol{q}$, and the IFCs decay slowly in the real space.

The calculation of the phonon dispersion of graphene has the following four steps:

1. Perform the SCF with `pw.x` calculation.
2. Calculate the dynamical matrix in Eq. (3.10) on a uniform mesh of $\boldsymbol{q}$-points by using `ph.x`.
3. Perform the inverse Fourier transform of the dynamical matrix in Eq. (3.12) to obtain a set of the IFCs in the real space by using `q2r.x`.
4. Perform the Fourier transform of the real space IFCs in Eq. (3.14) to obtain the dynamical matrix at any $\boldsymbol{q}'$ by using `matdyn.x`. This allows us to calculate the phonon frequencies for a set of points along lines between high-symmetry points in the Brillouin zone, which is similar to the electronic band structure in Sec. 3.2.2.

❑ **How to run:** The readers can run the following command lines:

```
1 $ cd ~/QE-SSP/gr/phonon/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 ph.x < ph.in > ph.out &
4 $ mpirun -np 4 q2r.x < q2r.in > q2r.out &
5 $ mpirun -np 4 matdyn.x < matdyn.in > matdyn.out&
```

- Line 1: Go to `phonon` directory.
- Line 2: Run SCF calculation with `pw.x` for step (1).
- Line 3: Run phonon calculation with `ph.x` for step (2).
- Line 4: Run `q2r.x` for step (3).

- Line 5: Run `matdyn.x` for step (4).

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the each end of the output files; `scf.out`, `ph.out`, `q2r.out`, and `matdyn.out`.

❑ **Input file:** The input file `scf.in` can be opened as follows:

```
$ vi scf.in
```

QE-SSP/gr/phonon/scf.in

```
 1 &CONTROL
 2 calculation     = 'scf'
 3 pseudo_dir      = '../pseudo/'
 4 outdir          = '../tmp/'
 5 prefix          = 'gr'
 6 /
 7 &SYSTEM
 8 ibrav           = 4
 9 a               = 2.4639055825
10 c               = 15.0
11 nat             = 2
12 ntyp            = 1
13 occupations     = 'smearing'
14 smearing        = 'mv'
15 degauss         = 0.02
16 ecutwfc         = 80
17 assume_isolated = '2D'
18 /
19 &ELECTRONS
20 mixing_beta     = 0.7
21 conv_thr        = 1.0D-6
22 /
23 ATOMIC_SPECIES
24 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
25 ATOMIC_POSITIONS (crystal)
26 C  0.333333333  0.666666666  0.500000000
27 C  0.666666666  0.333333333  0.500000000
28 K_POINTS (automatic)
29 12 12 1 0 0 0
```

☞**Explanation of scf.in:** Compared with the `scf.in` file in the previous tutorials (see Sec. 3.2.2 or 3.2.3), a new syntax `assume_isolated = '2D'` (line 17) is added in the namelist `SYSTEM`. This syntax can help to avoid the "flexural" phonons (negative or imaginary acoustic frequencies at near Gamma point) in the 2D materials [Sohier *et al.* (2017)].

☞**Note:** For phonon calculation, the readers should use a larger value of `ecutwfc = 80` (line 16) compared with the SCF calculation (`ecutwfc = 40`). This is because that the phonon frequencies have the unit of cm^{-1} ($\sim$ 0.00012 eV), which requires a large cutoff energy.

To see the input file `ph.in`, the readers can type:

```
$ vi ph.in
```

QE-SSP/gr/phonon/ph.in

```
 1 phonon calc.
 2 &INPUTPH
 3 outdir = '../tmp/'
 4 prefix = 'gr'
 5 tr2_ph = 1d-14
 6 ldisp  = .true.
 7 nq1    = 6
 8 nq2    = 6
 9 nq3    = 1
10 fildyn = 'gr.dyn'
11 /
```

☞**Explanation of ph.in:** In order to obtain the phonon dispersion, we need to set `ldisp = .true.` (line 6) and a grid of ***q***-points specified by `nq1, nq2, nq3`. In the case of the 2D materials, we can put `nq3 = 1`. Since the `ph.x` calculation is time-consuming, `nq1, nq2, nq3` are usually taken smaller than `nk1, nk2, nk3` of the `pw.x` calculation. The description of input variables of `ph.in` is given in Table 3.9.

To see the input file `q2r.in`, the readers can type:

```
$ vi q2r.in
```

QE-SSP/gr/phonon/q2r.in

```
&INPUT
fildyn = 'gr.dyn'
zasr   = 'crystal'
```

Table 3.9 Meaning of input variables in `ph.in` file.

Line	Syntax	Meaning
1	`title_line`	Title of the job, i.e., a line that is reprinted on output
4	`prefix`	Prefix of files saved by program `pw.x`.
5	`tr2_ph`	Threshold for self-consistency.
6	`ldisp`	The default is `ldisp = .false.`. If we set `ldisp = .true.`, the dynamical matrix is calculated for a grid of ***q***-points specified by `nq1, nq2, nq3`.
7–9	`nq1, nq2, nq3`	Parameters of the Monkhorst-Pack grid (no offset) used when `ldisp = .true.`. It is same meaning as for `nk1, nk2, nk3` in the input of `pw.x`.
10	`fildyn`	File name where the dynamical matrix is written.

```
4 flfrc   = 'gr.fc'
5 /
```

☞**Explanation of q2r.in:** The syntax `zasr` is used to impose the IFCs to adopt the acoustic sum rule (ASR).[5] There are several options for the ASR imposed. Here, we select `zasr = 'crystal'` (line 3), i.e., three translational ASR are imposed by optimized correction of the dynamical matrix. The file containing the IFCs in the real space is given by the `gr.fc` file.

☞ **Note:** `fildyn` in `q2r.in` must be the same as that for `ph.in` for `ph.x` calculation.

To see the input file `matdyn.in`, the readers can type:

```
$ vi matdyn_disp.in
```

[5] The ASR is a requirement for IFCs by translational and rotational invariance of the crystal. If we translate the whole solid by a uniform displacement or if we rotate the solid, the inter-atomic forces by the displacements on rotation should not appear. As a consequence of ASR, the frequencies of the acoustic modes become zero [Madelung (1978)].

QE-SSP/gr/phonon/matdyn.in

```
 1 &INPUT
 2 asr             = 'crystal'
 3 flfrc           = 'gr.fc'
 4 flfrq           = 'gr.freq'
 5 flvec           = 'gr.modes'
 6 loto_2d         = .true.
 7 q_in_band_form = .true.
 8 /
 9 4
10 gG  40
11 K   20
12 M   30
13 gG  0
```

☞**Explanation of matdyn.in:** The syntax `loto_2d` is set to `.true.` (line 6) to activate two-dimensional treatment of LO-TO splitting,[6] which occurs in a polar 2D materials as h-BN or MoS_2. It is noted that LO-TO splitting does not occur in the case of graphene since it is not a polar material. The setting `q_in_band_form = .true` (line 7) allows us to use the labels of high-symmetry points in the Brillouin-zone (see Sec. 3.2.2) to plot the phonon dispersion.

☞ **Note:** By changing `asr = 'crystal'` (line 2) to `asr = 'simple'` in `matdyn.in`, the readers might obtain the negative phonon frequency at near the Γ point. The option `asr = 'crystal'` is usually better than `asr = 'simple'`.

❑ **Output file:** The output file for the phonon frequencies of graphene is given by `gr.freq`. The readers can open this file by

```
$ vi gr.freq
```

QE-SSP/gr/phonon/gr.freq

```
&plot nbnd=   6, nks=  91 /
      0.000000  0.000000  0.000000
-0.0000 -0.0000 -0.0000 876.7549 1488.7595 1488.7595
      0.016667  0.000000  0.000000
 7.4052 31.1526 49.6068 876.5584 1489.4337 1490.5041
  ...
```

[6] In the ionic crystal, we know the splitting between LO and TO as $\omega_{\mathrm{LO}}^2 = (\varepsilon_0/\varepsilon_\infty)\omega_{\mathrm{TO}}^2$ [Kittel (1976)].

- Line 1: nbnd is the number of phonon modes and nks is the number of ***q***-points.
- Line 2: The coordinates (q_x, q_y, q_z) of the first ***q***-point in crystal coordinates.
- Line 3: The list of 6 phonon frequencies at the first ***q***-point in units of cm^{-1}.
- Next lines: This format from line 2 to line 3 is repeated until the final ***q***-point.

The matdyn.x calculation in the step (4) also automatically outputs a file gr.freq.gp, which has a simpler format than gr.freq. The gr.freq.gp file contains 7 columns, in which the first column is the ***q***-points and other columns are phonon frequencies (cm^{-1}). We will use this file for plotting as below.

❑ **Plotting data from gr.freq.gp:** The phonon dispersion is plotted by running JupyterLab plot-phonon.ipynb, which reads the data from the gr.freq.gp file:

```
$ jupyter-lab plot-phonon.ipynb
```

QE-SSP/gr/phonon/plot-phonon.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Number of phonon modes and q-points gr.freq
nbnd = 6; nks = 91
# Open gr.freq.gp file
qs, *ph = np.loadtxt('gr.freq.gp', unpack=True)
# Read phonon at each phonon index
ph0 = []
for iq in range(nks):
    ph0.append([])
    for ib in range(nbnd):
        tmp = ph[ib][iq]
        ph0[iq].append(float(tmp))
# Set high-symmetry points from matdyn.in
G1 = qs[0]; K = qs[40]; M = qs[60]; G2 = qs[90]
# Create figure object
plt.figure()
# Plot dotted lines at high-symmetry points
plt.axvline(K, c='gray')
plt.axvline(M, c='gray')
```

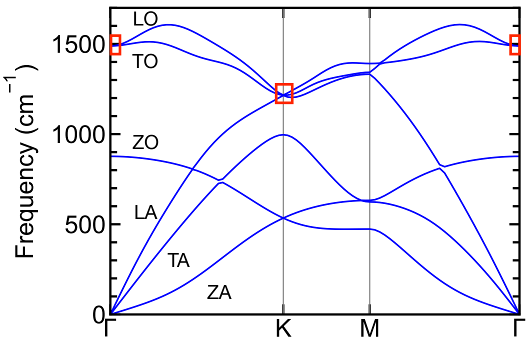


Figure 3.17 Phonon dispersion of graphene along the high symmetry-lines of hexagonal Brillouin zone. The red boxes show the Kohn anomalies at the Γ and the K points of the Brillouin zone.

```python
# Plot phonon dispersion
plt.plot(qs, ph0, c='b')
# Add the x and y-axis labels
plt.xlabel('')
plt.ylabel('Frequency (cm$^{-1}$)')
plt.xlim(G1, G2)
plt.ylim(0, 1700)
# Add labels for high-symmetry points
plt.xticks([G1, K, M, G2], ['$\Gamma$', 'K', 'M', '$
    \Gamma$'])
# Hide x-axis minor ticks
plt.tick_params(axis='x', which='minor', bottom=
    False, top=False)
# Save the figure
plt.savefig('plot-phonon.pdf')
# Show the figure
plt.show()
```

By running `plot-phonon.ipynb`, we obtain the phonon dispersion of graphene, as shown in Fig. 3.17. There are 6 phonon modes including three acoustic branches (ZA, TA, and LA) and three optical branches (ZO, TO, and LO). The red boxes in Fig. 3.17 show the Kohn anomaly[7] [Kohn (1959)] for certain phonon modes at the Γ

[7] The Kohn anomaly is an anomaly in the phonon dispersion of a metal. For a specific wavevector, the frequency of the associated phonon is softened, and there is a discontinuity in the derivative of frequency.

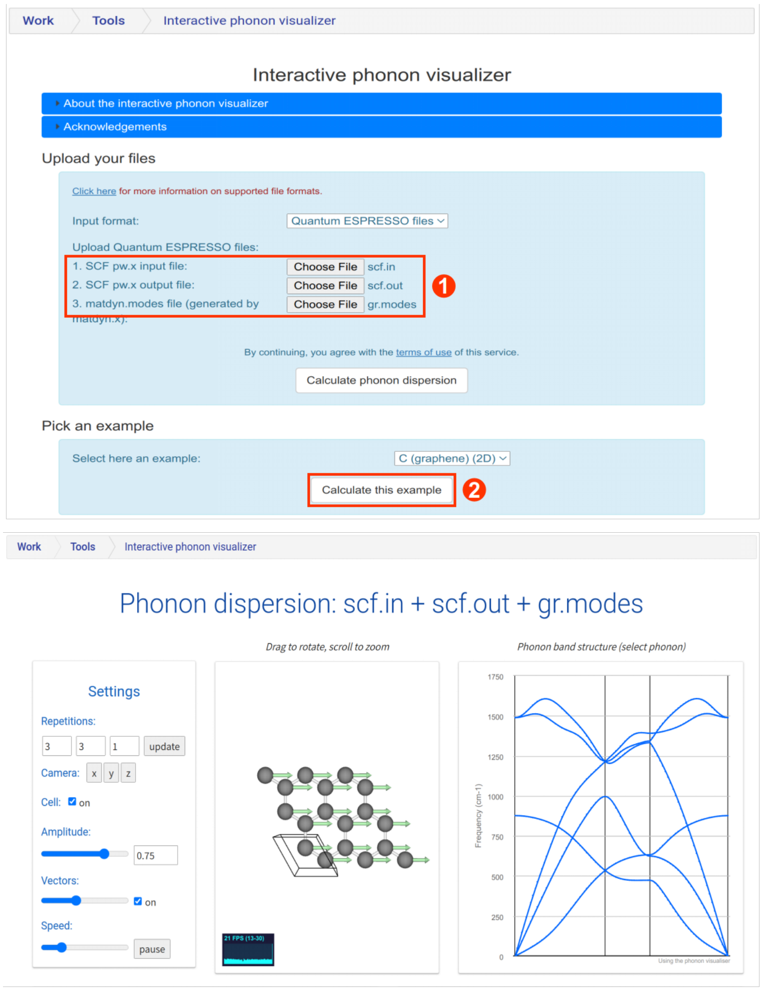
and the K points of the Brillouin zone, arising from perturbation of a phonon by electron-phonon interaction. Specifically, the Kohn anomaly is manifested by a non-zero slope of LO and TO at the Γ point and a non-zero slope of TO at the K point. This effect was observed experimentally by the Raman spectra of the G band, due to the LO/TO phonon, at different charge doping level [Lazzeri and Mauri (2006)].
☞ **Visualizing phonon dispersion:** The readers can also visualize the phonon dispersion calculated with a display graphically the phonon modes of the lattice vibrations. First, the readers need to upload the input file `scf.in` and the output files `scf.out` and `gr.modes` to the website: `https://www.materialscloud.org/work/tools/interactivephonon`, as shown in the red box of Fig. 3.18 (top). Then, the readers click on the button "Calculate phonon dispersion". A new window will appear, as shown in Fig. 3.18 (bottom). In the phonon section, the readers can click on any point in the phonon dispersion and see an animation of how the atoms vibrate according to this model.

Try It Yourself

1. Calculate phonon dispersion of graphene without `assume_isolated = '2D'` option in `scf.in`, and check if the negative phonon frequency appears for the ZA mode.
2. Calculate phonon dispersion of monolayer MoS_2.

3.3.2 Phonon density of states

❏ **Purpose:** Phonon density of states (DOS) is defined in exactly the same way as the electronic density of states (see Sec. 3.2.3), i.e., the number of phonon modes per volume per frequency. In this tutorial, we will plot the phonon DOS of graphene.
❏ **Background:** The phonon DOS can be obtained by choosing some input options for `matdyn.x` in step (4) of the phonon calculation (see Sec. 3.3.1). Therefore, the readers are asked to finish the phonon calculation in Sec. 3.3.1 before this tutorial.
❏ **How to run:** To run this tutorial, the readers should type the following command lines:



Work > Tools > Interactive phonon visualizer

Interactive phonon visualizer

About the interactive phonon visualizer

Acknowledgements

Upload your files

Click here for more information on supported file formats.

Input format: Quantum ESPRESSO files

Upload Quantum ESPRESSO files:

1. SCF pw.x input file: Choose File scf.in
2. SCF pw.x output file: Choose File scf.out
3. matdyn.modes file (generated by matdyn.x): Choose File gr.modes

By continuing, you agree with the terms of use of this service.

Calculate phonon dispersion

Pick an example

Select here an example: C (graphene) (2D)

Calculate this example

Work > Tools > Interactive phonon visualizer

Phonon dispersion: scf.in + scf.out + gr.modes

Settings

Repetitions: 3 3 1 update

Camera: x y z

Cell: ☑ on

Amplitude: 0.75

Vectors: ☑ on

Speed: pause

1
2
Drag to rotate, scroll to zoom
Phonon band structure (select phonon)
21 FPS (13-30)
Frequency (cm-1)
1750
1500
1250
1000
750
500
250
0
Using the phonon visualiser

Figure 3.18 Visualizing phonon dispersion by using free-online-tool "Interactive phonon visualizer".

```
1 $ cd ~/QE-SSP/gr/phdos/
2 $ cp ../phonon/gr.fc ./
3 $ mpirun -np 4 matdyn.x < matdyn.in> matdyn.out &
```

- Line 1: Go to `phdos` directory.

- Line 2: Copy the IFCs file `gr.fc` from Sec 3.3.1.
- Line 3: Run `matdyn.x` with the input file `matdyn.in` for the phonon DOS calculation. It is noted that this `matdyn.in` file is different with the `matdyn.in` file in Sec 3.3.1.

❑ **How to check:** When the calculations finish, you can find a message `JOB DONE` at the end of the output file `matdyn.out`.

❑ **Input file:** To see the input file `matdyn.in`, the readers can type:

```
$ vi matdyn.in
```

QE-SSP/gr/phdos/matdyn.in

```
 1 &INPUT
 2 asr       = 'crystal'
 3 flfrc     = 'gr.fc'
 4 flfrq     = 'gr.freq'
 5 flvec     = 'gr.modes'
 6 loto_2d   = .true.
 7 fldos     = 'gr.dos'
 8 dos       = .true.
 9 nk1       = 48
10 nk2       = 48
11 nk3       = 1
12 /
```

☞**Explanation of matdyn.in:** The syntax `dos` is set to `.true.` (line 8) to obtain the phonon DOS. This is similar to the electronic density of states in Sec. 3.2.3, we need to select a dense $\boldsymbol{q}$-points grid to calculate the phonon DOS compared with $\boldsymbol{q}$-points of the `ph.x` calculation. Here, we select $48 \times 48 \times 1$ for the case of graphene.

❑ **Output file:** The output file for the phonon DOS of graphene is given by `gr.dos`. The readers can use `vi` editor to see the `gr.dos` file as follows:

```
$ vi gr.dos
```

QE-SSP/gr/phdos/gr.dos

```
# Frequency[cm^-1] DOS PDOS
-2.4978196418E-05 0.0000000000E+00 0.0000E+00 0.0000E+00
 9.9997502180E-01 2.4555277523E-05 1.2278E-05 1.2278E-05
 1.9999750218E+00 4.9110570407E-05 2.4555E-05 2.4555E-05
...
```

- Column 1: The phonon frequencies in units of cm$^{-1}$. We can neglect a tiny negative value `-2.4978196418E-05` and assumed to be zero.
- Column 2: The phonon DOS in units of state/cm$^{-1}$/unit-cell.
- Columns 3 and 4: The partial phonon density of states (PDOS) for the C atoms 1 and 2 in the unit cell (in state/cm$^{-1}$/unit-cell).

☞ **Plotting data from gr.dos:** The phonon DOS is plotted by running JupyterLab `plot-phdos.ipynb`, which reads the data from the `gr.dos` file:

```
$ jupyter-lab plot-phdos.ipynb
```

QE-SSP/gr/edos/plot-dos.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Load data from gr.dos
omega, ph_tot, ph_c1, ph_c2 = np.loadtxt('gr.dos',
    unpack=True)
# Create figure object
plt.figure()
# Plot the phonon DOS
plt.plot(omega, ph_tot, c='b', label='Total')
plt.plot(omega, ph_c1, c='r', ls='dashed', label='C
    atom')
# Add the x and y-axis labels
plt.xlabel('Frequency (cm$^{-1}$)')
plt.ylabel('Phonon DOS (state/cm$^{-1}$/unit-cell)')
# Add the legend
plt.legend(loc='upper left')
# Set the axis limits
plt.xlim(0, 1700)
plt.ylim(0, 0.025)
# Save the figure
plt.savefig('plot-phdos.pdf')
# Show the figure
plt.show()
```

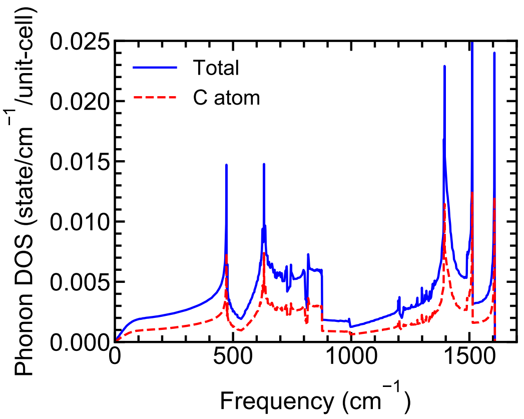


Figure 3.19 Phonon density of states (solid line) and phonon PDOS of a C atom (dashed line) of graphene.

By running `plot-phdos.ipynb`, we obtain the phonon DOS of graphene, as shown in Fig. 3.19. The total phonon DOS (solid line) of graphene is twice as large as the PDOS of a single C atom (dashed line) since there are two C atoms in the unit cell of graphene.

Try It Yourself

Calculate contributions of the phonon DOS of the Mo and S atoms to total phonon DOS of monolayer MoS_2.

3.3.3 Electron-phonon interaction

❑ **Purpose:** In solids, the electron-phonon interaction plays an important role in a variety of physical phenomena, such as temperature-dependent energy band, the Kohn anomaly, electrical conductivity, or superconductivity. As discussed in Sec. 5.8, the origin of electron-phonon interaction is the oscillation of atomic potential by lattice oscillation. In this tutorial, we calculate the electron-phonon interaction of graphene.

❑ **Background:** In Quantum ESPRESSO, the electron-phonon matrix elements, $g_{\nu\boldsymbol{q}}$, for the phonon mode ν at a wavevector $\boldsymbol{q}$ are defind as

$$g_{\nu\boldsymbol{q}}(\boldsymbol{k},i,j) = \left(\frac{\hbar}{2M\omega_{\nu\boldsymbol{q}}}\right)^{1/2} \langle\psi_{i,\boldsymbol{k}}|\partial_{\nu\boldsymbol{q}}\mathcal{V}_{\text{SCF}}|\psi_{j,\boldsymbol{k}+\boldsymbol{q}}\rangle, \tag{3.15}$$

where M is the total mass of the atoms in the unit cell, $\omega_{\nu\boldsymbol{q}}$ is the phonon frequency, $\partial_{\nu\boldsymbol{q}}\mathcal{V}_{\text{SCF}}$ is the derivative of the self-consistent potential $\mathcal{V}_{\text{SCF}}$ by ionic displacement by phonon mode ν at $\boldsymbol{q}$, and $\psi_{i,\boldsymbol{k}}$ and $\psi_{j,\boldsymbol{k}+\boldsymbol{q}}$ are the wavefunctions at initial state i and final state j, respectively.

The phonon linewidth $\gamma_{\nu\boldsymbol{q}}$, which is the imaginary part of the phonon self-energy, is defined by $g_{\nu\boldsymbol{q}}$ in Eq. (3.15) as

$$\gamma_{\nu\boldsymbol{q}} = 2\pi\omega_{\nu\boldsymbol{q}} \sum_{ij} \int_{\text{BZ}} \frac{\mathrm{d}\boldsymbol{k}}{\Omega_{\text{BZ}}} |g_{\nu\boldsymbol{q}}(\boldsymbol{k},i,j)|^2 \delta(\epsilon_{i,\boldsymbol{k}} - \epsilon_F)\delta(\epsilon_{j,\boldsymbol{k}+\boldsymbol{q}} - \epsilon_F), \tag{3.16}$$

where Ω_{BZ} is the volume of the Brillouin zone (BZ), ϵ_F is the Fermi energy, and $\epsilon_{i,\boldsymbol{k}}$ and $\epsilon_{j,\boldsymbol{k}+\boldsymbol{q}}$ are the energies of the initial state i and final state j, respectively.

The electron-phonon coupling constant $\lambda_{\nu\boldsymbol{q}}$ is defined by $\gamma_{\nu\boldsymbol{q}}$ in Eq. (3.16) as

$$\lambda_{\nu\boldsymbol{q}} = \frac{\gamma_{\nu\boldsymbol{q}}}{\pi\hbar D(\epsilon_F)\omega_{\nu\boldsymbol{q}}^2}, \tag{3.17}$$

where $D(\epsilon_F)$ is the density of states (DOS) at the Fermi energy.

The calculation of the electron-phonon interaction of graphene consists the following 5 steps:

1. Perform the SCF for a dense $\boldsymbol{k}$-points grid with `pw.x` calculation. The dense grid must contain all $\boldsymbol{k}$ and $\boldsymbol{k}+\boldsymbol{q}$ points, which is used in the electron-phonon calculation and must be dense enough to produce the phonon linewidth accurately.
2. Perform other SCF for a grid of $\boldsymbol{k}$-points that is suitable for the phonon calculation by using `pw.x`.
3. Perform the phonon calculation based on the SCF in step (2) by using `ph.x`.

4. Perform the inverse Fourier transform of dynamical matrix and the phonon linewidth, which are obtained in step (3), in real space by using `q2r.x`.
5. Perform the Fourier transform to obtain the dynamical matrix and the phonon linewidth for a set of points along lines between high-symmetry points in the Brillouin zone by using `matdyn.x`.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/elph/
2 $ mpirun -np 4 pw.x < scf-dense.in > scf-dense.out &
3 $ mpirun -np 4 pw.x < scf.in > scf.out &
4 $ mpirun -np 4 ph.x < ph.in > ph.out &
5 $ mpirun -np 4 q2r.x < q2r.in > q2r.out &
6 $ mpirun -np 4 matdyn.x < matdyn.in> matdyn.out &
```

- Line 1: Go to `elph` directory.
- Line 2: Run `pw.x` for the SCF calculation with a dense ***k***-points grid.
- Line 3: Run `pw.x` for the SCF calculation with a suitable ***k***-points grid for phonon calculation.
- Line 4: Run `ph.x` with the input file `ph.in` for the electron-phonon calculation.
- Line 5: Run `q2r.x` with the input file `q2r.in`.
- Line 6: Run `matdyn.x` with the input file `matdyn.in`.

❑ **How to check:** When the calculations finish, you can find a message `JOB DONE` at the end of each output file.

❑ **Input file:** The readers can open the input file `scf-dense.in` as follows:

```
$ vi scf-dense.in
```

QE-SSP/gr/elph/scf-dense.in

```
&CONTROL
calculation      = 'scf'
pseudo_dir       = '../pseudo/'
outdir           = '../tmp/'
prefix           = 'gr'
/
```

```
 7 &SYSTEM
 8 ibrav            = 4
 9 a                = 2.4639055825
10 c                = 15.0
11 nat              = 2
12 ntyp             = 1
13 occupations      = 'smearing'
14 smearing         = 'mv'
15 degauss          = 0.020
16 ecutwfc          = 80
17 assume_isolated  = '2D'
18 la2F             = .true.
19 /
20 &ELECTRONS
21 mixing_beta      = 0.7
22 conv_thr         = 1.0D-6
23 /
24 ATOMIC_SPECIES
25 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
26 ATOMIC_POSITIONS (crystal)
27 C  0.333333333  0.666666666  0.500000000
28 C  0.666666666  0.333333333  0.500000000
29 K_POINTS (automatic)
30 36 36 1 0 0 0
```

☞**Explanation of scf-dense.in:** The option `la2F = .true.` (line 18) in the namelist `SYSTEM` instructs the `pw.x` to save the eigenvalues on the dense ***k***-points grid into a file `gr.a2Fsave`, which can be found in `outdir` (line 4). Here, we select a dense ***k***-points grid is $36 \times 36 \times 1$ (line 30) in the namelist `K_POINTS`. The remaining part is same as Sec. 3.3.1.

☞**Note:** The dense ***k***-points grid must be unshifted (line 30) because the grid should include the Γ point (see Sec. 3.1.3). Since the dense grid must contain all ***k*** and ***k*** + ***q*** points, it should be a multiple of ***q***-point grid in `ph.in` for phonon calculations, otherwise the `ph.x` calculation will be stopped by showing the following error:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 task #         3
 from elphsum : error #         2
 q is not a vector in the dense grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The input file `scf.in` is same as Sec. 3.3.1. For the input file `ph.in`, the readers can open by typing:

```
$ vi ph.in
```

QE-SSP/gr/elph/ph.in

```
 1 phonon calc.
 2 &INPUTPH
 3 outdir          = '../tmp/'
 4 prefix          = 'gr'
 5 tr2_ph          = 1d-14
 6 ldisp           = .true.
 7 nq1             = 6
 8 nq2             = 6
 9 nq3             = 1
10 fildyn          = 'gr.dyn'
11 fildvscf        = 'grdv'
12 electron_phonon = 'interpolated'
13 el_ph_nsigma    = 10
14 el_ph_sigma     = 0.02
15 /
```

☞**Explanation of ph.in:** The electron-phonon calculation is performed by setting `electron_phonon = 'interpolated'` (line 12). The option `fildvscf = 'grdv'` (line 11) is the file name of derivative of the self-consistent potential $\partial_{\nu\boldsymbol{q}}\mathcal{V}_{\mathrm{SCF}}$ in Eq. (3.15), which is stored at `_ph0` in `outdir` (line 3) for each $\boldsymbol{q}$ point. `el_ph_nsigma` (line 13) is the number of the Gaussian broadening values, which is used for delta functions in Eq. (3.16). `el_ph_sigma` (line 14) is the spacing (in Ry) between the Gaussian broadening values. The remaining part is same as Sec. 3.3.1.

☞**Explanation of q2r.in and matdyn.in:** The input files `q2r.in` and `matdyn.in` are the same as Sec. 3.3.1, but the option `la2F = .true.` should be added in the namelist `INPUT` of these input files to obtain the electron-phonon coupling along lines between high-symmetry points in the Brillouin zone.

❑ **Output file:** The output files of both the phonon linewidth and the electron-phonon coupling are stored in the files `elph_dir/elph.inp_lambda.*` for each $\boldsymbol{q}$ point. For example, the readers can open the file `elph_dir/elph.inp_lambda.1` for the Γ point as

```
$ vi elph_dir/elph.inp_lambda.1
```

QE-SSP/gr/elph/elph_dir/elph.inp_lambda.1

```
 1 |    0.000000      0.000000      0.000000    10     6
 2 | 0.119733E-06  0.119733E-06  0.173580E-06  0.638932E-04
   |      0.183390E-03  0.183390E-03
 3 |   Gaussian Broadening:   0.020 Ry, ngauss=   0
 4 |   DOS =  0.122860 states/spin/Ry/Unit Cell at Ef= -4.238911
   |         eV
 5 |   lambda(    1)=  0.3002   gamma=    0.05 GHz
 6 |   lambda(    2)=  0.1533   gamma=    0.02 GHz
 7 |   lambda(    3)=  0.0001   gamma=    0.00 GHz
 8 |   lambda(    4)=  0.0000   gamma=    0.00 GHz
 9 |   lambda(    5)=  1.3810   gamma=  321.60 GHz
10 |   lambda(    6)=  1.3816   gamma=  321.72 GHz
11 |   Gaussian Broadening:   0.040 Ry, ngauss=   0
12 |   DOS =  0.242754 states/spin/Ry/Unit Cell at Ef= -4.244465
   |         eV
13 |   lambda(    1)=  2.5752   gamma=    0.77 GHz
14 | ...
```

- Line 1: The coordinates (q_x, q_y, q_z) of $\boldsymbol{q}$, `el_ph_nsigma`, and the number of phonon modes `3*nat`.
- Line 2: $\omega^2_{\nu\boldsymbol{q}}$ (in Ry^2) for each phonon mode ν from 1 to 6.
- Line 3: The Gaussian broadening value (in Ry).
- Line 4: The DOS at the Fermi energy (in states/spin/Ry/unit-cell).
- Lines 5 to 10: $\lambda_{\nu\boldsymbol{q}}$ and $\gamma_{\nu\boldsymbol{q}}$ (in GHz) for each ν.
- From line 11: The DOS, $\lambda_{\nu\boldsymbol{q}}$, and $\gamma_{\nu\boldsymbol{q}}$ for next Gaussian broadening value.

It is noted that only $\gamma_{\nu\boldsymbol{q}}$ is interpolated by `matdyn.x` calculation for each Gaussian broadening value. The name of this file is `elph.gamma.*`. For example, the readers can open the file `elph.gamma.1`, which corresponds to the Gaussian broadening = 0.02 Ry, as follows:

```
$ vi elph.gamma.1
```

QE-SSP/gr/elph/elph.gamma.1

```
  &plot nbnd=   6, nks=  91 /
  0.000000  0.000000  0.000000
0.0000   -0.0000    0.0000   -0.0000  321.6265  321.6265
```

```
  0.016667   0.000000   0.000000
-0.0000    -0.0715    -0.1440    -0.0000   316.9246   316.9963
...
```

- Line 1: nbnd is the number of phonon modes and nks is the number of ***q***-points.
- Line 2: The crystal coordinates (q_x, q_y, q_z) of the first ***q***-point.
- Line 3: The values of $\gamma_{\nu\boldsymbol{q}}$ for 6 phonon modes.
- Next lines: This format from line 2 to line 3 is repeated for 91 ***q***-points.

☞ **Plotting data from elph.gamma.*:** We will select the `elph.gamma.5` file to plot the phonon linewidth $\gamma_{\nu\boldsymbol{q}}$. Other files can be plotted similarly. First, the readers run the file `plotband.in` as

```
$ plotband.x < plotband.in
```

QE-SSP/gr/elph/plotband.in

```
elph.gamma.5
-30 300
elph.gamma.5.gnu
elph.gamma.5.ps
0.0
0.0 0.0
```

The `plotband.x` calculation will read the `elph.gamma.5` file, and output the `elph.gamma.5.gnu` and `elph.gamma.5.ps` files. Next, we will use the `elph.gamma.5.gnu` file to plot with the JupyterLab `gamma.ipynb` as

```
$ jupyter-lab gamma.ipynb
```

QE-SSP/gr/elph/gamma.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Convert from GHz to meV
```

```
ghz2mev = 4.13567587265e-3
# Load elph.gamma.5.gnu file
data = np.loadtxt('elph.gamma.5.gnu')
k = np.unique(data[:, 0])
gamma = np.reshape(data[:, 1], (-1, len(k)))
# Set high-symmetry points from matdyn.in
gG1 = k[0]; K = k[40]; M = k[60]; gG2 = k[90]
# Create figure object
plt.figure()
plt.axhline(0, c='gray', ls=':')
# Plot dotted lines at high-symmetry points
plt.axvline(K, c='gray')
plt.axvline(M, c='gray')
# Plot phonon linewidth
for i in range(len(gamma)):
    plt.plot(k, gamma[i, :]*ghz2mev, c='r', ls='None
        ', marker='o', markersize=7)
# Add the x and y-axis labels
plt.xlabel('')
plt.ylabel('Phonon linewidth (meV)')
# Set the axis limits
plt.xlim(gG1, gG2)
plt.ylim (-0.1, 1.1)
# Add labels for high-symmetry points
plt.xticks([gG1, K, M, gG2], ['$\Gamma$', 'K', 'M',
     '$\Gamma$'])
# Hide the x-axis minor ticks
plt.tick_params(axis='x', which='minor', bottom=
     False, top=False)
# Save the figure
plt.savefig('plot-gamma.pdf')
# Show the figure
plt.show()
```

In Fig. 3.20, we plot the phonon linewidth $\gamma_{\nu\boldsymbol{q}}$ of graphene. At the $\Gamma$ point, both the TO and LO modes show the maximum values about $\gamma_{\nu\boldsymbol{q}} = 1.0$ meV, while at the K point, only the TO shows the maximum value about $\gamma_{\nu\boldsymbol{q}} = 0.75$ meV. The maximum values of the phonon linewidth at the $\Gamma$ and K points are consistent with the observed Kohn anomaly at the $\Gamma$ point (TO and LO) and the K point (TO) of graphene, as discussed in Fig. 3.17 in Sec. 3.3.1.

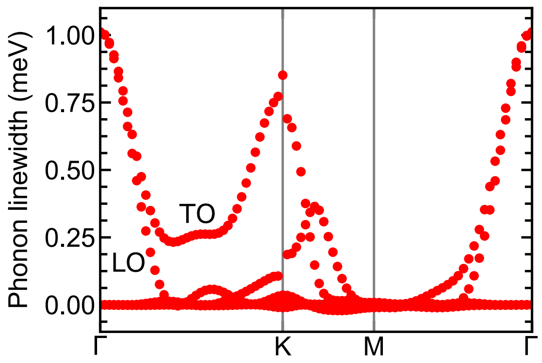


Figure 3.20 Phonon linewidth of graphene along the high symmetry-lines of the Brillouin zone.

Try It Yourself

1. Plot the phonon linewidth $\gamma_{\nu\boldsymbol{q}}$ with the Gaussian broadening about 0.02, 0.08, 0.14, and 0.2.
2. Calculate the electron-phonon coupling $\lambda_{\nu\boldsymbol{q}}$ based on $\gamma_{\nu\boldsymbol{q}}$ from the `elph.gamma.5` file by using Eq. (3.17), and plot $\lambda_{\nu\boldsymbol{q}}$ along lines between high-symmetry points in the Brillouin zone.

3.3.4 Eliashberg spectral function

❏ **Purpose:** The Eliashberg spectral function $\alpha^2F(\omega)$ is a sum over the contributions from scattering processes, in which the electrons are scattered by the phonons on the Fermi surface [Eliashberg (1960)]. $\alpha^2F(\omega)$ is an important parameter to determine the critical temperature of superconductors [McMillan (1968)]. In this tutorial, we will calculate $\alpha^2F(\omega)$ of graphene.

❏ **Background:** From Eq. (3.17) in Sec. 3.3.3, the isotropic Eliashberg spectral function [Eliashberg (1960)] can be obtained by taking an average over the Brillouin zone (BZ) as

$$\alpha^2F(\omega) = \frac{1}{2}\sum_{\nu}\int_{\mathrm{BZ}}\frac{\mathrm{d}\boldsymbol{q}}{\Omega_{\mathrm{BZ}}}\omega_{\nu\boldsymbol{q}}\lambda_{\nu\boldsymbol{q}}\delta(\omega-\omega_{\nu\boldsymbol{q}}). \tag{3.18}$$

The parameter λ, which is a dimensionless measure of the strength of $\alpha^2F(\omega)$, can also be defined as

$$\lambda = 2\int\frac{\alpha^2F(\omega)}{\omega}\mathrm{d}\omega = \sum_{\nu\boldsymbol{q}}\lambda_{\nu\boldsymbol{q}}. \tag{3.19}$$

The superconducting critical temperature T_c using the McMillan formula as a function of λ [McMillan (1968)] is given by

$$T_c = \frac{\omega_{\ln}}{1.2}\exp\left[\frac{-1.04(1+\lambda)}{\lambda(1-0.62\mu^*)-\mu^*}\right], \tag{3.20}$$

where $\omega_{\ln}$ is defined by [Dynes (1972)]

$$\omega_{\ln} = \exp\left[\frac{2}{\lambda}\int\frac{\mathrm{d}\omega}{\omega}\alpha^2F(\omega)\log(\omega)\right], \tag{3.21}$$

and μ^* in Eq. (3.20) is an empirical parameter that describes the Coulomb screening. μ^* has typical values between 0.1 and 0.16.

Since $\alpha^2F(\omega)$ is obtained by using $\lambda_{\nu\boldsymbol{q}}$ in Eq. (3.17), the readers are asked to finish the electron-phonon calculation in Sec. 3.3.3 before this tutorial.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
$ cd ~/QE-SSP/gr/alpha/
$ cp -rf ../elph/elph_dir/ ../elph/gr.fc ./
$ mpirun -np 4 matdyn.x < matdyn.in> matdyn.out &
```

- Line 1: Go to `alpha` directory.
- Line 2: Copy the `elph_dir` directory, which contains $\gamma_{\nu\boldsymbol{q}}$ and $\lambda_{\nu\boldsymbol{q}}$ in Sec. 3.3.3, and `gr.fc`, which contains the inter-atomic force constants in the real space, to the current direction (`./`).
- Line 3: Run `matdyn.x` with the input file `matdyn.in` to obtain $\alpha^2F(\omega)$ and λ.

❑ **How to check:** When the calculations finish, you can find a message `JOB DONE` at the end of each output file.
❑ **Input file:** The input file `matdyn.in` is same as Sec. 3.3.2. We need to add the option `la2F = .true.` and select `ndos = 100` (number of energy steps for DOS calculation) in the namelist `INPUT`.
❑ **Output file:** The Eliashberg spectral function $\alpha^2F(\omega)$ is given by the output files `a2F.dos*` for each Gaussian broadening. For example, the readers can open the file `a2F.dos1` for the Γ point as follows:

```
$ vi a2F.dos1
```

QE-SSP/gr/alpha/a2F.dos1

```
1 # Eliashberg function a2F (per both spin)
2 #  frequencies in Rydberg
3 # DOS normalized to E in Rydberg: a2F_total, a2F(mode)
4
5 0.739762E-04  -0.382895E-04   0.000000E+00  -0.291145E-04
       -0.917506E-05   0.000000E+00   0.000000E+00   0.000000E
       +00
6 0.221929E-03  -0.344606E-03  -0.105875E-29  -0.262031E-03
       -0.825757E-04   0.000000E+00   0.000000E+00   0.000000E
       +00
7 ...
```

- Lines 1–3: The head of the a2F.dos1 file.
- Lines 5–6: The 1st column is ω in Ry unit, the 2nd column is the total of $\alpha^2F(\omega)$, the 3rd–7th columns are the $\alpha^2F(\omega)$ for the each phonon mode ν from 1 to 6.
- Next lines: The format from lines 5 and 6 is repeated until the last value of ω.

☞ **Plotting data from a2F.dos*:** The readers can plot the file `a2F.dos*` as follows:

```
$ jupyter-lab alpha.ipynb
```

QE-SSP/gr/alpha/alpha.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
```

```

# Convert from Ry to cm^-1
ry2cm = 109737.07176

# Load data for the broadening of 0.1 Ry
omega1, aF1 = np.genfromtxt('a2F.dos5', usecols
     =(0,1), skip_footer=1, unpack=True)
# Load data for the broadening of 0.2 Ry
omega2, aF2 = np.genfromtxt('a2F.dos10', usecols
     =(0,1), skip_footer=1, unpack=True)

# Create figure object
plt.figure()
# Plot the JDOS
plt.plot(omega1*ry2cm, aF1, c='b', label='Broadening
      0.1 Ry')
plt.plot(omega2*ry2cm, aF2, c='r',ls='--', label='
     Broadening 0.2 Ry')
# Add the x and y-axis labels
plt.xlabel(r'$\omega$ (cm$^{-1}$)')
plt.ylabel(r'$\alpha^2 F(\omega)$')
# Add the the legend
plt.legend(loc='upper left')
# Set the axis limits
plt.xlim(0, 1700)
plt.ylim(0, 0.45)
# Save a figure to the pdf file
plt.savefig('plot-aF.pdf')
# Show plot
plt.show()
```

In Fig. 3.21, we plot the total $\alpha^2F(\omega)$ as a function of the phonon frequency $\omega$ of graphene with the Gaussian broadening of 0.1 Ry (solid line) and 0.2 Ry (dashed line). As shown in Fig. 3.20, the TO and LO modes show the highest electron-phonon coupling at the Γ and the K points, which correspond to $\omega = 1500$ cm$^{-1}$ and 1400 cm$^{-1}$, respectively (see Fig. 3.17). Moreover, the phonon DOS in Fig. 3.19 also shows highest value around $\omega = 1500$ cm$^{-1}$. Therefore, $\alpha^2F(\omega)$ becomes a highest value around $\omega = 1500$ cm$^{-1}$, as shown in Fig. 3.21. It is noted that the calculated result of $\alpha^2F(\omega)$ might not be converged since $\alpha^2F(\omega)$ converges very slowly with increasing size of the $\boldsymbol{k}$-point and $\boldsymbol{q}$-point grids. The readers should check the convergence by increasing $\boldsymbol{k}$-point and $\boldsymbol{q}$-point grids, and it is time-consuming for electron-phonon calculation.

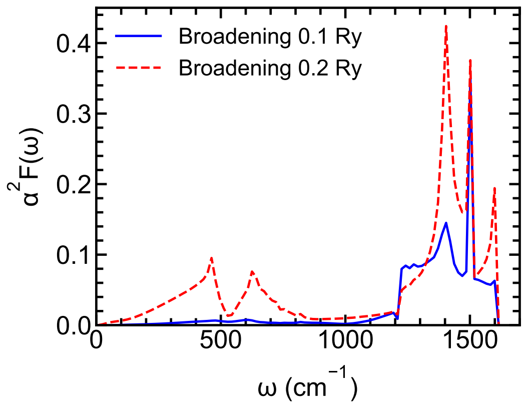


Figure 3.21 Electron-phonon Eliashberg spectral function $\alpha^2F(\omega)$ of graphene with the Gaussian broadening of 0.1 Ry (solid line) and 0.2 Ry (dashed line).

For the parameters λ and ω_{ln}, they are given by the output file `lambda` for each Gaussian broadening. The readers can open `lambda` as follows:

```
$ vi lambda
```

QE-SSP/gr/alpha/lambda

```
Electron-phonon coupling constant, lambda

Broadening   0.0200 lambda   0.1516 dos(Ef)   0.1229
      omega_ln [K]   3612.2615
Broadening   0.0400 lambda   0.0507 dos(Ef)   0.2428
      omega_ln [K]   3600.9008
...
```

This output file shows that $\lambda = 0.1517$ and $\omega_{\text{ln}} = 3611.6988$ K at the broadening of 0.02 Ry. By inserting these values and the constant $\mu^* = 0.1$ into Eq. (3.20), we obtain the critical temperature $T_c \sim 0$ for the graphene. This is because that graphene has the DOS $=$ 0 at the Fermi energy. Thus, it is not good for superconductivity. Nevertheless, graphene can become a superconductor by heavily electron-doped or twisted bilayer graphene. For example, Ca-intercalated graphite (CaC6) is a superconductor with $T_c = 11.5$

K [Emery *et al.* (2009)]. Cao *et al.* [Cao *et al.* (2018)] showed that bilayer graphene with a twist angle of about 1.1° shows T_c of up to 1.7.

Try It Yourself

1. Calculate the $\alpha^2 F(\omega)$ and T_c for the 3D Pb. The 3D Pb has the structure of face centered cubic and $T_c = 7.2$ K.
2. Explain why is T_c of the 3D Pb higher than that of graphene?

3.4 Optical properties

In this section, we learn how to calculate the optical properties of solids, such as optical absorption spectra and non-resonant Raman spectra. We select monolayer MoS_2 as an example for this section since it is a semiconductor. It is noted that some optical calculations in Quantum ESPRESSO work only for semiconductors or insulators. This is because the metallic system has an infinite dielectric constant at $\omega = 0$, which is required in some calculations.

3.4.1 Dielectric function and absorption spectra

❑ **Purpose:** In this tutorial, we show how to obtain the optical absorption of the monolayer MoS_2.
❑ **Background:** The optical absorption spectra $\alpha(\omega)$ can be calculated by using the real and imaginary parts of the dielectric function $\varepsilon(\omega)$, as shown in Eq. (5.32) in Sec. 5.9. $\varepsilon(\omega)$ is calculated by using the the executable `epsilon.x` in Quantum ESPRESSO. `epsilon.x` is based on the single-particle approximation.[8]
☞**Note:** `epsilon.x` supports for only the norm-conserving pseudopotentials.
❑ **How to run:** To run this tutorial, the readers should type the following command lines:

[8] In optical absorption, we have either single-particle excitation of an electron or collective excitations of electrons. In single-particle approximation, we calculate a sum of independent excitation of an electron from occupied state to empty states.

```
1 $ cd ~/QE-SSP/mos2/optic/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
4 $ mpirun -np 4 epsilon.x < epsilon.in > epsilon.out
      &
```

- Line 1: Go to `optic` directory.
- Line 2: Run SCF by using `pw.x` with the input file `scf.in`.
- Line 3: Run non-SCF by using `pw.x` with the input file `nscf.in`. It is noted that the non-SCF calculation might take about 20 minutes.
- Line 4: Run `epsilon.x` with the input file `epsilon.in`.

❑ **How to check:** When calculation finishes, a message `JOB DONE` is written at the end of each output file `scf.out`, `nscf.out`, and `epsilon.out`.

❑ **Input file:** The readers can use `vi` editor to open the input file `scf.in` as follows:

```
$ vi scf.in
```

QE-SSP/mos2/optic/scf.in

```
&CONTROL
calculation       = 'scf'
pseudo_dir        = '../pseudo/'
outdir            = '../tmp/'
prefix            = 'mos2'
/
&SYSTEM
ibrav             = 4
a                 = 3.1378055413
c                 = 20.0
nat               = 3
ntyp              = 2
ecutwfc           = 80.0
/
&ELECTRONS
mixing_beta       = 0.7
conv_thr          = 1.0d-10
/
ATOMIC_SPECIES
Mo   95.94   Mo.pz-hgh.UPF
S    32.065  S.pz-hgh.UPF
ATOMIC_POSITIONS (crystal)
```

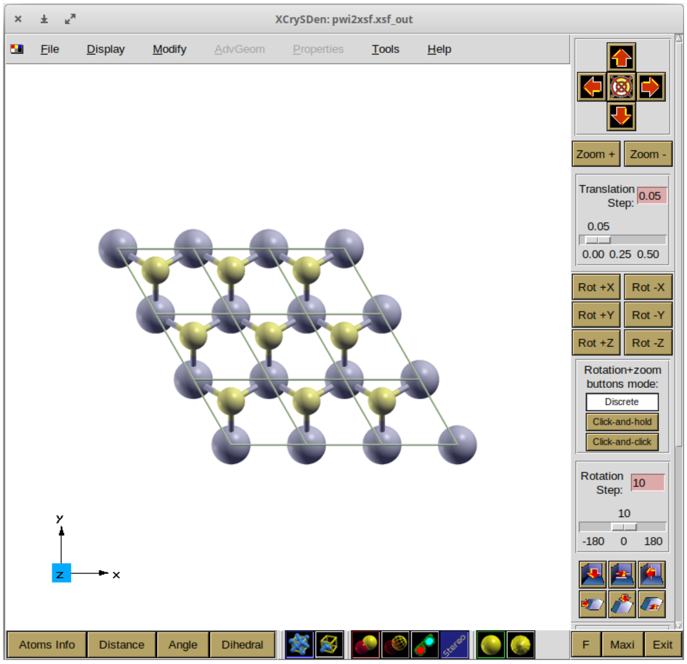


Figure 3.22 Atomic structure of monolayer MoS_2 in a $3 \times 3 \times 1$ supercell.

```
Mo   -0.0000000000   -0.0000000000    0.5000000000
S     0.3333333333    0.6666666667    0.4218571774
S     0.3333333333    0.6666666667    0.5781427066
K_POINTS (automatic)
6 6 1 0 0 0
```

☞ **Visualizing structure from scf.in:** By using the XCrySDen software to open the `scf.in` file, the readers can see the structure of the monolayer MoS_2, as shown in Fig. 3.22.

For the input file `nscf.in`, the readers can see it by typing as follows:

```
$ vi nscf.in
```

QE-SSP/mos2/optic/nscf.in

```
 1 &CONTROL
 2 calculation      = 'nscf'
 3 pseudo_dir       = '../pseudo/'
 4 outdir           = '../tmp/'
 5 prefix           = 'mos2'
 6 /
 7 &SYSTEM
 8 ibrav            = 4
 9 a                = 3.1378055413
10 c                = 20.0
11 nat              = 3
12 ntyp             = 2
13 nbnd             = 40
14 ecutwfc          = 80.0
15 nosym            = .true.
16 noinv            = .true.
17 /
18 &ELECTRONS
19 mixing_beta      = 0.7
20 conv_thr         = 1.0d-10
21 /
22 ATOMIC_SPECIES
23 Mo  95.94   Mo.pz-hgh.UPF
24 S   32.065  S.pz-hgh.UPF
25 ATOMIC_POSITIONS (crystal)
26 Mo  -0.0000000000  -0.0000000000   0.5000000000
27 S    0.3333333333   0.6666666667   0.4218571774
28 S    0.3333333333   0.6666666667   0.5781427066
29 K_POINTS (automatic)
30 24 24 1 0 0 0
```

☞**Explanation of nscf.in:** It is noted that `epsilon.x` does not support the reduction of the ***k***-points grid into the irreducible Brillouin zone.[9] Therefore, the non-SCF calculation must be performed with a uniform ***k***-points grid, and all ***k***-points weights must be equal to each other. Since the auto-symmetrization of ***k***-points grid in the namelist `K_POINTS (automatic)` can produce a non-uniform distribution of ***k***-points weights, we need to set `nosym = .true.` and `noinv = .true.` (lines 16 and 17), respectively, in the namelist `SYSTEM` to disable the auto-symmetrization.

[9] Irreducible Brillouin zone is the first Brillouin zone, which is reduced by all symmetries in the point group of the lattice.

☞**Note:** For the non-SCF calculations, the ***k***-points grid should be checked carefully to obtain the convergence of the dielectric function. In particular, for graphene, an extremely high density of ***k***-points is needed to obtain accuracy near the Dirac point (for low transition energies) because the Fermi surface is reduced to a dot. A ***k***-points grid of $600 \times 600 \times 1$ might be possible to get a good result. It is important to note that the default maximum number of the ***k***-points in Quantum ESPRESSO is 40,000. In order to run with the number of ***k***-points $> 40,000$, the readers need to change the variable "npk" in `Modules/parameters.f90` and recompile the Quantum ESPRESSO package.

For the input file `epsilon.in`, the readers can open as follows:

```
$ vi epsilon.in
```

QE-SSP/mos2/optic/epsilon.in

```
 1 &INPUTPP
 2 outdir      = '../tmp/'
 3 prefix      = 'mos2'
 4 calculation = 'eps'
 5 /
 6 &ENERGY_GRID
 7 smeartype   = 'gauss'
 8 intersmear  = 0.15
 9 intrasmear  = 0.0
10 wmin        = 0.0
11 wmax        = 10.0
12 nw          = 500
13 shift       = 0.0
14 /
```

☞**Explanation of epsilon.in:** The dielectric function $\varepsilon(\omega)$ is calculated by using the keyword `calculation = 'eps'` (line 4) in the namelist `INPUTPP`. The type of broadening of syntax `smeartype` (line 7) can be `gauss` or `lorentz` for a Gaussian or Lorentzian broadening, respectively. `intersmear` and `intrasmear` (lines 8 and 9), respectively, are the broadening parameters in eV for the interband and intraband transitions, respectively. It is noted that we do not use `intrasmear` for the semiconductor, but it is mandatory for a metal, like graphene. The readers should try several values for `intersmear` and `intrasmear` to find a best-converged value. A good value should be between 0.1 and 0.2 eV. [`wmin`, `wmax`] is the range of values for the photon energy (in units of eV), which is given by $\hbar\omega$, where $\hbar$ is the reduced Planck constant and ω is the angular frequency of light. `nw` (line 12) is the number of points of the photon energy mesh. Finally, `shift` (line 13) is an optional rigid shift (in units of eV) of the imaginary part of the dielectric function. `shift` is used when we want to correct the calculated energy band gap with an experimental value.

☞**Note:** `outdir` and `prefix` for `epsilon.in` must be the same as that for `scf.in` and `nscf.in`.

❑ **Output file:** There are four output files `epsr_mos2.dat`, `epsi_mos2.dat`, `eels_mos2.dat`, and `ieps_mos2.dat`. The first and second files contain the real $\mathrm{Re}(\varepsilon(\omega))$ and imaginary $\mathrm{Im}(\varepsilon(\omega))$ parts of diagonal part of the dielectric tensor, respectively, as a function of photon energy $\hbar\omega$ in eV. The third file contains the electron energy loss spectra, $\mathrm{Im}(1/\varepsilon(\omega))$, calculated from the diagonal elements of the dielectric tensor. The last file contains the diagonal components of the dielectric tensor, $\varepsilon(i\omega)$, which is calculated on the imaginary axis of frequency. Let us open the file `epsr_mos2.dat` as follows: ❑ **Output file:** There are four output files `epsr_mos2.dat`, `epsi_mos2.dat`, `eels_mos2.dat`, and `ieps_mos2.dat`. The first and second files two contain the real $\mathrm{Re}(\varepsilon(\omega))$ and imaginary $\mathrm{Im}(\varepsilon(\omega))$ parts of diagonal part of the dielectric tensor, respectively, as a function of photon energy $\hbar\omega$ in eV. The third file contains the electron energy loss spectra, $\mathrm{Im}(1/\varepsilon(\omega))$, calculated from the diagonal elements of the dielectric tensor. The last file contains the diagonal components of the dielectric tensor, $\varepsilon(i\omega)$, which is calculated on the imaginary axis of frequency. Let us open the file `epsr_mos2.dat` as follows:

```
$ vi epsr_mos2.dat
```

QE-SSP/gr/optic/epsr_mos2.dat

```
# energy grid [eV]     epsr_x   epsr_y   epsr_z
#
0.000000000  4.485200865  4.485200862  2.930873960
0.020040080  4.485316552  4.485316549  2.930898420
0.040080160  4.485663685  4.485663682  2.930971808
```

```
...
```

- Column 1: The photon energies, $\hbar\omega$, in units of eV.
- Columns 2 to 4: The real parts of the dielectric function, $\mathrm{Re}(\varepsilon_{xx}(\omega))$, $\mathrm{Re}(\varepsilon_{yy}(\omega))$, and $\mathrm{Re}(\varepsilon_{zz}(\omega))$ in the x-, y-, and z-directions, respectively.

Other output files can be opened in a similar way.

☞ **Note:** For the 2D material, we must reduce the value of $\mathrm{Re}(\varepsilon(\omega))$ and $\mathrm{Im}(\varepsilon(\omega))$ by a dimensionless c/L, where c is the height of the unit cell including the vacuum region and L is the real thickness of the 2D material. For the monolayer MoS_2, c is 20 Å in `scf.in` and $L = 6.5$ Å is given by the experiment [Eda *et al.* (2011)].

☞ **Plotting data from epsr_mos2.dat and epsi_mos2.dat:** In order to plot the dielectric function and the absorption coefficient, the readers can run the JupyterLab `eps.ipynb`, which reads both real and imaginary parts of the dielectric function:

```
$ jupyter-lab eps.ipynb
```

QE-SSP/mos2/optic/eps.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Set parameter
c = 299792458 # velocity of light (m/s)
hbar = 6.582119569e-16 # reduced Planck constant (eV
    .s)
l = 2.0/0.65 # to reduce thinkness the unit cell (2
    nm) to real thickness (0.65 nm)
# Load the real part of dielectric tensor
ener, repsx, repsy, repsz = np.loadtxt('epsr_mos2.
    dat', unpack=True)
# Load the imaginary part of dielectric tensor
ener, iepsx, iepsy, iepsz = np.loadtxt('epsi_mos2.
    dat', unpack=True)
# Absorption coefficient in x-, y-, z-directions
alphax = 2*(ener/hbar)*np.sqrt((np.sqrt((l*repsx)
    **2+(l*iepsx)**2)-l*repsx)/2)/c
alphay = 2*(ener/hbar)*np.sqrt((np.sqrt((l*repsy)
    **2+(l*iepsy)**2)-l*repsy)/2)/c
```

```
alphaz = 2*(ener/hbar)*np.sqrt((np.sqrt((1*repsz)
     **2+(1*iepsz)**2)-1*repsz)/2)/c
# Create figure object
fig, (ax1, ax3) = plt.subplots(1, 2,
     constrained_layout=True, figsize=(10, 4))
ax2 = ax1.twinx()
# Plot the epsilon
ax1.plot(ener, 1*repsx, 'b-')
ax2.plot(ener, 1*iepsx, 'r--')
# Add the x and y-axis labels
ax1.set_xlabel('Photon energy (eV)')
ax1.set_ylabel(r'Re$(\varepsilon_{xx})$', color='b')
ax1.tick_params(axis='y', labelcolor='b')
ax2.set_ylabel(r'Im$(\varepsilon_{xx})$', color='r')
ax2.tick_params(axis='y', labelcolor='r')
# Set the axis limits
ax1.set_xlim(0, 6)
ax1.set_ylim(0, 36)
ax2.set_ylim(0, 36)
# Plot the abosorption coefficient
ax3.plot(ener, alphax/10**8, c='k')
# Add the x and y-axis labels
ax3.set_xlabel('Photon energy (eV)')
ax3.set_ylabel(r'$\alpha_{xx}\times 10^8$ (1/m)')
# Set the axis limits
ax3.set_xlim(0, 6)
ax3.set_ylim(0, 1.4)
# Save the figure
plt.savefig('plot-optic.pdf')
# Show the figure
plt.show()
```

By running `eps.ipynb`, we obtain the in-plane optical properties of monolayer MoS_2, as shown in Fig. 3.23. In Fig. 3.23 (a), we plot the real part and imaginary part of isotropic dielectric function in the in-plane with $\varepsilon_{xx} = \varepsilon_{yy}$, in which x- and y-directions correspond to zigzag and armchair directions, as shown in Fig. 3.22. The dielectric function shows two peaks A and B (see Fig. 3.23 (a)) around 2 eV, which correspond to transition energies at the K point of the Brillouin zone [Li *et al.* (2014)].[10] We highly recommend readers to calculate the electronic energy dispersion of the monolayer MoS_2 (see the

[10] Quantum ESPRESSO does not support the calculation of exciton energies. Nevertheless, peaks A and B show excitation energy. This is because the correction to the transition energy from the exciton binding energy is largely offset by many-body corrections to the band gap in the DFT. Therefore, the predicted transition energy within a single-particle calculation is closer to the experiment than might be expected [Li *et al.* (2014)].

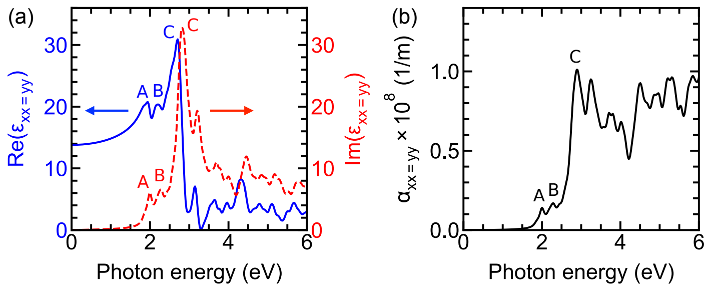


Figure 3.23 In-plane optical properties of monolayer MoS_2. (a) Real part Re(ε_{xx}) and imaginary part Im(ε_{xx}) of the dielectric tensor are plotted by the solid and dashed lines, respectively. (b) Absorption coefficient in x-direction, $\alpha_{xx=yy}$, of the monolayer MoS_2. It is noted that $\varepsilon_{xx} = \varepsilon_{yy}$ and $\alpha_{xx} = \alpha_{yy}$. The peaks labeled A and B correspond to transition energies at the K point of the Brillouin zone, and C peak corresponds to higher-lying interband transitions, including the transitions between the K and Γ points.

tutorial in Sec. 3.2.2). By checking the energy dispersion, the readers will see that the transitions occur at the K point for the photon energy of 2 eV. For the peak C in Fig. 3.23 (a), the transitions occur between the K and Γ points in the Brillouin zone. Based on Re(ε_{xx}) and Im(ε_{xx}), we can obtain the in-plane absorption coefficient α_{xx} by using the Eq. 5.32, as shown in Fig. 3.23 (b). It is noted that $\varepsilon_{xx} = \varepsilon_{yy}$ and $\alpha_{xx} = \alpha_{yy}$. The calculated α_{xx} is consistent with the experiment, in which $\alpha_{xx} \sim 0.25 \times 10^8$ 1/m for the peaks A and B, and $\alpha_{xx} \sim 1.0 \times 10^8$ 1/m for the peak C [Liu *et al.* (2020)].

Try It Yourself

1. Change the values of `nbnd` and the number of $\boldsymbol{k}$-points grid in `nscf.in` to check the convergence of the dielectric function of monolayer MoS_2.
2. Change the values of `intersmear` and check the convergence of the dielectric function of monolayer MoS_2.
3. Calculate absorption coefficient of graphene. It is noted that, for graphene, since the Fermi surface is reduced to a dot at low transition energies, an extremely high density of $\boldsymbol{k}$-points in `nscf.in` have to be used to achieve accurate results near the Fermi level.

3.4.2 Joint density of states

❑ **Purpose:** In this tutorial, we calculate the joint density of states (JDOS) to understand the peaks of the absorption spectra in Sec. 3.4.1.
❑ **Background:** As discussed in Sec. 5.9, the JDOS is the number of states for a pair of the initial and final states by given energy. Therefore, a large peak in the JDOS corresponds to the peak in the absorption spectra.[11] The JDOS is defined by Eq. (5.34), and the JDOS is obtained by using the executable `epsilon.x`.
❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/mos2/optic/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
4 $ mpirun -np 4 epsilon.x < epsilon-jdos.in > epsilon
      -jdos.out &
```

- Line 1: Go to `optic` directory.
- Line 2: Run SCF calculation by using `pw.x`.
- Line 3: Run non-SCF calculation by using `pw.x`.
- Line 4: Run JDOS calculation by using `epsilon.x`

☞ **Note:** The SCF and non-SCF calculations can be omitted if the readers have successfully finished the tutorial in Sec. 3.4.1.
❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of each output file `scf.out`, `nscf.out`, and `epsilon-jdos.out`.
❑ **Input file:** The input files `scf.in` and `nscf.in` are shown in Sec. 3.4.1. For the input file `epsilon-jdos.in`, the readers can see by typing as follows:

```
$ vi epsilon-jdos.in
```

[11] The optical absorption intensity is proportional to the product of JDOS and the square of the matrix element of the electron-photon matrix element.

QE-SSP/mos2/optic/epsilon-jdos.in

```
 1  &INPUTPP
 2  outdir      = '../tmp/'
 3  prefix      = 'mos2'
 4  calculation = 'jdos'
 5  /
 6  &ENERGY_GRID
 7  smeartype   = 'gauss'
 8  intersmear  = 0.15
 9  intrasmear  = 0.0
10  wmin        = 0.0
11  wmax        = 10.0
12  nw          = 500
13  shift       = 0.0
14  /
```

☞**Explanation of epsilon-jdos.in:** The JDOS calculation is performed by using the keyword `calculation = 'jdos'` (line 4) in the namelist `INPUTPP`. The remaining part is same as Sec. 3.4.1.

❑ **Output file:** The executable `epsilon.x` produces the file `jdos_mos2.dat`, which contains the photon energy in the first column in eV and the JDOS in the second column in states/eV/unit-cell.

☞ **Plotting data from jdos_mos2.dat:** In order to plot the JDOS of the monolayer MoS_2, the readers can run the JupyterLab `jdos.ipynb` as follows:

```
$ jupyter-lab jdos.ipynb
```

QE-SSP/mos2/optic/jdos.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Load data
ener, jdos = np.loadtxt('jdos_mos2.dat', unpack=True)
# Create figure object
plt.figure()
# Plot the JDOS
plt.plot(ener, jdos, c='b')
```

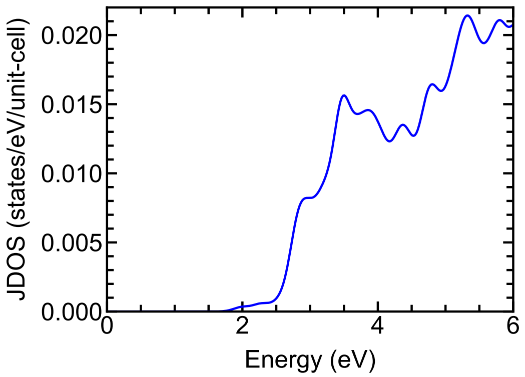


Figure 3.24 Joint density of states (JDOS) of monolayer MoS_2.

```
# Add the x and y-axis labels
plt.xlabel('Energy (eV)')
plt.ylabel('JDOS (1/eV/unit-cell)')
# Set the axis limits
plt.xlim(0, 6)
plt.ylim(0, 0.022)
# Save the figure
plt.savefig('plot-jdos.pdf')
# Show the figure
plt.show()
```

In Fig. 3.24, we show the JDOS of the monolayer MoS_2. Since the peaks A and B of the absorption coefficient in Fig. 3.23 (b) are relatively weak, we can not see the corresponding peaks in the JDOS. Nevertheless, we can see the peak in the JDOS around 3 eV, which corresponds to the peak C in Fig. 3.23 (b).

Try It Yourself

Change the values of `nbnd` and the number of $\boldsymbol{k}$-points grid in `nscf.in` to check the convergence of the JDOS of monolayer MoS_2.

3.4.3 Non-resonant Raman spectra

❑ **Purpose:** The Raman spectroscopy is a common spectroscopic technique, which is used to determine phonon modes of a solid. As discussed in Sec. 5.13, the peak positions of Raman spectra correspond to the optical phonon frequency of the Raman active mode. In this tutorial, we show how to plot the non-resonant Raman spectra of monolayer MoS_2.
❑ **Background:** The Raman calculation in Quantum ESPRESSO is implemented based on second-order response of DFT developed by Lazzeri and Mauri [Lazzeri and Mauri (2003)]. The non-resonant Raman intensities are given by Eq. (5.53) in Sec. 5.13, which can be obtained by using the executable `dynmat.x` in Quantum ESPRESSO. It is noted that the calculation is only implemented for norm-conserving pseudopotentials and LDA functional. If the material is a metal, the dielectric function will become infinite, which is not suitable for non-resonant Raman calculation, in which we use the dielectric function. Thus, the non-resonant Raman calculation works only for the semiconductor and insulator systems.
❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/mos2/raman/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 ph.x < ph.in > ph.out &
4 $ mpirun -np 4 dynmat.x < dynmat.in> dynmat.out &
```

- Line 1: Go to `raman` directory.
- Line 2: Run SCF by using `pw.x` with the input file `scf.in`.
- Line 3: Run phonon by using `ph.x` with the input file `ph.in`.
- Line 4: Run `dynmat.x` with the input file `dynmat.in`.

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of each output file `scf.out`, `ph.out`, and `dynmat.out`.
❑ **Input file:** The input file `scf.in` is shown in Sec. 3.4.1. It is noted that the option `assume_isolated = '2D'` (not shown here) is added to `scf.in` in the namelist `SYSTEM` for the phonon calculation of the 2D materials (see Sec. 3.3.1). For the input file `ph.in`, the readers can open with `vi` editor as follows:

```
$ vi ph.in
```

QE-SSP/mos2/raman/ph.in

```
 1 phonon calc.
 2 &INPUTPH
 3 outdir    = '../tmp/'
 4 prefix    = 'mos2'
 5 fildyn    = 'mos2.dmat'
 6 tr2_ph    = 1d-14
 7 lraman    = .true.
 8 epsil     = .true.
 9 trans     = .true.
10 asr       = .true.
11 /
12 0.0 0.0 0.0
```

☞**Explanation of ph.in:** The non-resonant Raman calculation is performed by setting `lraman = .true.` (line 7) in the namelist `&INPUTPH`. We also have to set `epsil = .true.` (line 8) and `trans = .true.` (line 9) to calculate the effective charges. We apply the acoustic sum rule (see Sec. 3.3.1) for the dynamical matrix by setting `asr = .true.` (line 10). Since we only consider the first-order Raman scattering, only the Γ point (`0.0 0.0 0.0`) (line 12) is calculated.

For the input file `dynmat.in`, the readers can see by typing as follows:

```
$ vi dynmat.in
```

QE-SSP/mos2/raman/dynmat.in

```
&INPUT
fildyn = 'mos2.dmat'
asr    = 'crystal'
/
```

❑ **Output file:** The output file containing the Raman intensities is `dynmat.out`. The readers can use `vi` editor to open the output as follows:

```
$ vi dynmat.out
```

QE-SSP/mos2/raman/dynmat.out

```
  IR activities are in (D/A)^2/amu units
  Raman activities are in A^4/amu units
  multiply Raman by 0.256128 for Clausius-Mossotti
       correction

# mode [cm-1]   [THz]      IR      Raman   depol.fact
1    -0.00   -0.0000    0.0000     0.0018    0.7500
2    -0.00   -0.0000    0.0000     0.2344    0.7500
3     0.00    0.0000    0.0000     0.2361    0.7500
4   283.62    8.5028    0.0000     0.0008    0.7500
5   283.62    8.5028    0.0000     0.0008    0.7500
6   384.72   11.5335    1.2891   178.7551    0.7500
7   384.72   11.5335    1.2891   178.7551    0.7500
8   403.67   12.1018    0.0000   176.8187    0.1061
9   464.86   13.9363    0.0045     0.0000    0.7500
```

For lines 6 to 14:

- Column 1: The ordering numbers of photon modes.
- Column 2: The phonon frequencies in units of cm^{-1}.
- Column 3: The infrared (IR) intensities.
- Column 4: The Raman intensities.
- Column 5: The depolarization ratio factor.[12]

☞ **Plotting data from dynmat.out:** The Raman spectra $I(\omega)$ can be plotted as a smoothed function of the frequency ω by fitting the Gaussian function, which is defined by

$$I(\omega) = I_0 \exp\left[-\left(\frac{\omega - \omega_0}{\Gamma}\right)^2\right], \tag{3.22}$$

where ω_0 and I_0 are the phonon frequencies and Raman intensities, respectively, which are obtained from the `dynmat.out` file. Γ is a broadening parameter, which can be obtained by the full width at half maximum (FWHM) in the experiment.[13] We run the JupyterLab `raman.ipynb` to plot the Raman spectra as follows:

[12] Depolarization ratio is the intensity ratio of the perpendicular component to the parallel component for the Raman peak intensity.

[13] FWHM is inversely proportional to phonon lifetime. A typical value of the FWHM is 10 cm^{-1}, whose phonon lifetime is 2 ps.

```
$ jupyter-lab raman.ipynb
```

QE-SSP/mos2/raman/raman.ipynb

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np
# Define the Gaussian function
def gaussian(w, G, w0, I0):
    return I0*np.exp(-((w-w0)/G)**2)
# Peaks list of non-resonant Raman in dynmat.out
peak =[(384.72,178.7551),(403.67,176.8187)]
# Fitting with the Gaussian function
def fit(w, G):
    fit = gaussian(w, G, 0, 0)
    for w0, I0 in peak:
        fit = fit + gaussian(w, G, w0, I0)
    return fit
w = np.linspace(300, 500, 500)
# Create figure object
plt.figure()
# Plot the non-resonant Raman
plt.plot(w, fit(w,2), c='b')
# Add the x and y-axis labels
plt.xlabel('Raman shift (cm$^{-1}$)')
plt.ylabel('Intensity (a.u.)')
# Hide y-axis minor ticks
plt.tick_params(axis='y', which='both', right=False,
      left=False, labelleft=False)
plt.tick_params(axis='x', which='both',top=False)
# Set the axis limits
plt.xlim(370, 420)
# Save the figure
plt.savefig('plot-raman.pdf')
# Show the figure
plt.show()
```

In Fig. 3.25, we plot the non-resonant Raman spectra of monolayer MoS_2. The calculated result shows the two peaks at 385 cm^{-1} and 404 cm^{-1}, which correspond to the E^1_{2g} and A_{1g} Raman active modes [Li *et al.* (2012)]. These peaks have the same intensity and it is consistent with the experimentally observed Raman spectra ($\sim$ 384 cm^{-1} and $\sim$ 403 cm^{-1} for the E^1_{2g} and A_{1g} modes, respectively [Li *et al.* (2012)]).

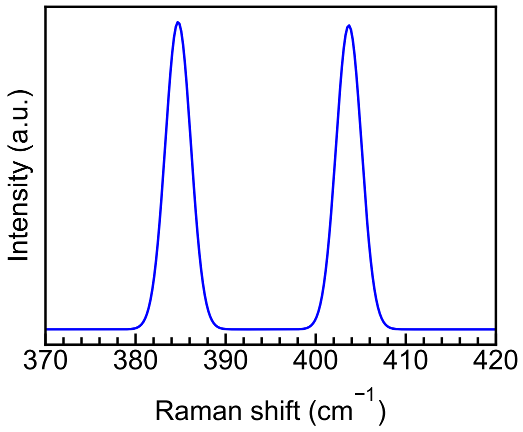


Figure 3.25 Non-resonant Raman spectra of monolayer MoS_2.

Try It Yourself

1. Change lattice constant `a` in the `scf.in` of monolayer MoS_2, and plot the Raman spectra as a function of the lattice constant.
2. Calculate non-resonant Raman spectra of bulk Si.

3.5 Subjects for two-dimensional materials

This section will calculate three important tutorials for two-dimensional (2D) materials: the spin-orbit coupling in the monolayer MoS_2, the van der Waals interaction of bilayer graphene, and an external electric field to bilayer graphene. These calculations can modify the electronic energy dispersion and the optimized structure of the materials.

3.5.1 Spin-orbit coupling

❑ **Purpose:** In Sec. 3.4, the monolayer MoS_2 is calculated without spin-orbit coupling[14] (SOC). However, it is observed that the SOC leads to the band splitting in the valence bands at the K points [Roch *et al.* (2019)] for the monolayer MoS_2. In this tutorial, we show how to calculate the band splitting of the monolayer MoS_2 in the presence of the SOC.

❑ **Background:** Since the SOC is a relativistic effect, we need to use the fully relativistic pseudopotentials (PPs) for SOC calculation. As shown in Sec. 3.1.6, the fully relativistic PPs can be obtained from the PSlibrary (`https://www.quantum-espresso.org/pseudopotentials/ps-library`). In this tutorial, we select `Mo.rel-pbe-spn-rrkjus_psl.1.0.0.UPF` and `S.rel-pbe-nl-rrkjus_psl.1.0.0.UPF` for the Mo and S atoms, respectively.

☞ **Note**: Some calculation does not support the fully relativistic PPs, for example, the optimizing structure (`relax` or `vc-relax`). Therefore, the readers should optimize the structure by using scalar relativistic PPs, such as `Mo.pbe-spn-rrkjus_psl.1.0.0.UPF` and `S.pbe-nl-rrkjus_psl.1.0.0.UPF` for the Mo and S atoms, respectively. Then, replacing the scalar relativistic PPs with the fully relativistic PPs for the SOC calculation.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/mos2/soc/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
4 $ mpirun -np 4 bands.x < bands.in> bands.out &
```

- Line 1: Go to `soc` directory.
- Line 2: Run SCF calculation by using `pw.x`.
- Line 3: Run non-SCF calculation by using `pw.x`.
- Line 4: Run `bands.x` with the input file `bands.in`.

[14] The spin-orbit coupling is a relativistic interaction between a spin and an orbital angular momentum of an electron.

❑ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of each output file `scf.out`, `nscf.out`, and `bands.out`.

❑ **Input file:** For the input file `scf.in`, the readers can open with `vi` editor as follows:

```
$ vi scf.in
```

QE-SSP/mos2/soc/scf.in

```
 1 &CONTROL
 2 calculation      = 'scf'
 3 pseudo_dir       = '../pseudo/'
 4 outdir           = '../tmp/'
 5 prefix           = 'mos2'
 6 /
 7 &SYSTEM
 8 ibrav            = 4
 9 a                = 3.1825188839
10 c                = 20.0
11 nat              = 3
12 ntyp             = 2
13 ecutwfc          = 60.0
14 noncolin         = .true.
15 lspinorb         = .true.
16 /
17 &ELECTRONS
18 mixing_beta      = 0.7
19 conv_thr         = 1.0d-6
20 /
21 ATOMIC_SPECIES
22 Mo  95.94   Mo.rel-pbe-spn-rrkjus_psl.1.0.0.UPF
23 S   32.065  S.rel-pbe-nl-rrkjus_psl.1.0.0.UPF
24 ATOMIC_POSITIONS (crystal)
25 Mo    0.0000000000   0.0000000000   0.5000000000
26 S     0.3333333333   0.6666666667   0.4217548051
27 S     0.3333333333   0.6666666667   0.5782450789
28 K_POINTS (automatic)
29 6 6 1 0 0 0
```

☞**Explanation of scf.in:** The SOC calculation is performed by setting `noncolin = .true.` and `lspinorb = .true.` (lines 14 and 15), respectively. The first setting allows a non-collinear calculation, and the second one allows using a PP with spin-orbit such as fully relativistic PP. In collinear calculation, up- or down-spin is taken into account on the calculation. In non-collinear calculation, the spin direction can be taken in any direction. If all PPs are scalar relativistic, the calculation is non-collinear, but there is no SOC.

For the input file `nscf.in`, the readers can open as follows:

```
$ vi nscf.in
```

QE-SSP/mos2/soc/nscf.in

```
&CONTROL
calculation      = 'bands'
pseudo_dir       = '../pseudo/'
outdir           = '../tmp/'
prefix           = 'mos2'
/
&SYSTEM
ibrav            = 4
a                = 3.1825188839
c                = 20.0
nat              = 3
ntyp             = 2
nbnd             = 60
ecutwfc          = 60.0
noncolin         = .true.
lspinorb         = .true.
/
&ELECTRONS
mixing_beta      = 0.7
conv_thr         = 1.0d-6
/
ATOMIC_SPECIES
Mo  95.94   Mo.rel-pbe-spn-rrkjus_psl.1.0.0.UPF
S   32.065  S.rel-pbe-nl-rrkjus_psl.1.0.0.UPF
ATOMIC_POSITIONS (crystal)
Mo   0.0000000000   0.0000000000   0.5000000000
S    0.3333333333   0.6666666667   0.4217548051
S    0.3333333333   0.6666666667   0.5782450789
K_POINTS (crystal_b)
4
gG 40
K  20
M  30
gG 0
```

☞**Note for nscf.in:** Since we consider both spin-up and spin-down of an electron, the number of Kohn-Sham states (`nbnd`) should be

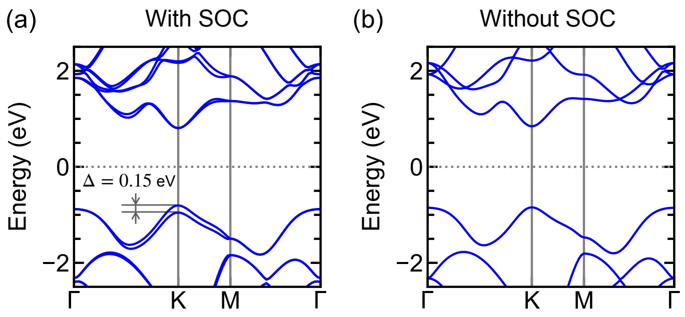


Figure 3.26 Electronic band structure of monolayer MoS_2 (a) with SOC and (b) without SOC.

selected more than twice the number of electrons. Here, we set `nbnd = 60` (line 13) since there are 26 electrons in the monolayer MoS_2, which can be found in the `scf.out` file.

For the input file `bands.in`, the readers can open as follows:

```
$ vi bands.in
```

QE-SSP/mos2/soc/bands.in

```
&BANDS
 outdir  = '../tmp/'
 prefix  = 'mos2'
 filband = 'mos2.bands'
/
```

❑ **Plotting band structure with SOC:** The band structure including the SOC is given by output file `mos2.bands.gnu`. To plot this output file, the readers can run JupyterLab `plot-bands.ipynb` as follows:

```
$ jupyter-lab plot-bands.ipynb
```

It is noted that `plot-bands.ipynb` is obtained from the Python script in page 77 (Sec. 3.2.2) by changing the filename to `mos2.bands.gnu` and the value of the Fermi energy to `-0.0925`. By running `plot-bands.ipynb`, we obtain the band structure of the monolayer MoS_2 with the SOC, as shown in Fig. 3.26 (a). Compared with the band structure without SOC in Fig. 3.26 (b), we can see that the electronic bands split by the spin-orbit interaction that is generally larger in the valence band than in the conduction band. The split bands have the energy difference, Δ, because the two electronic states with spin-up and spin-down are separated. As discussed above, we can not separate spin-up and spin-down in the band structure for non-collinear calculation. The largest value $\Delta = 0.15$ eV is found at the K point for the valence bands, which reproduces the experimental value for bulk MoS_2 ($\Delta = 0.17$ eV [Latzke *et al.* (2015)]).

Try It Yourself

1. Calculate band structure of monolayer MoS_2 without SOC by selecting `noncolin = .false.` and `lspinorb = .false.` with the scalar relativistic PPs. The band structure will be obtained as shown in Fig. 3.26 (b).
2. Calculate the spin-orbit splitting (Δ) of monolayer WSe_2, and discuss the reason why Δ of monolayer WSe_2 is larger than that of monolayer MoS_2?

3.5.2 Van der Waals interaction

❏ **Purpose:** Van der Waals (vdW) interaction is a weak attractive interaction between induced dipole moments that occurs even where there is no overlap between electron densities. Therefore, the vdW interaction is important for stacking layered materials. In this tutorial, we check the effect of the vdW interaction by calculating the interlayer distance of the bilayer graphene. We use the bilayer graphene with the AB-stacking (Bernal) structure, as shown in Fig. 3.27.

❏ **Background:** The vdW energy should be included in the total energy calculated by the exchange-correlation functional. However, the exchange-correlation functionals, such as LDA and GGA, do not describe the vdW energy since the LDA and GGA assume overlapping

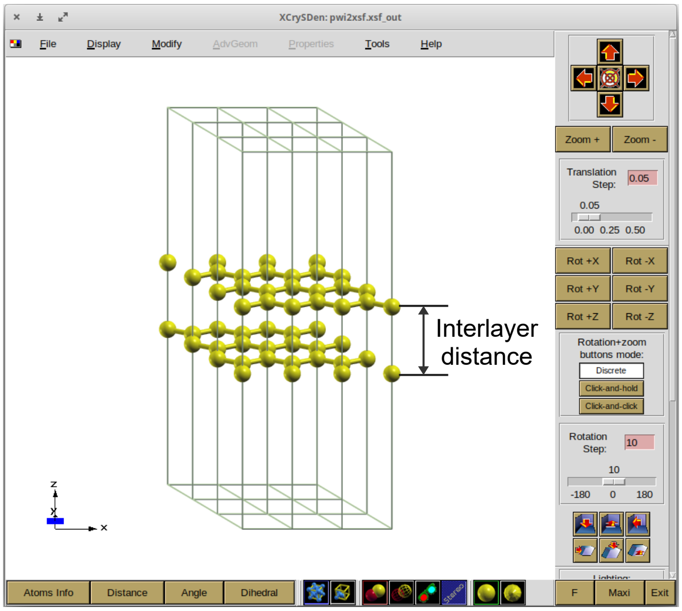


Figure 3.27 Atomic structure of bilayer graphene with the AB-stracking in a $3 \times 3 \times 1$ supercell.

electron densities (see Sec. 3.1.1). Therefore, several methods have been developed to solve this problem. Each method has a different way of calculating vdW energy from others. Details of each method are listed as follows:

DFT-D: It is empirically known that the vdW energy is described by damped inter-atomic potential, which is proportional to $1/R^6$, where R is the inter-atomic distance [Grimme (2004)]. Thus, a simple method is that we add an energy term with $1/R^6$ into the total energy as

$$E_{\text{DFT-D}} = E_{\text{DFT}} + E_{\text{vdW}}, \tag{3.23}$$

where E_{vdW} denotes the vdW energy. E_{vdW} is given by

$$E_{\text{vdW}} = -\frac{s_6}{2} \sum_{I \neq J} \frac{C_6^{IJ}}{R_{IJ}^6} f_d(R_{IJ}), \tag{3.24}$$

where s_6 is a global scaling factor depending on the specific GGA and C_6^{IJ} denotes the dispersion coefficient for atom pair IJ. The C_6^{IJ} coefficients are taken by a least-square fitting procedure from the work of Wu and Yang [Wu and Yang (2002)]. $f_d(R_{IJ})$ is a damping function, which is given by [Grimme (2004)]:

$$f_d(R_{IJ}) = \frac{1}{1 + e^{-d(R_{IJ}/R_r - 1)}}, \tag{3.25}$$

where the parameters d and R_r are fitted to experimental or accurate theoretical data. The DFT-D method is an inexpensive and straightforward calculation, but it is not a fully first-principles approach.

DFT-D3: This method is a refined model of the DFT-D method with more parameters and more interaction terms [Grimme *et al.* (2010)].

Tkatchenko-Scheffler (TS): For the TS method, the C_6^{IJ} coefficients in Eq. (3.24) are computed from the ground-state electron density and reference values for the free atoms [Tkatchenko and Scheffler (2009)].

Exchange-dipole model (EDM): The EDM shows another way to obtain the C_6^{IJ} coefficients in Eq. (3.24) based on second-order perturbation theory [Becke and Johnson (2007), Otero-De-La-Roza and Johnson (2012)], which is referred to as exchange-dipole model.

Nonlocal vdW functionals: This method replaces the vdW energy term E_{vdW} in Eq. (3.23) by a nonlocal energy functional of the electron density $n(\boldsymbol{r})$, which is given by a six-dimensional integral as

$$E_{\mathrm{vdW}}^{\mathrm{nl}}[n(\boldsymbol{r})] = \frac{1}{2} \iint n(\boldsymbol{r})\Phi(\boldsymbol{r},\boldsymbol{r}')n(\boldsymbol{r}')\mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{r}', \tag{3.26}$$

where $\Phi(\boldsymbol{r},\boldsymbol{r}')$ is a function depending on $\boldsymbol{r} - \boldsymbol{r}'$ and the densities n in the vicinity of $\boldsymbol{r}$ and $\boldsymbol{r}'$ [Dion *et al.* (2004)]. Several nonlocal vdW functionals have been proposed, such as vdW-DF [Dion *et al.* (2004), Thonhauser *et al.* (2015)], vdW-DF2 [Lee *et al.* (2010)], vdW-DF3-opt1 [Chakraborty *et al.* (2020)], vdW-DF3-opt2 [Chakraborty *et al.* (2020)], vdW-DF-C6 [Berland *et al.* (2019)], or rVV10 [Sabatini *et al.* (2013)]. A full list of the nonlocal vdW functionals that are supported by Quantum ESPRESSO can be found in Modules/funct.f90.

Since the nonlocal vdW functionals implemented in Quantum ESPRESSO are sufficiently fast and can be calculated at almost the same cost as LDA and GGA, we recommend using the nonlocal vdW functionals (vdW-DF, rVV10, etc.) for calculating the vdW interaction. Therefore, in this tutorial, we consider only the nonlocal vdW functionals to optimize the structure of bilayer graphene.

❏ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/bi-gr/vdw/
2 $ ./run.sh &
```

- Line 1: Go to `vdw` directory.
- Line 2: Run a bash script file `run.sh`, which generates and runs many jobs with changing the vdW functionals in `scf.in`.

❏ **How to check:** To check whether or not the output file exists, the readers can use the `ls` command by typing:

```
$ ls
```

The `ls` displays a listing of the all files in the `vdw` folder as

```
run.sh                     vc-relax.vdw-df3-opt1.out
vc-relax.pbe.in            vc-relax.vdw-df3-opt2.in
vc-relax.pbe.out           vc-relax.vdw-df3-opt2.out
vc-relax.rvv10.in          vc-relax.vdw-df-C6.in
vc-relax.rvv10.out         vc-relax.vdw-df-C6.out
vc-relax.vdw-df2.in        vc-relax.vdw-df.in
vc-relax.vdw-df2.out       vc-relax.vdw-df.out
vc-relax.vdw-df3-opt1.in
```

❏ **Input file:** All input files are generated automatically by a bash script file `run.sh` as

QE-SSP/bi-gr/vdw/run.sh

```
#!/bin/bash
# Set a variable vdw for 1 GGA and 6 vdW functionals
    .
```

```
for vdw in pbe vdw-df vdw-df2 vdw-df3-opt1 \
           vdw-df3-opt2 vdw-df-C6 rvv10; do
# Make input file for the vc-relax calculation.
cat > vc-relax.$vdw.in << EOF
&CONTROL
calculation    = 'vc-relax'
pseudo_dir     = '../pseudo/'
outdir         = '../tmp/'
prefix         = 'bi-gr'
etot_conv_thr = 1.0D-5
forc_conv_thr = 1.0D-4
/
&SYSTEM
ibrav          = 4
a              = 2.4639055825
c              = 20.0
nat            = 4
ntyp           = 1
occupations    = 'smearing'
smearing       = 'mv'
degauss        = 0.02
ecutwfc        = 60
input_dft      = '$vdw'
/
&ELECTRONS
mixing_beta    = 0.7
conv_thr       = 1.0D-9
/
&IONS
ion_dynamics   = 'bfgs'
/
&CELL
cell_dynamics = 'bfgs'
press_conv_thr= 0.05
cell_dofree    = '2Dxy'
/
ATOMIC_SPECIES
C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
ATOMIC_POSITIONS (crystal)
C   0.000000000   0.000000000   0.412500000
C   0.333333333   0.666666666   0.412500000
C   0.000000000   0.000000000   0.587500000
C   0.666666666   0.333333333   0.587500000
K_POINTS (automatic)
8 8 1 0 0 0
EOF
# Run pw.x for vc-relax calculation.
mpirun -np 4 pw.x <vc-relax.$vdw.in> vc-relax.$vdw.
    out
```

```
# End of for loop.
done
```

☞ **Explanation of run.sh:** Six nonlocal vdW functionals (vdW-DF, vdW-DF2, vdW-DF-opt1, vdW-DF-opt2, vdW-DF-C6, and rVV10) are calculated in this tutorial. These vdW functionals are controlled by setting `input_dft = $vdw` (line 26) in the namelist `SYSTEM`, in which the values of `$vdw` (line 3) are selected as a loop of `vdw`. For the case of `input_dft = pbe`, the bilayer graphene is calculated by the GGA without the vdW correction. It is noted that, in order to apply other methods DFT-D, DFT-D3, TS, and EMD, `input_dft = $vdw` should be replaced by `vdw_corr = $vdw` with `vdw = dft-d`, `dft-d2`, `ts`, and `emd`, respectively.
☞**Note:** If the readers are using the Quantum ESPRESSO version $<$ 6.5, the readers might get an error in the output as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    from read_kernel_table : error #         1
    No \"vdW_kernel_table\" file could be found
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

In order to solve this error, the readers need to run: `<Quantum ESPRESSO-PATH>/bin/generate_vdW_kernel_table.x` to generate the `vdW_kernel_table` file. For the the Quantum ESPRESSO version $>$ 6.5, the `vdW_kernel_table` is no longer needed for calculation.
❏ **Output file:** The interlayer distance d can be obtained from the optimized structures from the output files `vc-relax.*.out`. In Fig. 3.28, we show $d - d_{\text{exp}}$ for the PBE and the vdW functionals, in which $d_{\text{exp}} = 3.48$ Å is taken by the experimental value of bilayer graphene [Razado-Colambo *et al.* (2018)]. Without the vdW correction (or the PBE case), d is larger than the experimental value $d_{\text{exp}} = 3.48$ Å about 0.77 Å. With the vdW correction, d becomes close to d_{exp}, and the closest value is 0.06 Å of vdW-DF2. We note that even vdW-DF2 shows a good vdW functional for the case of bilayer graphene, there is no strict guideline for the proper use of the vdW functionals. Therefore, for any system, the readers should carefully select several vdW functionals to find the suitable one.

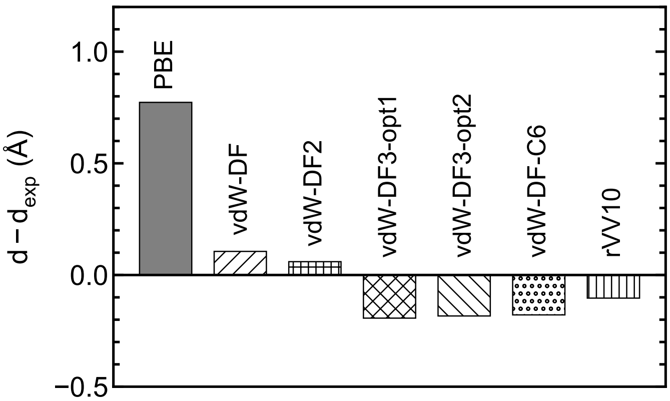


Figure 3.28 Interlayer distance d of bilayer graphene is calculated with PBE and 6 nonlocal vdW functionals. The experimental value d_{exp} of bilayer graphene is 3.48 Å [Razado-Colambo *et al.* (2018)].

Try It Yourself

Calculate the interlayer distance d of the 3D MoS_2 with the vdW correction, and find the suitable vdW functional for the 3D MoS_2. The experimental value of d for MoS_2 is 6.15 Å [Rasamani *et al.* (2017)].

3.5.3 External electric field

❑ **Purpose:** The bilayer graphene in Sec. 3.5.3 has a zero band gap. However, the band gap can be tuned by applying an external electric field perpendicular to the graphene layer, in which a band gap is opened at the Dirac point. In this tutorial, we show how to obtain the electrostatic potential and the band structure of the bilayer graphene under the external electric field.

❑ **Background:** In Quantum ESPRESSO, an external electric field E can be applied for a 2D system by using a saw-tooth potential $\mathcal{V}_{ext}$, as shown in Fig. 3.29. Assuming that the 2D material is set to be in the

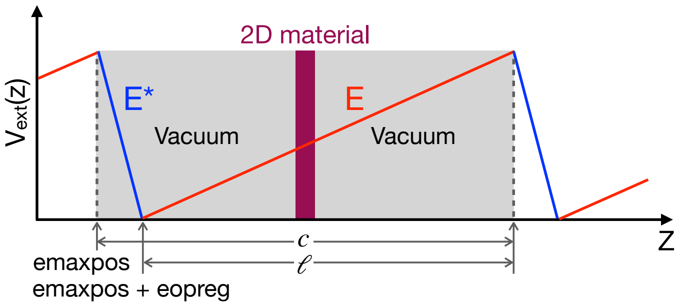


Figure 3.29 Schematics of saw-tooth potential. The dark area represents the 2D material and the light gray spaces are the vacuums in the unit cell. The solid lines with labels E and E^* denote the linear external electric potential and its counterpart, respectively. c is the size of the periodic cell along z direction, and ℓ is the length of external electric potential. `emaxpos` and `eopreg` are input parameters for Quantum ESPRESSO.

xy plane and E is applied in the z direction, $\mathcal{V}_{ext}$ is defined by

$$\mathcal{V}_{ext}(z) = -eEz. \tag{3.27}$$

If $\mathcal{V}_{ext}(z)$ changes monotonically, $\mathcal{V}_{ext}(z)$ would violate periodic boundary conditions. In order to satisfy periodic boundary conditions, we need to change the electric field in some irrelevant region (e.g., in the vacuum far from any surfaces), so that the potential restores its original value on the boundary. In Fig. 3.29, we show the schematic of the saw-tooth potential $\mathcal{V}_{ext}(z)$ in the z direction with the unit cell containing 2D material (shaded area). As shown in Fig. 3.29, the external electric field E changes abruptly in the vacuum. The saw-tooth potential also leads to an artificial field E^*, which is always opposite to E and given by

$$E^* = -E\frac{\ell}{c-\ell}, \tag{3.28}$$

where ℓ and c are the length of the external electric field E and the size of the unit cell in the z direction, respectively.

We note that the unit cell of the 2D material contains a dipole moment in the vacuum space. In particular, when the bottom and top layers of the 2D material are not equivalent to each other, there is a dipole moment between these layers due to the periodic boundary conditions. This dipole moment introduces an artificial electric field in the z direction of the unit cell. In Quantum ESPRESSO, a dipole correction can be applied to suppress the artificial dipole field by introducing an additional ramp-shaped potential $\mathcal{V}_{dip}$. Assuming that the 2D material is set to be in the xy plane, $\mathcal{V}_{dip}$ is given by [Bengtsson (1999)]:

$$\mathcal{V}_{dip}(z) = 4\pi m \left(\frac{z}{c} - \frac{1}{2} \right), \tag{3.29}$$

where m is a surface dipole moment. The dipole correction introduces a jump in electrostatic potential which should be placed within the vacuum region of the cell.

❏ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/bi-gr/elec-field/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 pp.x < pp.in > pp.out &
4 $ average.x < average.in > average.out &
5 $ mpirun -np 4 pw.x < nscf.in > nscf.out &
6 $ mpirun -np 4 bands.x < bands.in > bands.out &
```

- Line 1: Go to `elec-field` directory.
- Line 2: Run SCF calculation by using `pw.x`.
- Line 3: Run `pp.x` to obtain the electrostatic potential.
- Line 4: Run `average.x` to obtain the planar-average potential along the z direction from the electrostatic potential. It is noted that this program supports for *only single processor*.
- Line 5: Run non-SCF calculation by using `pw.x`.
- Line 6: Run `bands.x` to obtain the band structure.

❏ **How to check:** When the calculations finish, a message `JOB DONE` is written at the end of each output file `scf.out`, `pp.out`, `average.out`, `nscf.out`, and `bands.out`.

❏ **Input file:** For the input file `scf.in`, the readers can open with `vi` editor as follows:

```
$ vi scf.in
```

QE-SSP/bi-gr/elec-field/scf.in

```
 1 &CONTROL
 2 calculation    = 'scf'
 3 pseudo_dir     = '../pseudo/'
 4 outdir         = '../tmp/'
 5 prefix         = 'bi-gr'
 6 tefield        = .true.
 7 dipfield       = .true.
 8 /
 9 &SYSTEM
10 ibrav          = 4
11 a              = 2.4857910097
12 c              = 20.0
13 nat            = 4
14 ntyp           = 1
15 occupations    = 'smearing'
16 smearing       = 'mv'
17 degauss        = 0.02
18 ecutwfc        = 80
19 input_dft      = 'vdw-df2'
20 edir           = 3
21 emaxpos        = 0
22 eopreg         = 0.05
23 eamp           = 0.0097234527
24 /
25 &ELECTRONS
26 mixing_beta    = 0.7
27 conv_thr       = 1.0D-9
28 /
29 ATOMIC_SPECIES
30 C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
31 ATOMIC_POSITIONS (crystal)
32 C   0.0000000000   0.0000000000   0.4115083725
33 C   0.3333333333   0.6666666667   0.4114773543
34 C   0.0000000000   0.0000000000   0.5884916275
35 C   0.6666666667   0.3333333333   0.5885226457
36 K_POINTS (automatic)
37 8 8 1 0 0 0
```

☞**Explanation of scf.in:** The external electric field and the dipole correction are performed by setting `tefield = .true.` and `dipfield = .true.` (lines 6 and 7), respectively, in the namelist `CONTROL`. For the direction of the electric field or dipole correction is parallel to the *z*-direction, we set `edir = 3` (line 20). The atomic structure (lines 32–35) of the bilayer graphene was optimized by using the van der Walls correction (vdW-DF2), as shown in Sec. 3.5.2,

in which the positions of the C atoms are located at the center of the unit cell. Thus, as shown in Fig. 3.29, the position of the maximum of the saw-tooth potential will be set to 0 by `emaxpos = 0` (line 21). We set the width of the window, where the saw-tooth potential decreases, about 1 Å by `eopreg = 0.05` (line 22). We note that `emaxpos` and `eopreg` have the unit of $c = 20$ Å. The magnitude of the electric field is set to 0.5 V/Å by `eamp = 0.0097234527` (line 23), in which `eamp` has the unit of a.u. (1 V/Å= 0.0194469054 a.u.).
☞**Note:** For the external electric field, the readers should use a large value of `ecutwfc` (line 18). For example, if the readers use `ecutwfc = 60`, the readers might get an error in `scf.out` as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     Error in routine electrons (1):
     charge is wrong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

For the input file `pp.in`, the readers can open as follows:

```
$ vi pp.in
```

QE-SSP/bi-gr/elec-field/pp.in

```
1 &INPUTPP
2 outdir   = '../tmp/'
3 prefix   = 'bi-gr'
4 filplot  = 'bi-gr.dat'
5 plot_num = 11
6 /
```

☞**Explanation of pp.in:** In order to plot the electrostatic potential that considers the bare potential (local part of the ionic potential) and Hartree potential, we set `plot_num = 11` (line 5). The name of the output file of the potential is set by `filplot = 'bi-gr.dat'` (line 4).

For the input file `average.in`, the readers can open as follows:

```
$ vi average.in
```

QE-SSP/bi-gr/elec-field/average.in

```
1
bi-gr.dat
1.0
300
3
3.0
```

☞**Explanation of average.in:** The program `average.x` from Quantum ESPRESSO is used to calculate the planar- and macroscopic-average potential from the potential file `bi-gr.dat` that is generated by `pp.x`. The input variables of `average.in` are given as follows:

- Line 1: The number of files containing the desired quantities.
- Line 2: The name of the file.
- Line 3: The weight of the quantity read from file.
- Line 4: The number of points for the final interpolation of the planar- and macroscopic-average.
- Line 5: The direction (1, 2, and 3 correspond to *x*-, *y*-, and *z*-direction, respectively), in which the planar-average is calculated in the plane orthogonal to this direction.
- Line 6: The size of the window for macroscopic average (a.u.).

For the input file `nscf.in`, the readers can open as follows:

```
$ vi nscf.in
```

QE-SSP/bi-gr/elec-field/nscf.in

```
&CONTROL
calculation    = 'bands'
pseudo_dir     = '../pseudo/'
outdir         = '../tmp/'
prefix         = 'bi-gr'
tefield        = .true.
dipfield       = .true.
/
&SYSTEM
ibrav          = 4
a              = 2.4857910097
c              = 20.0
nat            = 4
ntyp           = 1
```

```
nbnd            = 30
occupations     = 'smearing'
smearing        = 'mv'
degauss         = 0.02
ecutwfc         = 80
input_dft       = 'vdw-df2'
edir            = 3
emaxpos         = 0
eopreg          = 0.05
eamp            = 0.0097234527
/
&ELECTRONS
mixing_beta     = 0.7
conv_thr        = 1.0D-9
/
ATOMIC_SPECIES
C 12.0107 C.pbe-n-rrkjus_psl.0.1.UPF
ATOMIC_POSITIONS (crystal)
C   0.0000000000   0.0000000000   0.4115083725
C   0.3333333333   0.6666666667   0.4114773543
C   0.0000000000   0.0000000000   0.5884916275
C   0.6666666667   0.3333333333   0.5885226457
K_POINTS (crystal_b)
4
gG  40
K   20
M   30
gG  0
```

For the input file `nscf.in`, the readers can open as follows:

```
$ vi bands.in
```

QE-SSP/bi-gr/elec-field/bands.in

```
&BANDS
 outdir  = '../tmp/'
 prefix  = 'bi-gr'
 filband = 'bi-gr.bands'
/
```

❏ **Output file:** The planar-average of electrostatic potential is written in `avg.dat`, which is output file of `average.x`. The readers can use `vi` editor to see the `avg.dat` file as following:

```
$ vi avg.dat
```

QE-SSP/bi-gr/elec-field/avg.dat

```
    0.000000000     0.550530876     0.441707911
    0.125981742     0.519365693     0.424317560
    0.251963483     0.477007077     0.405271088
    0.377945225     0.443319530     0.384610570
...
```

- Column 1: The coordinate (a.u) along *z* direction.
- Column 2: The planar-averaged potential in units of Ry.
- Column 2: The macroscopic-averaged potential in units of Ry.

❑ **Plotting data from avg.dat:** The planar-averaged electrostatic potential is plotted by running `plot-potential.ipynb`, which reads the data from the `avg.dat` file:

```
$ jupyter-lab plot-potential.ipynb
```

QE-SSP/bi-gr/elec-field/plot-potential.ipynb

```python
# Import the necessary packages and modules
import matplotlib.pyplot as plt
plt.style.use('../../matplotlib/sci.mplstyle')
import numpy as np

# Load data
z, V = np.loadtxt('avg.dat', usecols=(0,1), unpack=
    True)
# Convert a.u. to Angstrom
au2a = 0.529177249

# Create figure object
plt.figure(figsize=(4.5,4.5))
# Plot the data, using blue color
plt.plot(z*au2a, V, c='b')
# Plot a dashed line at zero
plt.axhline(0, c='gray', ls='--')
# Set the axis limits
plt.xlim(0, 20)
plt.ylim(-3, 1)
# Add the x and y-axis labels
plt.xlabel('z ($\AA$)')
plt.ylabel('Electrostatic potential (Ry)')
```

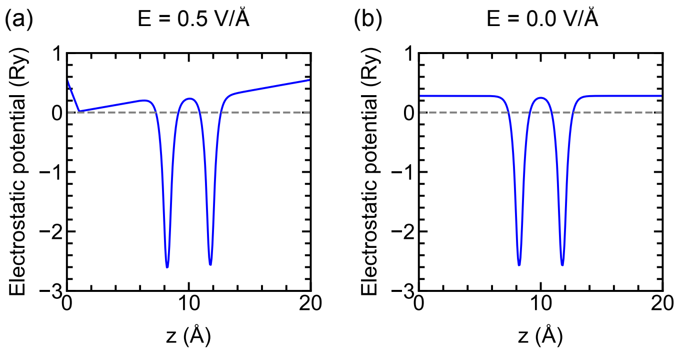


Figure 3.30 Planar-averaged electrostatic potential along the z direction of bilayer graphene with an external electric field E of (a) 0.5 V/Å and (b) 0.0 V/Å.

```
# Save the figure
plt.savefig('plot-V.pdf')
# Show the figure
plt.show()
```

By running `plot-potential.ipynb`, we obtain the planar-averaged electrostatic potential along the z direction of bilayer graphene in the presence of $E = 0.5$ V/Å, as shown in Fig. 3.30 (a). There are two minimum values of the electrostatic potential at $z =$ 8.2 Å and 11.8 Å, which correspond to the positions of each graphene layer. We can see that the saw-tooth potential decreases from $z = 0$ Å to 1 Å and increases from $z = 1$ Å to 20 Å. In contrast, for $E =$ 0.0 V/Å, the electrostatic potential does not change in the vacuum region, as shown in Fig. 3.30 (b). If the readers want to calculate the electrostatic potential without the external electric field ($E = 0.0$ V/Å), the readers can set `eamp = 0.0` (line 23) in the `scf.in` file.

❑ **Plotting data band structure:** The band structure is given by `bi-gr.bands.gnu`, which is the output file of `bands.x`. In a similar way to plot band structure of single-layer graphene in Sec. 3.2.2, the readers can run the file `plot-bands.ipynb` to plot the band structure of the bilayer graphene. In Fig. 3.31 (a) and (b), we show the band structures of the bilayer graphene for $E = 0.5$ V/Å and 0.0 V/Å, respectively. If there is no external electric field (i.e., $E = 0.0$ V/Å), the bilayer graphene has zero-gap, as shown in Fig. 3.31 (b),

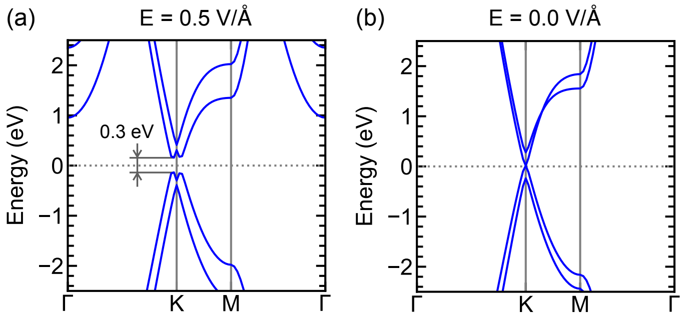


Figure 3.31 Electronic band structures of bilayer graphene with an external electric field E of (a) 0.5 V/Å and (b) 0.0 V/Å. In the presence of E, a band gap is opened at the K point for the bilayer graphene.

which is similar to the case of single-layer graphene. However, if the electric field is applied perpendicular to bilayer graphene, the two layers are no longer identical. Therefore, the two $2p_z$ orbitals around the Fermi energy change the energies since the electric field affects the p_z orbital, but not for the p_x and p_y orbitals, which are parallel to the graphene plane. The band gap thus increases by increasing the external electric field. In particular, the band gap is opened about 0.3 eV for $E = 0.5$ V/Å, as shown in Fig. 3.31 (a). It is noted that the vacuum size will affect the electric field screening, which modifies the energy band gap. Therefore, the readers should consider a sufficiently large vacuum size by changing the value of c in the input files.

Try It Yourself

Plot the planar-average of electrostatic potential with the size of the unit cell $c = 30$, 40, and 50 Å. The positions of the C atoms in the namelist `ATOMIC_POSITIONS` in `scf.in` must rescale for each c value since these postions are in crystal coordinates.

3.6 Maximally-localized Wannier functions

Maximally-localized Wannier functions (MLWFs) are helpful to compute the properties of the materials that require very dense grid of $\boldsymbol{k}$-points and to build a tight-binding model (see Sec. 5.14). In Quantum ESPRESSO, the MLWFs are employed by Wannier90 code [Mostofi *et al.* (2008), Pizzi *et al.* (2020)] (see Chapter 2 for the Wannier90 installation). In this section, we learn how to calculate the MLWFs from Wannier90 and Quantum ESPRESSO (Sec. 3.6.1) and apply the Wannier interpolation for hybrid functional (Sec. 3.6.2).

3.6.1 Wannier functions, energy dispersion, and tight-binding parameters

❑ **Purpose:** In this tutorial, we visually check the Wannier functions (WFs), and we show how to plot energy dispersion and obtain tight-binding parameters based on the MLWFs of graphene.
❑ **Background:** The WFs can be obtained from the Bloch functions by using Eq. (5.59). As discussed in Sec. 5.14, there is a freedom of choice in the Bloch function that corresponds to the shape and the spatial distribution of the WF. We can select any phases of the Bloch function, which modifies the spatial distribution of the WF.

In order to obtain the MLWFs, we minimize the spread of the WF, Ω in Eq. (5.66), by using a steepest-descent algorithm [Marzari *et al.* (2012a)]. After a ground state $\psi_{n\boldsymbol{k}}$ is obtained from the SCF calculation, the Wannier90 code will calculate the MLWFs, which require the following two matrices:

- The overlap matrix $M_{mn}^{\boldsymbol{k},\boldsymbol{b}} = \langle u_{m\boldsymbol{k}} | u_{n,\boldsymbol{k}+\boldsymbol{b}} \rangle$ between the periodic part of the Bloch functions $|u_{n,\boldsymbol{k}+\boldsymbol{b}}\rangle$ (see Eq. (5.72)).
- A starting guess the projection matrix $A_{mn}^{\boldsymbol{k}} = \langle \psi_{m\boldsymbol{k}} | g_n \rangle$ of the Bloch function $\psi_{n\boldsymbol{k}}$ onto trial localized orbitals $|g_n\rangle$ (see Eq. (5.78)).

The calculation of the MLWF of graphene consists of the following five steps:

1. First, we perform the SCF to take the Bloch functions from a ground-state calculation by using `pw.x`. In the case that we need the Bloch functions in a more dense $\boldsymbol{k}$-mesh compared with the previous SCF step, we can run non-SCF after the SCF is finished.

2. It is important to note that Wannier90 requires a full list of $\boldsymbol{k}$-points in the Brillouin zone, while `pw.x` in step (1) uses a reduced $\boldsymbol{k}$-points. Therefore, an executable `open_grid.x` in Quantum ESPRESSO is used to obtain the full grid of $\boldsymbol{k}$-points from the reduced one.
3. Next, we run Wannier90 in post-processing mode (`wannier90.x -pp`) to obtain the `.nnkp` file, which contains the relevant information from the Wannier90 input file in a format to be used in the next step.
4. Then, we run the executable `pw2wannier90.x` (of the Quantum ESPRESSO distribution), which reads the Bloch functions from the SCF (or non-SCF) and the `.nnkp` file to compute $M_{mn}^{\boldsymbol{k},\boldsymbol{b}}$ and $A_{mn}^{\boldsymbol{k}}$ that will be written in the `.mmn` and `.amn` files, respectively.
5. Finally, we run Wannier90 (`wannier90.x`) by reading the files produced by the previous step to minimize the spread to calculate the MLWFs.

❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/gr/w90/
2 $ mpirun -np 4 pw.x < scf.in > scf.out &
3 $ mpirun -np 4 open_grid.x < open_grid.in >
     open_grid.out &
4 $ wannier90.x -pp gr &
5 $ mpirun -np 4 pw2wannier90.x < pw2wan.in > pw2wan.
     out &
6 $ wannier90.x gr &
```

- Line 1: Go to `w90` directory.
- Line 2: Run `pw.x` for the SCF calculation for the step (1).
- Line 3: Run `open_grid.x` to unfold the reduced $\boldsymbol{k}$-points onto the full $\boldsymbol{k}$-points for the step (2).
- Line 4: Run Wannier90 post-processing for the step (3). It requires the second input file `gr.win`.
- Line 5: Run Wannier90 interface `pw2wannier90.x` with the input file `pw2wan.in` for the step (4).
- Line 6: Run `wannier90.x` with the input file `gr.win` for the step (5).

☞ **Note**: `wannier90.x` can only run on a single processor (`mpirun` can not be used), while `pw2wannier90.x` can run in parallel, in which the number of processors `-np 4` must to be the same as `pw.x`.

❑ **How to check:** When the calculations finish, you can find a message `JOB DONE` at the end of `scf.out`, `open_grid.out`, and `pw2wan.out` files, and a message `All done: wannier90 exiting` at the end of the output file `gr.wout`. To obtain the final value of the spread of the WFs (see Sec. 5.14.2), the readers can use `grep` command as follows:

```
$ grep 'Final Spread' gr.wout
```

The total spread is printed in the terminal as

```
Final Spread (Ang^2)       Omega Total  =    3.773324125
```

❑ **Input file:** The input file `scf.in` is similar to `scf.in` in Sec. 3.1.1 for graphene, but we add `nbnd = 16` in the namelist `SYSTEM`. For the input file `open_grid.in`, the readers can open with `vi` editor as follows:

```
$ vi open_grid.in
```

QE-SSP/gr/w90/open_grid.in

```
&INPUTPP
outdir  = '../tmp/'
prefix  = 'gr'
```

☞**Note for `open_grid.in`:** `outdir` and `prefix` for `open_grid.in` must be the same as that for `scf.in` in the step (1).

For the input file `gr.win`, the readers can open as

```
$ vi gr.win
```

QE-SSP/gr/w90/gr.win

```
# CONTROL
num_bands          = 16
num_wann           = 5

dis_win_min        = -22
dis_win_max        =  11
dis_froz_min       = -22
dis_froz_max       = 0.5

num_iter           = 20

# TIGHT-BINDING
write_hr           = .true.

# PLOTTING
wannier_plot       = .true.
bands_plot         = .true.

wannier_plot_supercell = 3

begin kpoint_path
G 0.0000 0.0000 0.0000  K 0.3333 0.3333 0.0000
K 0.3333 0.3333 0.0000  M 0.5000 0.0000 0.0000
M 0.5000 0.0000 0.0000  G 0.0000 0.0000 0.0000
end kpoint_path

# SYSTEM
begin unit_cell_cart
ang
  2.463906   0.000000   0.000000
 -1.231953   2.133805   0.000000
  0.000000   0.000000  15.000000
end unit_cell_cart

begin atoms_frac
C  0.333333333  0.666666666  0.500000000
C  0.666666666  0.333333333  0.500000000
end atoms_frac

# PROJECTIONS
guiding_centres  = .true.

begin projections
C: pz
f= 0.5, 0.5, 0.5: s
f= 0.5, 0.0, 0.5: s
f= 0.0, 0.5, 0.5: s
```

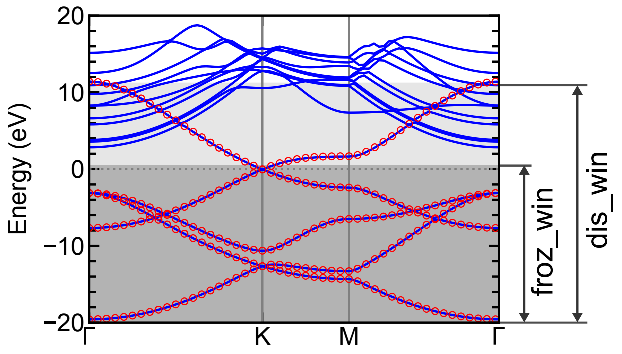


Figure 3.32 An optimized selection for the frozen `froz_win` and the disentanglement `dis_win` for the band structure of graphene. Solid lines are the energy bands that are calculated by the DFT. Solid lines with circles are "fitted energy bands" to the WFs, while only solid lines denote "not-fitted energy bands". `froz_win` should be taken as large as possible with the condition that we should not include "not-fitted bands" (only solid lines). `dis_win` should be taken as small as possible with the condition that we should include all "fitted bands" (lines with circles).

```
48 end projections
49
50 # KPOINTS
51 mp_grid   = 12 12 1
52
53 begin kpoints
54   0.00000000   0.00000000   0.00000000   0.0069444
55   0.00000000   0.08333333   0.00000000   0.0069444
56   ...
57 -0.083333333 -0.08333333   0.00000000   0.0069444
58 end kpoints
```

☞**Explanation of gr.win:** The input variables in `gr.win` are listed in Table 3.10. The `gr.win` file can be divided into 6 parts starting with a hash mark "#". It is noted that the comments in Wannier90 can begin with "#", "%", or "!". Each part is explained as follows:

1. **CONTROL**: This part contains the parameters for minimizing the spread of the WF. These parameters are essential to obtain well-localized WFs. Here, we show how to choose these parameters. First, the number of energy bands `num_bands` (line 2) should be equal to `nbnd = 16` in `scf.in`. Second, the number of the

Table 3.10 Meaning of input variables in `gr.win` file.

Line	Syntax	Meaning
2	`num_bands`	Number of energy bands, which is equal to the number of the Kohn-Sham states `nbnd = 16` in `scf.in`.
3	`num_wann`	Number of the WFs, which is equal to the number of projections.
5	`dis_win_min`	Minimum energy (in eV) of the disentanglement window.
6	`dis_win_max`	Maximum energy (in eV) of the disentanglement window.
7	`dis_froz_min`	Minimum energy (in eV) of the frozen window.
8	`dis_win_max`	Maximum energy (in eV) of the frozen window.
10	`num_iter`	Maximum number of iterations for the minimization of spread.
13	`write_hr`	If `write_hr = .true.`, the code will write the tight-binding Hamiltonian in output file `*_hr.dat`. The default is `.false.`
16	`wannier_plot`	If `wannier_plot = .true.`, the code will write out the Wannier functions in in output files `*.xsf`, which are opened by XCrySDen or VESTA codes. The default is `false`.
17	`bands_plot`	If `bands_plot = .true.`, the code will calculate the band structure, through Wannier interpolation, and write in output file `*_band.dat`. The default is `false`.
19	`wannier_plot_supercell`	Size of the supercell for plotting the WF. The default value is 2.
21–25	`kpoint_path`	Defines the path in $\boldsymbol{k}$-space along which to calculate the bandstructure.
28–32	`unit_cell_cart`	Lattice vectors in the Cartesian coordinates.

Continued

Table 3.10 – *Continued*

Line	Syntax	Meaning
33	`ang`	The unit of the lattice vectors (Angstrom).
35–38	`atoms_frac`	Atomic positions in fractional coordinates.
41	`guiding_centres`	If `guiding_centres = .true.`, the projection centres are used as the guiding centres during the minimization to avoid local minima. The default is `false`.
43–48	`projections`	Defines a set of localized functions used to generate the projections matrix $A^{\boldsymbol{k}}_{mn}$.
51	`mp_grid`	Dimensions of the $\boldsymbol{k}$-points grid.
53–58	`kpoints`	The positions of each $\boldsymbol{k}$ point.

WFs `num_wann` (line 3) should be equal to the number of the projectors, which will be explained in part (5). We can see that `num_wann` is not equal to `num_bands`. In order to solve this problem, Souza *et al.* [Souza *et al.* (2001)] introduced the disentanglement method in Wannier90. This method requires the disentanglement `dis_win` and frozen `froz_win` energy windows. The optimized way to choose `dis_win` and `froz_win` is shown in Fig. 3.32. `dis_win` is a window to extract the target bands for the WFs. Therefore, `dis_win` should cover the energy range of all WFs, and `dis_win` should be as small as possible to reduce the computational cost. For graphene, the WFs include four valence bands and one conduction band since we have five WFs. These bands have the energy range from -20 to 12 eV (see Fig. 3.14). Therefore, after shifting with the Fermi energy (-1.6786 eV, see in `scf.out`), we can choose `dis_win_min = -22` and `dis_win_max = 11` in units of eV (lines 5 and 6), respectively. It is noted that these values do not need to be very precise. However, if `dis_win` is too small, the readers will get an error in `gr.wout` as follows:

```
Exiting.......
dis_windows: Energy window contains fewer states
     than number of target WFs
```

froz_win is a window used for the disentanglement procedure. This window should not contain “not-fitted energy bands”, and it should be as large as possible for optimizing the disentanglement procedure. For graphene, the lowest energy of 6-th band at 2.5 eV, as shown in Fig. 3.14. Therefore, after shifting with the Fermi energy, we can choose `froz_win_min = -22` and `froz_win_max = 0.5` in units of eV (lines 5 and 6), respectively. We recommend to run the energy band structure in Sec. 3.2.2 to estimate the values for the `froz_win` and `dis_win`.

Finally, we choose the maximum number of iterations `num_iter = 20` for the minimization of spread. Large number of `num_iter` would mix the WFs such that the output WFs are not atomic like. Therefore, if you want atomic like WFs, it is better to set `num_iter` less than 20.

2. **TIGHT-BINDING**: By setting `write_hr = .true.`, the tight-binding Hamiltonian in the WF basis (see Sec. 5.14.3) will be written in output file `gr_hr.dat`.
3. **PLOTTING**: This part contains the parameters for plotting. It is noted that the WF is not periodic in the unit cell. Therefore, it is not generally sufficient to plot it in a single unit cell. `wannier_plot_supercell` controls the number of units cell in which we construct the WF. For graphene, we select `wannier_plot_supercell = 3` (line 19), which is sufficient to give a good picture of a WF. In order to plot the band structure with `bands_plot = .true.` (line 17), the path in ***k***-space need to list in the block of `kpoint_path` from line 22 to line 24. Each line gives the start and end points (with labels) for a section of the path. Values are in fractional coordinates with respect to the primitive reciprocal lattice vectors. The number of ***k*** points along each path is given in output file `gr_band.labelinfo.dat`.
4. **SYSTEM**: This part includes two blocks `unit_cell_cart` (lines 28–33) and `atoms_frac` (lines 35–38), in which `unit_cell_cart` are the lattice vectors (in Cartesian coordinates) and `atoms_frac` are the atomic positions (in

fractional coordinates). These values can be obtained from `scf.in`. The unit of lattice vectors is defined by `ang` (line 29) for Angstrom or `bohr` for Bohr.

5. **PROJECTIONS**: This part includes the parameters for selecting the initial projections, which need to generate the matrix $A_{mn}^{\boldsymbol{k}}$. For graphene, we project the Bloch functions onto the s and p_z orbitals of the C atoms. The positions of these orbitals are set in the block `projections` from line 44 to line 47. `C: pz` (line 44) means that the p_z orbitals are located at the positions of the C atoms, while s orbitals are located at the center of the C-C bonds. Since we have two C atoms and three C-C bonds in the unit cell, we have the 5 WFs. Therefore, we can set `num_wann = 5` (line 3). It is noted that the option `guiding_centres = .true.` (line 41) is used during the minimization to avoid local minima. For this reason, we recommend to use `guiding_centres = .true.` where the block `projections` is defined.
6. **KPOINTS**: The grid of $\boldsymbol{k}$-points is `mp_grid = 12 12 1` (line 51), which is same as $\boldsymbol{k}$-points in `scf.in`. The full list of $\boldsymbol{k}$-points in the block `kpoints` is copied and pasted from the output file `open_grid.out` in the step (2).

For the input file `pw2wan.in`, the readers can open by

```
$ vi pw2wan.in
```

QE-SSP/gr/w90/pw2wan.in

```
1 &INPUTPP
2 outdir    = '../tmp/'
3 prefix    = 'gr_open'
4 seedname  = 'gr'
5 write_unk = .true.
6 /
```

☞ **Explanation of pw2wan.in:** The executable `open_grid.x` in the step (2) generates a new folder `gr_open.save` (in the `tmp` directory), to store the wavefunctions of the full $\boldsymbol{k}$-points, while `gr.save` stores the wavefunctions of the reduced $\boldsymbol{k}$-points. Therefore, we must to setting `prefix = 'gr_open'` (line 3). In order to plot the Wannier functions in real space, we need to set `write_unk = .true.` (line

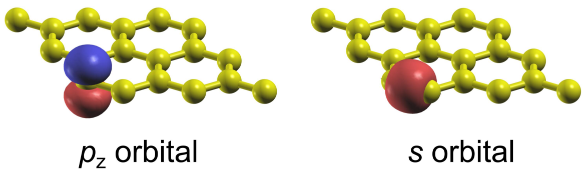
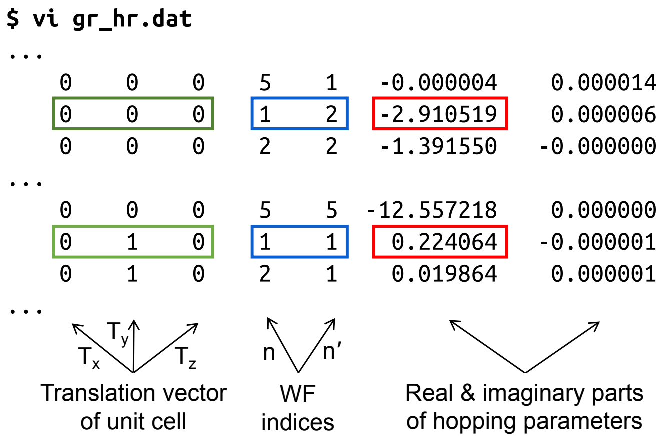


Figure 3.33 The Wannier functions of graphene includes p_z (left) and s (right) orbitals.

5). This option also produces a set of wavefunctions $u_{n\boldsymbol{k}}(\boldsymbol{r})$ on a real-space grid in the output files `UNK00001.1`, `UNK00002.1`, ...
☞**Note:** If the $\boldsymbol{k}$-points in `gr.win` and `open_grid.out` are not the same or `prefix` in `pw2wan.in` is not correct, the readers will get an error in `pw2wan.out` as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     Error in routine pw2wannier90 (144):
     Wrong number of k-points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

❑ **Output file:** The main output files for this tutorial are `gr_*.xsf`, `gr_band.dat`, and `gr_hr.dat`. We will visualize the WFs from `gr_*.xsf`, plot the band structure from `gr_band.dat`, and extract the tight-binding parameters from `gr_hr.dat`.
☞ **Visualizing the Wannier functions:** The `gr_*.xsf` file can open directly by using VESTA. For XCrySDen, we can open as **File → Open Structure → Open XSF (XCrySDen Structure File)** and select `gr_*.xsf` file. Then, we select **Tools → Data Grid**, click on the **OK** button, enter a value in the box Isovalue (2 is a good choice for graphene), click on the **Submit** button. In Fig. 3.33 (a) and (b), we show the p_z and s orbitals for the output files `gr_00001.xsf` and `gr_00003.xsf`, respectively.
☞ **Plotting energy dispersion:** By running step (5), the band structure based on the Wannier interpolation (see Sec. 5.14.3) is written in the output file `gr_band.dat`. In a similar way to plot band structure of graphene in Sec. 3.2.2, the readers can run the file `plot-bands.ipynb` to plot the band structure from output file `gr_band.dat`. The band structure is plotted in Fig. 3.32, in which



```
$ vi gr_hr.dat
...
    0    0    0    5    1   -0.000004    0.000014
    0    0    0    1    2   -2.910519    0.000006
    0    0    0    2    2   -1.391550   -0.000000
...
    0    0    0    5    5  -12.557218    0.000000
    0    1    0    1    1    0.224064   -0.000001
    0    1    0    2    1    0.019864    0.000001
...
```

T_x
T_y
T_z
Translation vector of unit cell
n
n′
WF indices
Real & imaginary parts of hopping parameters

Figure 3.34 The meaning of output file `gr_hr.dat`.

the open dots and solid lines denote the band structure obtained from `wannier90.x` and `pw.x`, respectively. For five lowest energy-bands, which correspond to three s- and two p_z-orbitals, the Wannier interpolation can reproduce the band structure from SCF calculation. The advantage of the Wannier interpolation is that the band structure can be accurately calculated for a dense $\boldsymbol{k}$-mesh based on the SCF calculation for a coarse $\boldsymbol{k}$-mesh. This is helpful for the case that the SCF calculation is time-consuming for a dense $\boldsymbol{k}$-mesh, such as the hybrid functional calculation, which will be explained in Sec. 3.6.2.

☞ **Extracting tight-binding parameters:** The tight-binding parameters of graphene are given by the output file `gr_hr.dat`, as shown in Fig. 3.34. The readers can use the `vi` editor to open this file. Each line in `gr_hr.dat` includes 7 values as follows: the first three values give the translation vector $\boldsymbol{T}$ of the unit cell in x, y, and z directions, respectively, the next two values give the WF indices n and n', and the least two values are the real and imaginary parts of the hopping parameters in units of eV $t_{nn'} = \langle w_{n,0}|\mathcal{H}|w_{n',\boldsymbol{T}}\rangle$ (see Eq. 5.80). For graphene, we consider the hopping parameters up to the 1st nearest neighbor unit cell, i.e., t_{12} and t_{11} for $T = (0,0,0)$ and $(0,1,0)$, respectively. Then, we obtain $t_{12} = -2.91$ eV and $t_{11} = 0.224$ eV, as shown in Fig. 3.34.

Try It Yourself

1. Recompute band structure of graphene by changing the values of `dis_win_max` and `dis_froz_max`.
2. Calculate the MLWFs and band structure of monolayer MoS_2.

3.6.2 Wannier interpolation for hybrid functional

❑ **Purpose:** In this tutorial, we show how to obtain the band structure of the monolayer MoS_2 with the HSE (Heyd-Scuseria-Ernzerhof) hybrid functional and Wannier interpolation.
❑ **Background:** As discussed in Sec. 4.11.3, the LDA and GGA approximations often underestimate the value of the energy band gap. In order to obtain a reasonable energy gap, the HSE functional can be used in the SCF calculation in Quantum ESPRESSO. However, Quantum ESPRESSO only supports the HSE functional for a uniform ***k***-mesh. In addition, the HSE functional requires a non-local term of the Hartree-Fock exchange interaction, which is time-consuming for a dense ***k***-mesh. For this reason, the Wannier interpolation is necessary to plot the band structure along with the high-symmetry points using a non-uniform and dense ***k***-mesh for HSE functional.
❑ **How to run:** To run this tutorial, the readers should type the following command lines:

```
1 $ cd ~/QE-SSP/mos2/w90/
2 $ mpirun -np 8 pw.x < scf.in > scf.out &
3 $ mpirun -np 8 open_grid.x < open_grid.in >
     open_grid.out &
4 $ wannier90.x -pp mos2 &
5 $ mpirun -np 8 pw2wannier90.x < pw2wan.in > pw2wan.
     out &
6 $ wannier90.x mos2 &
```

The meaning of each step is given by the tutorial in Sec. 3.6.1. Since the SCF calculation with the HSE functional (line 2) is time-consuming, we run 8 processes in parallel (`mpirun -np 8`).

❑ **How to check:** When the calculations finish, the messages `JOB DONE` and `All done: wannier90 exiting` are written at the end of the output files, as shown in Sec. 3.6.1.

❑ **Input file:** For the input file `scf.in`, the readers can open as follows:

```
$ vi scf.in
```

QE-SSP/mos2/w90/scf.in

```
&CONTROL
calculation      = 'scf'
pseudo_dir       = '../pseudo/'
outdir           = '../tmp/'
prefix           = 'mos2'
/
&SYSTEM
ibrav            = 4
a                = 3.1825188839
c                = 20.0
nat              = 3
ntyp             = 2
nbnd             = 30
ecutwfc          = 60.0
input_dft        = 'hse'
exx_fraction     = 0.25
nqx1             = 2
nqx2             = 2
nqx3             = 1
/
&ELECTRONS
mixing_beta      = 0.7
conv_thr         = 1.0d-6
/
ATOMIC_SPECIES
Mo  95.94   Mo.pbe-spn-rrkjus_psl.1.0.0.UPF
S   32.065  S.pbe-nl-rrkjus_psl.1.0.0.UPF
ATOMIC_POSITIONS (crystal)
Mo   0.0000000000   0.0000000000   0.5000000000
S    0.3333333333   0.6666666667   0.4217548051
S    0.3333333333   0.6666666667   0.5782450789
K_POINTS (automatic)
6 6 1 0 0 0
```

☞**Explanation of scf.in:** The HSE functional is calculated by setting `input_dft = 'hse'` (line 15) in the namelist `SYSTEM`. `exx_fraction` (line 16) is the mixing coefficient, which is 0.25 for the HSE functional, as shown in Table 4.3. `nqx1`, `nqx2`, and `nqx3` (lines 17, 18, and 19), respectively, are dimensions of the ***q***-grid for the Hartree-Fock exchange interaction. The readers should check the convergence of the energy band by changing the ***q***-grid. The calculation of the ***q***-grid convergence will be very time-consuming compared to the ***k***-grid convergence in Sec. 3.1.3. Therefore, we recommend increasing the number of processes in parallel, which might require a workstation. Here, we select a coarse $2 \times 2 \times 1$ grid for ***q***-points, which is appropriate to run with a PC with 8 processes, such as Intel Core i7 or i9.

For the input file `open_grid.in`, the readers can open as follows:

```
$ vi open_grid.in
```

QE-SSP/mos2/w90/open_grid.in

```
&INPUTPP
outdir          = '../tmp/'
prefix          = 'mos2'
/
```

For the input file `mos2.win`, the readers can open as follows:

```
$ vi mos2.win
```

QE-SSP/mos2/w90/mos2.win

```
# CONTROL
num_bands          = 30
num_wann           = 11

num_iter           = 20

dis_win_min        = -8
dis_win_max        = 5
dis_froz_min       = -8
dis_froz_max       = 3.5
```

```
11
12 # PLOTTING
13 bands_plot       = .true.
14
15 begin kpoint_path
16 G 0.0000 0.0000 0.0000  K 0.3333 0.3333 0.0000
17 K 0.3333 0.3333 0.0000  M 0.5000 0.0000 0.0000
18 M 0.5000 0.0000 0.0000  G 0.0000 0.0000 0.0000
19 end kpoint_path
20
21 #SYSTEM
22 begin unit_cell_cart
23 ang
24   3.182518   0.000000   0.000000
25  -1.591259   2.756142   0.000000
26   0.000000   0.000000  20.000000
27 end unit_cell_cart
28
29 begin atoms_frac
30 Mo   0.0000000000   0.0000000000   0.5000000000
31 S    0.3333333333   0.6666666667   0.4217548051
32 S    0.3333333333   0.6666666667   0.5782450789
33 end atoms_frac
34
35 # PROJECTIONS
36 guiding_centres  = .true.
37
38 begin projections
39 Mo: dxy; dyz; dxz; dx2-y2; dz2
40 S: px; py; pz
41 end projections
42
43 # KPOINTS
44 mp_grid  = 6 6 1
45
46 begin kpoints
47   0.00000000  0.00000000  0.00000000  0.0277777
48   0.00000000  0.16666666  0.00000000  0.0277777
49   ...
50  -0.16666666 -0.16666666  0.00000000  0.0277777
51 end kpoints
```

☞**Explanation of mos2.win:** The detail of parameters in the `mos2.win` file is given in Table 3.10. Here, five d orbitals ($d_{xy}, d_{yz}, d_{xz}, d_{x^2-y^2}$, and d_{z^2}) and three p orbitals (p_x, p_y, and p_z) are selected as the initial projectors for the Mo and S atoms, respectively. Thus, we have 11 WFs `num_wann = 11` (line 3).

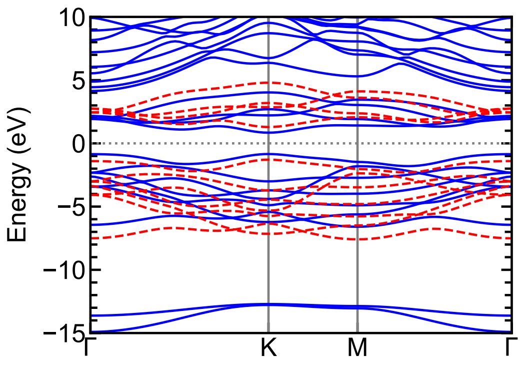


Figure 3.35 Energy band structure of monolayer MoS_2. Solid and dashed lines denote the band structure with the GGA and HSE functionals, respectively.

For the input file `pw2wan.in`, the readers can open as follows:

```
$ vi pw2wan.in
```

QE-SSP/mos2/w90/pw2wan.in

```
&INPUTPP
outdir     = '../tmp/'
prefix     = 'mos2_open'
seedname   = 'mos2'
/
```

❏ **Output file:** The band structure is written in the output file `mos2_band.dat`. The readers can run the file `plot-bands.ipynb` to plot the band structures from `mos2_band.dat` (with HSE functional) and `mos2_bands.gnu` (with only GGA functional). It is noted that the file `mos2_bands.gnu` is obtained by `bands.x` for a non-SCF calculation (see Sec. 3.2.2). In Fig. 3.35, we show the band structure of the monolayer MoS_2 with the GGA (solid lines) and HSE (dashed lines) functionals. The energy gaps are 1.688 and 2.578 eV for the GGA and HSE functionals, respectively. Compared with the experimental value (2.40 $\pm$ 0.05 eV [Huang *et al.* (2015)]), the HSE functional gives a better value of the energy gap.

Try It Yourself

1. Calculate the energy gap of the bulk Si with the LDA, GGA, and HSE functionals, and compare to the experimental value (1.12 eV).
2. Plot the Wannier function of the bulk Si with the HSE functional.