ToolBoX Seminar Lecture Notes

A Practical Introduction to Density Functional Theory

L. Rademaker[1*]

1 Department of Theoretical Physics, University of Geneva, 1211 Geneva, Switzerland
* louk.rademaker@gmail.com

March 19, 2020

Abstract

These notes contain a brief practical introduction to doing density functional theory calculations for crystals using the open source Quantum Espresso software. The level is aimed at graduate students who are studying condensed matter or solid state physics, either theoretical or experimental.

Contents

1 Introduction: What is Density Functional Theory?

Any material on earth, whether in crystals, amorphous solids, molecules or yourself, consists of nothing else than a bunch of atoms, ions and electrons bound together by electric forces. All these possible forms of matter can be explained by virtue of one simple equation: the many-particle Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}\Phi(\mathbf{r};t) = \left(-\sum_i^N \frac{\hbar^2}{2m_i}\frac{\partial^2}{\partial \mathbf{r}_i^2} + \sum_{i<j}^N \frac{e^2 Z_i Z_j}{|\mathbf{r}_i - \mathbf{r}_j|}\right)\Phi(\mathbf{r};t). \quad (1)$$

Here $\Phi(\mathbf{r};t)$ is the many-body wavefunction for N particles, where each particle has its own mass m_i, charge Z_i and position $\mathbf{r}_i$. The only interaction is the Coulomb interaction e^2/r.

Despite its apparent simplicity, Eq. (1) is notoriously difficult to solve. This is where density functional theory (DFT) comes in. Using a set of reasonable physical approximations we can simplify the many-particle Schrödinger equation to something that we can actually solve numerically.

1.1 Born-Oppenheimer approximation

The first approximation arises from the physical problem we want to study: the ground state of a collection of interacting ions and electrons. Because even the lightest ion is more than a thousand times heavier than an electron, we will forget about the dynamics of the ions all-together. This is known as the *Born-Oppenheimer approximation.* We then write the time-independent Schrödinger equation for a collection of N electrons *subject to the electric potential created by the fixed ions*,

$$\left(\sum_i^N\left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial \mathbf{r}_i^2} + V(\mathbf{r}_i)\right) + \sum_{i<j}^N \frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|}\right)\Psi(\mathbf{r}) = E_0\Psi(\mathbf{r}) \quad (2)$$

where $\mathbf{r}_i$ are the positions of the electrons. The potential $V(\mathbf{r}_i)$ is created by the charged ions,

$$V(\mathbf{r}_i) = -\sum_j \frac{e^2 Z_j}{|\mathbf{r}_i - \mathbf{R}_j|} \quad (3)$$

where $\mathbf{R}$ is the (static) positions of the ions and Z_j their charge. Note that the above Hamiltonian – the left hand side of Eq. (2) – contains three terms: the kinetic energy (T), the potential energy (V) and the interaction energy (U).

The electronic density is obtained by integrating out all electron degrees of freedom exact one,

$$n(\mathbf{r}) = \int d^3\mathbf{r}_2 \cdots d^3\mathbf{r}_N \left|\Psi(\mathbf{r}_1 \cdots \mathbf{r}_n)\right|^2. \quad (4)$$

The total potential energy V is just given by the integral over the potential $V(\mathbf{r})$ times the density,

$$V = \int d^3\mathbf{r} V(\mathbf{r}) n(\mathbf{r}). \quad (5)$$

1.2 Hohenberg-Kohn theory

Assume we found a solution of Eq. (2), with ground state energy E_0 and a certain electronic density $n(\mathbf{r})$. The strength of the Coulomb interaction and the mass of an electron are constants of nature, so the only input that can possibly influence the electronic density $n(\mathbf{r})$ and the energy E_0 of our ground state is our choice of potential $V(\mathbf{r})$. In other words, the ground state energy is a *functional* of the input potential,

$$E_0[V(\mathbf{r})] = \mathcal{F}_E[V(\mathbf{r})] \tag{6}$$

A functional is nothing else than a function whose input is another function; in this case the functional $\mathcal{F}$ takes as input the electric potential generated by the ions and outputs the ground state energy based on Eq. (2).

At first this results seems counterintuitive. After all, the ground state energy clearly contains the kinetic energy T, the interaction energy U and the potential energy. Only the latter term *explicitly* depends on the potential. We can thus write the ground state energy in terms of a separate functional for the kinetic and interaction energy, and the potential energy

$$E[n(\mathbf{r})] = \mathcal{F}'_E[V(\mathbf{r})] + \int d^3\mathbf{r} V(\mathbf{r}) n(\mathbf{r}) \tag{7}$$

Hohenberg and Kohn [1] came to the elegant insight that the *potential* $V(\mathbf{r})$ *and electronic density* $n(\mathbf{r})$ *are conjugate variables.* Other conjugate variables you may know are for example pressure and volume in thermodynamics or momentum and position in classical physics. The fact that the potential $V(\mathbf{r})$ and the density $n(\mathbf{r})$ are conjugate means you can equally well describe any solution of Eq. (2) using the potential *or* the density.

Formally known as a Legendre transform (in the same way you go from the Hamiltonian to the Lagrangian formulation of classical mechanics), we can change the functional of Eq. 7 to depend on the density $n(\mathbf{r})$ rather than the potential $V(\mathbf{r})$. This is the *Hohenberg-Kohn theorem*: there exists a *universal* functional of electronic density, $\mathcal{F}[n(\mathbf{r})]$, such that for the correct density $n(\mathbf{r})$ it provides the ground state energy of Eq. (2),

$$E[n(\mathbf{r})] = \mathcal{F}[n(\mathbf{r})] + \int d^3\mathbf{r} V(\mathbf{r}) n(\mathbf{r}). \tag{8}$$

Knowing this functional, for any given potential $V(\mathbf{r})$ we minimize the right hand side by checking all possible electronic density distributions.

There are only two minor problems. We don't know what this functional looks like. And even if we did, we don't know how to find the right electronic density.

1.3 Approximating the functional

The unknown functional $\mathcal{F}[n(\mathbf{r})]$ should describe the kinetic and interaction energy of a system described by Eq. 2. Even though we cannot find its exact shape, we can look at its shape in some limiting cases that we can solve.

We know that a free homogeneous electron gas with density n has a ground state energy of

$$E_0 = \frac{3\hbar^2 \left(3\pi^2\right)^{2/3}}{10m} n_0^{5/3}. \tag{9}$$

For a slowly varying electronic density, we can approximate the kinetic energy contribution to the full functional $\mathcal{F}[n(\mathbf{r})]$ as the energy of Eq. (9) evaluated at each point separately,

$$\mathcal{T}_0[n(\mathbf{r})] = \frac{3\hbar^2 \left(3\pi^2\right)^{2/3}}{10m} \int d^3\mathbf{r} (n(\mathbf{r}))^{5/3}. \tag{10}$$

Furthermore, we know from perturbation theory that the lowest order energy contribution from Coulomb interactions is given by the Hartree term,

$$\mathcal{U}_{\mathrm{H}}[n(\mathbf{r})] = \frac{e^2}{2} \int d^3\mathbf{r} d^3\mathbf{r}' \, \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}. \tag{11}$$

It is natural to write out the full functional as containing the homogeneous electron gas term and the Hartree term. The remaining terms, though still unknown, should be small. This unknown part is conventionally called the *exchange-correlation potential* $E_{xc}[n(\mathbf{r})]$. The full Hohenberg-Kohn functional, including the potential energy, is thus

$$\mathcal{E}_{\mathrm{HK}}[n(\mathbf{r})] = \mathcal{T}_0[n(\mathbf{r})] + \int d^3\mathbf{r} V(r) n(\mathbf{r}) + \mathcal{U}_{\mathrm{H}}[n(\mathbf{r})] + E_{xc}[n(\mathbf{r})]. \tag{12}$$

We will later discuss some general choices of exchange-correlation functionals in Sec. 2.2.

1.4 Kohn-Sham equation

We replaced an intractable problem (solving Eq. (2)) with the task of minimizing an unknown functional $\mathcal{F}[n(\mathbf{r})]$ over infinitely many possible electronic densities $n(\mathbf{r})$. In the previous section we already gave some first suggestions for the functional. But once we found it, how to find the right electronic density $n(\mathbf{r})$?

Because the correct density minimizes the functional, we can find the functional by setting it's derivative to zero,

$$\frac{\delta \mathcal{F}[n(\mathbf{r})]}{\delta n(\mathbf{r})} = 0. \tag{13}$$

Using the functional Eq. (12), we write out[1]

$$\frac{\delta \mathcal{T}[n(\mathbf{r})]}{\delta n(\mathbf{r})} + V(\mathbf{r}) + \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d^3\mathbf{r}' + \frac{\delta E_{xc}[n(\mathbf{r})]}{\delta n(\mathbf{r})} = 0. \tag{14}$$

The idea of Kohn and Sham [2] was to treat this as if it is a *single-particle problem.* The first term represents the kinetic energy, and the remaining terms form the *Kohn-Sham potential*

$$V_{\mathrm{KS}}(\mathbf{r}) = V(\mathbf{r}) + \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d^3\mathbf{r}' + \frac{\delta E_{xc}[n(\mathbf{r})]}{\delta n(\mathbf{r})}. \tag{15}$$

The *Kohn-Sham equation* is the single-particle Schrödinger equation with the potential given by Eq. (15),

$$\left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial \mathbf{r}^2} + V_{\mathrm{KS}}(\mathbf{r})\right)\psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}). \tag{16}$$

We solve these equations numerically, which is tractable because it's just a linear differential equation. The electronic density is obtained by occupying the N solutions $\psi_i(\mathbf{r})$ with the lowest energy,

$$n(\mathbf{r}) = \sum_{i=1}^{N} |\psi_i(\mathbf{r})|^2. \tag{17}$$

Now the electronic density obtained this way can be used to calculate a new Kohn-Sham potential following Eq. (15). We continue this iterative procedure until we reach convergence.

[1]There is a small subtlety included in this equation: the kinetic component of the functional $\mathcal{T}[n(\mathbf{r})]$ should not be the one obtained for the free homogeneous noninteracting electron gas of Eq. (10), but the one for a noninteracting gas subject to the Kohn-Sham potential.

A final comment is in order: in the above derivation we completely ignored the spin of electrons. Of course, real electrons have spin so that you need that degree of freedom as well. This does not change anything fundamental about how to use the Kohn-Sham equation.

1.5 Limitations

We have reached the end-goal: using the Born-Oppenheimer approximation, with an appropriate choice of functional, we use the Kohn-Sham equations to find the ground state energy and electronic density of a system of interacting electrons and ions. This combination of approximations and techniques is called *density functional theory* (DFT).

Despite is sometimes shaky assumptions, DFT turned out to be a resounding success. A large majority of crystalline materials, many molecules and molecular structures have been explained using DFT. Walter Kohn – the man who was involved in both the Hohenberg-Kohn theory and the Kohn-Sham equations – received the Nobel Prize for DFT in 1998. Mainly because of this success, I assume, you want to learn DFT in this ToolBoX.

However, let me briefly shed some clouds over DFT's success.

- A major limitation of DFT is that it is impossible to tell you the size of the errors that exist due to the assumptions. If you get a self-consistent solution, that is nice, but only a comparison with the experimental system will tell you whether it is a good solution.
- The electronic properties of many materials can be described using band theory, meaning for every quasimomentum $\mathbf{k}$ we have a set of energies $\epsilon_n(\mathbf{k})$. The solutions of the Kohn-Sham equation Eq. (16) are commonly interpreted as these electronic bands. However, it is important to bear in mind that in principle there is no connection to the *actual* electronic energy levels in a material. It just turns out that, in many materials, the Kohn-Sham energies happens to be a good approximation.
- As a corollary to the previous limitation: when using DFT for an insulator or semiconductor, you can also compute the Kohn-Sham energy gap between the highest occupied and lowest unoccupied state. This is *not* the actual gap of the semiconductor or insulator. In practice, it turns out that DFT typically underestimates the real gap.
- Note that even if we had the exact functional, solving the corresponding Kohn-Sham equations would not give you the exact solution for the ground state energy.
- Because the Kohn-Sham equations describe non-interacting electrons, many materials with strong correlations cannot be described using DFT. In general, this is true for materials with partially filled d or f-orbitals; or materials with localized electrons. In particular, applying DFT to the class of high-temperature superconductors such as cuprates, pnictides, and heavy fermions is relatively unsuccessful.
- Materials with ground state degeneracy – for example in the case of spontaneous symmetry breaking – are known to be difficult to compute using DFT.
- There is some subtlety involved in choosing the right functional. Commonly, functionals become widely accepted because of a good overlap with experiments.

A good book to learn more about the theoretical side of DFT is Ref. [3].

2 A Practical Guide

DFT as introduced in the last section is a numerical technique – there are no analytical expansions or solutions. And like many modern numerical techniques, it is better to use an existing developed code than to write your own. There are numerous DFT software packages, both open source and payed. In this section we will outline the most important choices we need to make in order to start running some code.

2.1 Wavefunction basis sets

The core of any code consists of computing the solutions to the Kohn-Sham equation. Because this is a linear differential equation, we need to choose a *basis* over which we can expand the Kohn-Sham equation. In this way, we transformed the continuum differential equation into a matrix equation, for which there are many known techniques for diagonalizing it.

The two most popular choices of basis are *plane waves* (PW) and *Gaussian type orbitals* (GTO). If you are interested in the structure of molecules, the most logical basis set is GTO where you take a certain set of polynomials multiplied by a Gaussian envelope, to make sure the electronic density remains close to the ions.

For crystals, on the other hand, the most logical basis set is plane waves $\psi(\mathbf{k}) = e^{i\mathbf{k}\cdot\mathbf{r}}$ in a box with periodic boundary conditions. Because this course is aimed at condensed matter physicists, we will use a plane-wave code.

A full list of existing DFT codes, with their preferred basis set, is maintained on Wikipedia [4].

2.2 Which functional?

In Sec. 1.3 we introduced some basic ideas regarding the precise shape of the functional. We separated the kinetic energy of a noninteracting gas and the Hartree-Fock interaction energy from the exchange-correlation functional. Here we will discuss in more detail some possible exchange-correlation functionals that have been proposed, without any attempt at being encyclopedic.

It is known – see for example the textbook [5], chapter 5 – that the energy of the homogeneous electron gas can be expanded in powers of the Fermi momentum $k_F \sim n^{1/3}$. Kohn and Sham [2] suggested to use these analytical results for the exchange-correlation functional, which is now known as the *local density approximation* (LDA).

A natural next step is to have a functional that not only depends on the density $n(\mathbf{r})$ but also on its derivative $\nabla n(\mathbf{r})$. Such functionals are known as generalized gradient approximations (GGA). [6] A popular version of a GGA functional is the *Perdew–Burke-Ernzerhof functional* (PBE) [7] - its publication is cited more than 100000 times! In this work we will use the PBE functional, since it reproduces experimental band-structures relatively accurate.

2.3 Pseudopotentials

Are we ready to start computing? Well, not yet. If we are interested in doing DFT for a crystal, we would give our code the size of the unit cell, the type of atoms and their positions. For example, silicon has an *fcc* crystal structure with two atoms per unit cell. Silicon itself is element number 14, which means there are 28 electrons per unit cell. However, both physically and numerically it is nonsensical to include all 28 electrons in our calculation.

Physically speaking, the electronic configuration of silicon is Ne 3s2 3p2. The core electrons, given by the electronic configuration of neon, are completely irrelevant in the physics of binding a silicon crystal. Numerically, the fact that a bare atomic core has a diverging potential $1/r$ creates a lot problems. Both these problems can be solved by putting a *pseudopotential* at the position of the silicon atoms. A pseudopotential is a smeared-out potential that includes the charge of the physically irrelevant core electrons, that is therefore more numerically stable than the diverging $1/r$ potential of a bare atomic core. Using a pseudopotential for silicon, for example, means you now do DFT with only the four outermost electrons, known as the *valence* electrons.

Like with the choice of functional, there are many different ways to compute a pseudopotential. In fact, most pseudopotentials are tailored to work with certain functionals, so in these notes we will use pseudopotentials that work well with the PBE functional.

2.4 Quantum ESPRESSO

Now we are ready to select a DFT implementation. In these notes we will use the open-source plane-wave DFT code Quantum ESPRESSO (`http://www.quantum-espresso.org/`). Its development was started in Trieste, Italy, but has by now many contributors from all around the world. If you ever use Quantum ESPRESSO scientifically, make sure you explicitly acknowledge the code and cite their original journal publications [8, 9].

Our first task is to *install* Quantum ESPRESSO on your own computer. The simulations we will do in these lecture notes are light enough that they can be done on a standard laptop.

The full source package of Quantum ESPRESSO can be downloaded from

`https://www.quantum-espresso.org/download`

If you are comfortable doing so, you can install Quantum ESPRESSO using the source package. Otherwise, on the above webpage you will also find links for stable binaries for typical platforms such as Windows. For Mac OS X I recommend using MacPorts (`https://www.macports.org/`), which has a port called `quantum-espresso`. It automatically takes care of dependencies, such as OpenMPI and Fortran libraries.

2.5 Materials Cloud

We will use the pseudopotential libraries collected by *Materials Cloud*, [10, 11] an online tool and repository for doing DFT calculations developed by the EPFL and the ETH. On their website, they have collected various pseudopotentials, benchmarked them, and selected for each element the best choice of pseudopotential.

We will download their collection of pseudopotentials, which are all computed for the PBE functional.

1. Go to the website `https://www.materialscloud.org/`, and navigate to the *Discover* page using the top menu.
2. Click on *Standard solid-state pseudopotentials (SSSP).*
3. You will see a periodic table (see Fig. 1). By clicking on the button that reads *Pseudo*, you will download a `tar.gz` file containing all the pseudopotentials.
4. Once you downloaded the library, look into the folder. You will see files with names like `Si.pbe-n-rrkjus_psl.1.0.0.UPF` and `C.pbe-n-kjpaw_psl.1.0.0.UPF`. These are the pseudopotentials for silicon and carbon, respectively. Directly after that, you can see that these pseudopotentials are computed for the PBE functional.

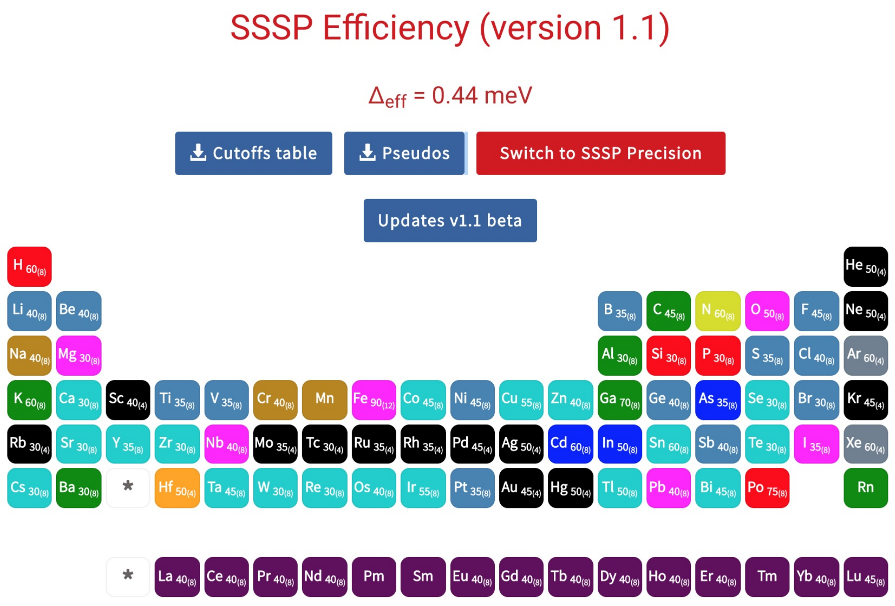

SSSP Efficiency (version 1.1)
Δeff = 0.44 meV
Cutoffs table
Pseudos
Switch to SSSP Precision
Updates v1.1 beta
H 60(8) He 50(4)
Li 40(8) Be 40(8) B 35(8) C 45(8) N 60(8) O 50(8) F 45(8) Ne 50(4)
Na 40(8) Mg 30(8) Al 30(8) Si 30(8) P 30(8) S 35(8) Cl 40(8) Ar 60(4)
K 60(8) Ca 30(8) Sc 40(4) Ti 35(8) V 35(8) Cr 40(8) Mn Fe 90(12) Co 45(8) Ni 45(8) Cu 55(8) Zn 40(8) Ga 70(8) Ge 40(8) As 35(8) Se 30(8) Br 30(8) Kr 45(4)
Rb 30(4) Sr 30(8) Y 35(8) Zr 30(8) Nb 40(8) Mo 35(4) Tc 30(4) Ru 35(4) Rh 35(4) Pd 45(4) Ag 50(4) Cd 60(8) In 50(8) Sn 60(8) Sb 40(8) Te 30(8) I 35(8) Xe 60(4)
Cs 30(8) Ba 30(8) * Hf 50(4) Ta 45(8) W 30(8) Re 30(8) Os 40(8) Ir 55(8) Pt 35(8) Au 45(4) Hg 50(4) Tl 50(8) Pb 40(8) Bi 45(8) Po 75(8) Rn
* La 40(8) Ce 40(8) Pr 40(8) Nd 40(8) Pm Sm Eu 40(8) Gd 40(8) Tb 40(8) Dy 40(8) Ho 40(8) Er 40(8) Tm Yb 40(8) Lu 45(8)

Figure 1: The *Standard solid-state pseudopotentials library* from Materials Cloud [10, 11] contains for all elements a choice of pseudopotential, which can be downloaded on `https://www.materialscloud.org/`.

3 Example 1: Silicon

Silicon is one of the most abundant materials on earth, and one of the most used in modern technology. It is therefore logical that we start with silicon as the first material we are going to study.

To start off, create a folder where we will do all of our calculations, with three subfolders: `silicon`, `out` and `pseudo`. Then move the pseudopotential file for silicon that we downloaded from Materials Cloud into the `pseudo` directory.

3.1 Self-consistent field

3.1.1 Writing the input file

Quantum ESPRESSO works with input files. These are text-only files that contain all the parameters that you want to give to your code. We will now build together an input file for computing the ground state of a silicon crystal. Note that all possible parameters are summarized in the file `INPUT_PW.txt` or `INPUT_PW.html` that came with the source package. If you are unsure on what to write, please ***

So let's get coding! Open your favorite plain text editor, and create a new file called `silicon.scf.in`. The input file we will create is structured with *cards*, of the form `&NAME ... /`.

The first card named `CONTROL` describes calculation parameters.

```
&CONTROL
  calculation = 'scf'
  prefix = 'silicon'
  outdir = '../out/'
  pseudo_dir = '../pseudo/'
  tprnfor = .true.
  verbosity = 'high'
```

```
/
```

- `calculation` decides what calculation we will do. Choose `'scf'`, which is short for self-consistent field calculation. It is the default option of Quantum ESPRESSO. In the scf-mode, the self-consistent loop described in Sec. 1 is done to find the ground state energy and the electronic density.
- `prefix` is the name of your set of calculations. This allows you to later use the output of this calculation in further calculations, for example to obtain the band structure.
- `outdir` is the directory where all the detailed output files will be put, in files of the form `prefix.xml` and `prefix.save`, with the exception of the command line output.
- `pseudo_dir` is the directory where you put the pseudopotentials.
- The last two flags are optional. We set `tprnfor` true, which means the code will calculate and output the forces acting on the ions. This is a useful way to check that your proposed crystal structure is stable. We also set `verbosity` to high, which means the output file will contain a lot of extra information, such as the calculated symmetry representations of the crystal. I can really advise to run both with high and low verbosity and to compare the output files.

The next card describes the system we are studying. What kind of lattice, how many atoms, and what are the cut-offs for our plane wave basis.

```
&SYSTEM
  ibrav = 2
  celldm(1) = 10.2
  nat = 2
  ntyp = 1
  occupations = 'fixed'
  ecutwfc = 30
  ecutrho = 120
/
&ELECTRONS
/
```

- `ibrav` selects the type of Bravais unit cell. In the case of silicon, we choose 2, which is the number for face-centered cubic (fcc). In this crystal structure, the lattice vectors are

$$\mathbf{a}_1 = a/2(-1,0,1); \quad \mathbf{a}_2 = a/2(0,1,1); \quad \mathbf{a}_3 = a/2(-1,1,0), \tag{18}$$

  where a is the lattice constant given in `celldm(1)`, see also Fig. 2, left side. The parameter `celldm(1)` is given in atomic units, meaning Bohr radii. A full list of possible Bravais lattices can be found in the file `INPUT_PW.txt`.
- `nat` is the total number of atoms per unit cell, `ntyp` is the total number of *different* atoms per unit cell. In fcc silicon, there is only one type (silicon) but two atoms per unit cell.
- `occupations` tells the code how to occupy the computed Kohn-Sham states. `fixed` means we just occupy all the states below the Fermi level and empty all the states above. Choose this option for insulators; however, for metals (such as graphene in the next section) we need a different option.

- Recall that QUANTUM ESPRESSO uses a plane-wave basis to describe the wavefunctions. We should tell the system *how many* plane waves should be included in the calculation. This is characterized by a kinetic energy cut-off `ecutwfc`, which implies that we take all the plane waves with momenta such that $\frac{\hbar^2|\mathbf{k}|^2}{2m} \leq$ `ecutwfc`. The parameter is given in Rydberg units (1 Ry = 13.606 eV). On the Materials Cloud website, see Fig. 1, there is a suggested minimum wavefunction cut-off for every pseudopotential, which for silicon is 30 Ry. Making it larger makes the code slower, but should give a more accurate result.

- The code not only stores the plane waves of the electronic states, but also directly the density distribution. Therefore a second cut-off is necessary, `ecutrho`. Because density scales as the square of wavefunctions, and the kinetic energy scales as the square of the momentum, we need to include a density cut-off *at least* four times `ecutwfc`. For our case, exactly four times suffices.

- The card `&ELECTRONS` allows us to tell the system how to solve the Kohn-Sham equation. In our case, we just use the default values. However, we need to include the card!

The third part of the input contains explicit information about the atoms: their pseudopotentials and positions.

```
ATOMIC_SPECIES
  Si     28.086     Si.pbe-n-rrkjus_psl.1.0.0.UPF
ATOMIC_POSITIONS alat
  Si     0.00 0.00 0.00
  Si     0.25 0.25 0.25
```

- Below the line `ATOMIC_SPECIES` you list all the *types* of atoms that exist in your unit cell. In our case, it is just silicon. For each type of atom, you give its name (`Si`), its atomic mass in units of u (`28.086`), and the relevant pseudopotential file name.

- After `ATOMIC_POSITIONS` you list the positions of all the atoms. The flag `alat` means the positions are given in cartesian coordinates in units of the lattice parameter a (`celldm(1)`). Alternatively, one can use `angstrom` (cartesian coordinates in Angstrom) or `crystal` (multiples of the primitive lattice vectors). Here we have a silicon atom at the origin, and a second silicon atom at $a(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.

In the final part of the input we tell the code at which momentum points we will do the calculation.

```
K_POINTS automatic
  6 6 6 1 1 1
```

The simplest option is to choose the flag `automatic`, which generates a Monkhorst-Pack grid [12]. The first three numbers indicate the number of k-points in each of the three directions (`6 6 6`). The last three provide a possible offset in each direction: `0` means no offset and thus the inclusion of high-symmetry points like Γ; `1` means that you place the momentum points exactly in between the points generated by `0`. A finer momentum mesh is generated if you choose `1`, so that is what we will choose typically.

3.1.2 Running the code

Congratulations, you have now written your first DFT input file! To run it, simply use the following command on the command line:

```
pw.x -in silicon.scf.in > silicon.scf.out
```

The program `pw.x` is the main component of QUANTUM ESPRESSO. It takes the input file `silicon.scf.in` and does the self-consistent calculation of the ground state energy and density. On most modern computers, it should not take longer than a few seconds for a system as simple as silicon.

3.1.3 Reading the output

The heavy part of the output is sent to the output directory `../out/`. The human-readable part is saved in the file `silicon.scf.out`. Let's read through it together.

The beginning of the output file lists the properties of this calculation, many of them were given by your input file. Some of them were implicit, yet properly picked up. For example, on line 45 you can see that the program will use the PBE exchange-correlation functional. On line 39 we read that we are going to use 4 Kohn-Sham states, with spin degeneracy this corresponds to 8 electrons per unit cell. After that follows properties of the crystal structure, including its symmetries, and the list of momentum points in our grid.

Just before the actual calculation starts, the program estimates the amount of memory needed for this calculation, in the line `Estimated max dynamical RAM per process`. This might be useful to know for more complicated crystal structures, but for silicon this poses no problems.

Between the lines *Self-consistent Calculation* and *End of self-consistent calculation* the self-consistent DFT loop is iterated until we have reached convergence. After this, the file contains for each momentum point the resulting energies of the Kohn-Sham states. This is directly followed by:

```

     highest occupied level (ev):     6.2506

!    total energy              =     -22.83862400 Ry
```

This is the main output of a DFT calculation: the ground state energy. For clarity, the code also includes the energy of the highest occupied level. This is not exactly the same as the Fermi level, because there might be occupied states with a higher energy at momentum points that were not included in your momentum grid.

3.2 Homework: Find the lattice constant and bulk modulus of silicon

a. The ground state energy by itself is not a measurable quantity. However, one of the ideas of DFT is that we can find the lattice constant of a crystal by calculating the ground state energy *as a function of the input lattice constant*, $E_0(a)$. The predicted actual lattice constant is where this function is minimal. So can you predict with your DFT code the lattice constant of silicon?

Hint: Write a code that automatically generates input files with different values of the lattice constant a. Then extract the total energy for every value of a.

b. In the previous exercise you calculated the function $E_0(a)$. The change of energy under uniform compression is characterized by the *bulk modulus*. What is the value of the bulk modulus you calculated?

3.3 Bands

Strictly speaking, the Kohn-Sham energies do not correspond to anything physical. However, it turns out that they are a pretty good approximation to the electron band energies

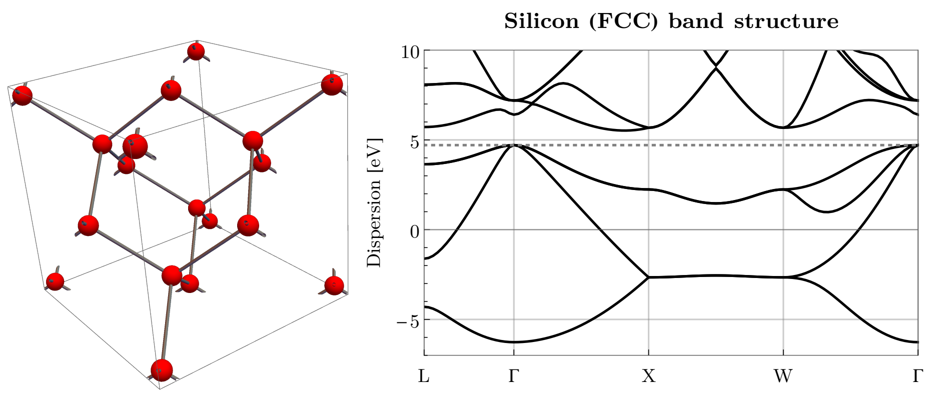


Figure 2: **Left:** Crystal structure of face-centered cubic silicon. **Right:** Band structure of silicon as computed using the steps of Sec. 3.3.

of weakly interacting materials. Therefore it is instructive to calculate the Kohn-Sham energies for many points in the Brillouin zone, to get a sense of silicons *band structure*.

A calculation of the bands can only be done after you finished a self-consistent field calculation with the same parameters. This is because the bands calculation takes the electronic density (and thus the Kohn-Sham potential) obtained in an `scf` calculation, and recomputes the Kohn-Sham energies for a new set of chosen momentum points. A good starting point for the bands calculation input file is therefore to copy the `scf` input file to a new file called `silicon.bands.in`. If you want to write your own bands file from scratch, make sure it has the same `prefix` as the previous self-consistent field calculation.

At three points, we will change this new input file. First, we need to tell the code that we want to calculate the band structure, so replace the `calculation` line with this:

```
 calculation = 'bands'
```

For an insulator or semiconductor like silicon, the default setting is that only occupied bands will be computed. If we are interested in the band gap, we also need to calculate the unoccupied bands. To achieve this, add to the `&SYSTEM` card a line that says we want to calculate 8 bands:

```
 nbnd = 8
```

The final change we need is to the momentum point grid. A customary way to visualize a bandstructure is to choose a *path* in the Brillouin zone and compute the bands along this path. This can be done by the command `tpiba_b`, which means that the `K_POINTS` are in units of $2\pi/a$. The subscript `_b` indicates that we can define a path for a bandstructure calculated. For a path L – Γ – X – W – Γ, we replace the old `K_POINTS` card by

```
K_POINTS tpiba_b
#  tpiba_b = k-points in units of 2pi/a, in format for band calculation
#  number of k-points (use high-symmetry points only)
 5
#  kx, ky, kz, n. of points between this and next one
   0.5 0.5 0.5  20
   0.0 0.0 0.0  30
   0.0 0.0 1.0  30
   0.0 1.0 1.0  30
   0.0 0.0 0.0   0
```

Notice that here we added some comment lines (the one starting with #). These will be ignored by the code, and can be useful for our own understanding of the input files. Here, the comments explain us that we have a path with 5 momentum points, and that we have 20 momentum points in between L and Γ, and so forth.

We can run the bands calculation with this command,

```
pw.x -in silicon.bands.in > silicon.bands.out
```

It might take a few seconds longer than the `scf` calculation, because we have more momentum points.

The output file `silicon.bands.out` starts out with listing the parameters of the calculation. After the line `Band Structure Calculation` it will calculate the Kohn-Sham energies of each desired momentum point. At the end of the calculation, you will find lines looking like this:

```
     End of band structure calculation

          k = 0.5000 0.5000 0.5000 (   754 PWs)   bands (ev):

   -3.3153  -0.6624   5.1803   5.1803   7.9984   9.7300   9.7300  14.1551

          k = 0.4750 0.4750 0.4750 (   748 PWs)   bands (ev):

   -3.3519  -0.6101   5.1849   5.1849   8.0036   9.7355   9.7355  14.1635
```

It signals the end of the calculation, and then it will list for each momentum point all the Kohn-Sham energies in eV. The momentum points themselves are given in units of $\frac{2\pi}{a}$ where a is the lattice constant (`celldm(1)`).

The Quantum ESPRESSO code itself comes with a set of post-processing tools, one of them allows you to plot the band-structure thus calculated. You can also import the output file into your favorite tool (Python, Mathematica, Matlab, GNUplot) and plot it there. The resulting band-structure is shown in Fig. 2, right. Notice it is very similar to the actual band-structure!

The band gap calculated using this code is about 0.8 eV, significantly smaller than the actual band gap in silicon of about 1.1 eV. This is common among DFT calculations of semiconductors. On the other hand, many qualitative features – including the fact that silicon has an indirect band gap – are reproduced with our simple calculation!

3.4 Try at home

Congratulations, you have successfully predicted the properties of a material, purely from first principles! To get comfortable with this technique, try some other three-dimensional semiconductors, and calculate their lattice constant and band-structure.

1. First try other zincblende structures like C-diamond, β-SiC, and GaAs.
2. Next, calculate a different crystal structure: rock-salt NaCl. Can you find in `PW_input.txt` how to program its simple cubic structure?
3. Finally, study hexagonal α-SiC. Which one has a lower ground state energy, α- or β-SiC?

4 Example 2: Graphene

The study of silicon in the previous section allows you to calculate crystal and electronic band structures of any three-dimensional semiconductor. In the remainder of these notes, we will switch gears and introduce a few new concepts: we discuss how to deal with metals, with two-dimensional materials, and how we can optimize the structure within a unit cell, and how to have more complicated unit cells.

Graphene is a perfect material to do this. It is, as you know, a two-dimensional semi-metal with a hexagonal unit cell.

4.1 Compute the band-structure

As with the silicon, we need to write input files for a self-consistent calculation first. Make a new folder `graphene` and start an input file named `graphene.scf.in`. The first card, `&CONTROL`, is the same as in the silicon case but with only the prefix changed to `graphene`.

The `&SYSTEM` card will have some significant changes. Let's write it out in its full totality,

```
&SYSTEM
  assume_isolated = '2D'
  ibrav = 4
  celldm(1) = 4.65
  celldm(3) = 6
  nat = 2
  ntyp = 1
  occupations = 'smearing'
  smearing = 'mv'
  degauss =   1.5000000000d-02
  ecutwfc = 45
  ecutrho = 180
/
&ELECTRONS
/
```

- In the standard DFT implementation, we have periodic boundary conditions in all three direction. The command `assume_isolated = '2D'` ensures that there is no periodicity (neither in the charge density nor in the Coulomb interactions) in the z-direction.

- Graphene has an hexagonal lattice, which has `ibrav = 4`. The lattice vectors are given by
$$\mathbf{a}_1 = a(1,0,0); \quad \mathbf{a}_2 = a(-\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}, 0); \quad \mathbf{a}_3 = a(0,0,c/a), \tag{19}$$
where as before a =`celldm(1)` in Bohr, and c/a =`celldm(3)` is the ratio between the horizontal and vertical lattice size. The value for `celldm(3)` should be such that, for our 2d set-up in graphene, the vertical unit cell size should be large than the cut-off of the pseudopotentials – in this case at least 20 Bohr.

- In the case of a metal or semimetal, just computing the occupied Kohn-Sham energies is very numerically unstable. Tiny changes can lead to different shapes of the Fermi surface. It is therefore necessary to smear the occupations of the Kohn-Sham states, which is ensured by setting `occupations = 'smearing'`. We then also need to set which type of smearing we will use; here we opted for Marzari-Vanderbilt (`mv`) smearing [13], with a width set by `degauss` in Ry units.

- Notice we changed the wavefunction and density cut-offs.

The last part of the input file tells us where the carbon atoms are going to be, and our choice of momentum points. The only subtleties are in the placement of the carbon

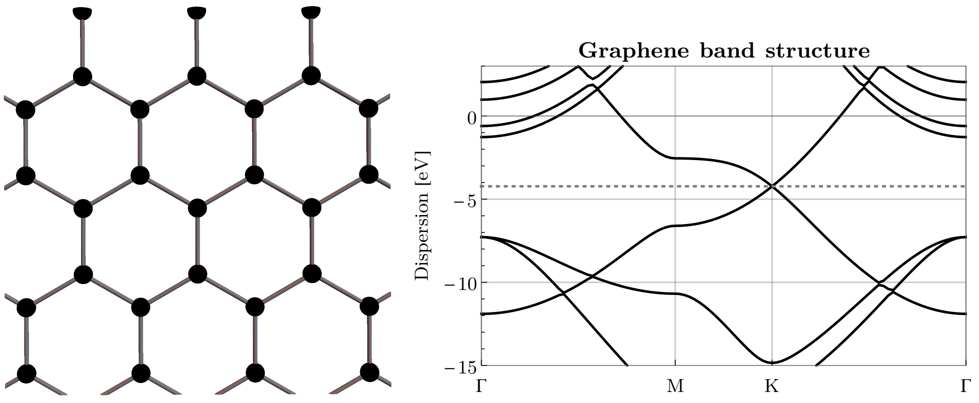


Figure 3: **Left:** The honeycomb crystal structure of graphene. **Right:** Band structure of graphene with the Dirac cone clearly visible.

atoms, see Fig. 3, left, and in the choice of `K_POINTS`: because we have a two-dimensional system we only need one momentum point in the z-direction.

```
ATOMIC_SPECIES
  C      12.0107     C.pbe-n-kjpaw_psl.1.0.0.UPF
ATOMIC_POSITIONS alat
  C      0.000000     0.000000     0.000000
  C      0.000000     0.5773503    0.000000
K_POINTS automatic
  9 9 1 1 1 1
```

Run the self-consistent field calculation by

```
pw.x -in graphene.scf.in > graphene.scf.out
```

The next step is to make our bands calculation input file. Like before, we can just copy the `scf` input file to a new file `graphene.bands.in`. Make sure you change the type of `calculation`, and the list of `K_POINTS`. For the latter, I suggest a path Γ − M − K − Γ. Because K and M are particularly easily expressed in terms of the reciprocal lattice vectors, we write the `K_POINTS` in units `crystal_b`:

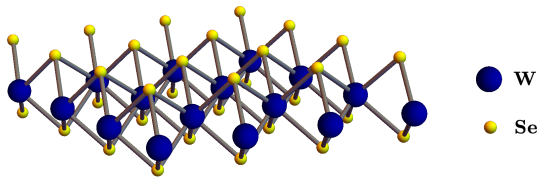
```
K_POINTS crystal_b
 4
   0.000000     0.000000     0.000000      40
   0.500000     0.000000     0.000000      20
   0.333333     0.333333     0.000000      40
   0.000000     0.000000     0.000000      0
```

In the silicon case we explicitly asked the code to compute more than just the occupied bands, using `nbnd`. Because we are computing a (semi)metal, using the `smearing` flag, the code automatically calculates some unoccupied bands as well. We do not need to specify the number of bands `nbnd`.

As before, the bands calculation can now be run by the command

```
pw.x -in graphene.bands.in > graphene.bands.out
```

The resulting band-structure, with the characteristic Dirac cone at the Fermi level, can be seen in Fig. 3.



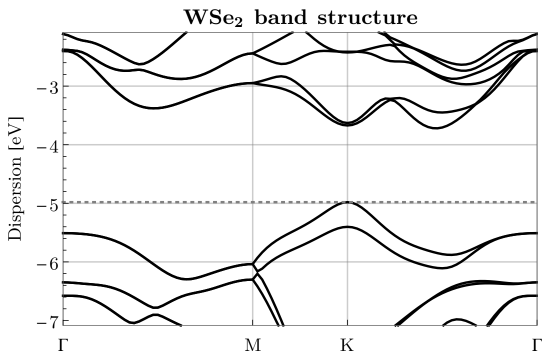


Figure 4: **Left:** Crystal structure of WSe_2. **Right:** Final band-structure with spin-orbit coupling, with a sizeable spin-orbit splitting at K of $\Delta^v_{\mathrm{SOC}} = 0.4$ eV.

5 Example 3: WSe_2

The third and final materials that allows us to learn some new features of DFT is the two-dimensional material WSe_2. In-plane it has a honeycomb lattice structure, with on one sublattice the W atoms, and on the other sublattice two Se atoms, displaced in the positive/negative z-direction, as shown in Fig. 4, left. Using lattice relaxation calculations, we will be able to find the exact displacement of the Se atoms. Furthermore, WSe_2 is a semiconductor with sizeable spin-orbit coupling, and we will show how to include that.

5.1 Relax

In a *relaxation* calculation, the DFT code not only computes the ground state energy but also the derivative of the energy with respect to atomic displacements. This corresponds to the forces acting on each atom. If your initial guess of atomic positions is not stable, there will be nonzero forces. The code will suggest a new set of atomic positions based on the direction of those forces. By repeating this until you have no more forces acting on the atoms, you have relaxed the structure and minimized the ground state energy. We will use this feature to calculate the position of the Se atoms in monolayer WSe_2.

As before, we start by making a new folder `wse2` with in there an input file, which we will call `wse2.relax.in`. We will calculate the position of the Se atoms, using a `relax` calculation. The first card of the input file therefore contains the lines

```
  calculation = 'relax'
  prefix = 'wse2'
```

In the `&CONTROL` card we can also include a force convergence threshold `forc_conv_thr`, which determines how close to zero we want the final forces to be. In our simple calculation we only need to use the default value, so we do not need include it in our input file.

The remainder of the input file looks like this:

```
&SYSTEM
  assume_isolated = '2D'
  ibrav = 0
  nat = 3
  ntyp = 2
  occupations = 'fixed'
  ecutwfc = 30
  ecutrho = 120
/
```

```
&ELECTRONS
/
&IONS
/
ATOMIC_SPECIES
  W      183.840     W_pbe_v1.2.uspp.F.UPF
  Se      78.960     Se_pbe_v1.uspp.F.UPF
ATOMIC_POSITIONS angstrom
  W      0.000000     0.000000       0.000000     0   0   0
  Se     0.000000     1.919689645    1.500000     0   0   1
  Se     0.000000     1.919689645   -1.500000     0   0   1
K_POINTS automatic
  8 8 1 1 1 1
CELL_PARAMETERS angstrom
   3.32500000      0.0000000000      0.0000000000
  -1.66250000      2.8795344676      0.0000000000
   0.00000000      0.0000000000     32.0000000000
```

- In the `&SYSTEM` card, we reverted back to `fixed` occupations since WSe_2 is a semiconductor. Notice how we changed the cut-offs and the number of atoms and types of atoms.

- Because we are interested in the atomic positions, it is worthwhile to write out the lattice vectors and the initial atomic positions explicitly in units of Angstrom. We can do so by selecting `ibrav = 0`, meaning we have a free form of the unit cell. We should then explicitly write out the three unit vectors in a new card called `CELL_PARAMETERS`.

- We are interested in finding the z-position of the Se atoms. We know already that their in-plane coordinates are given by the honeycomb lattice, which are given as the second and third column of the lines after `ATOMIC_POSITIONS angstrom`. In the third column we put the z-position. We set W at $z = 0$, and we guess an initial distance of the Se atoms at $z = \pm 1.5$ A. The last three numbers indicate which atomic position coordinates we will relax. For the Se atoms, `0  0  1` means we keep the x, y coordinates fixed, and will minimize the ground state energy with respect to the z-coordinate.

- We do need to include a new card `&IONS`, where we can specify the properties of the atomic displacements the code will do. Having this card empty just means we choose the default values, but for a `relax` calculation it has to be there!

Run the `pw.x` code as usual. In the output file `wse2.relax.out` you can see the two self-consistent loops. Given a set of atomic positions, the ground state energy and forces are calculated. If the forces are larger than the threshold, a new set of atomic positions is proposed after the line saying `ATOMIC_POSITIONS`. After a few iterations, we have reached convergence and we find the following lines containing the *final coordinates*:

```
Begin final coordinates

ATOMIC_POSITIONS (angstrom)
W         0.000000000    0.000000000    0.000000000    0   0   0
Se        0.000000000    1.919689645    1.678711660    0   0   1
Se        0.000000000    1.919689645   -1.678711660    0   0   1
End final coordinates
```

We predict the distance between the two Se atoms to be 3.36 $\AA$.

5.2 Spin-orbit coupling

In the above lattice relaxation calculation we did not take into account spin-orbit coupling. In general, spin-orbit coupling becomes more important the heavier the element is, which in our case applies to the tungsten (W). Because spin-orbit will likely not influence the position of the Se atoms, we take the atomic positions from the previous calculations and do the standard `scf` followed by `bands` to calculate the bandstructure of WSe_2. Only this time, we will have spin-orbit coupling.

Make the `wse2.scf.in` and `wse2.bands.in` input files. You can combine the elements from the `wse2.relax.in` and `graphene.bands.in`. Make sure you copy the atomic positions from the previous `relax` output file into our new input files. To turn on spin-orbit coupling, we need to add the following two lines to the `&SYSTEM` card:

```
   lspinorb = .true.
   noncolin = .true.
```

The first term turns on spin-orbit coupling, and the second allows for noncollinear spins (so not only up and down but also superpositions). Additionally, we need to have a fully relativistic pseudopotential to study spin-orbit coupling. In the W pseudopotential we have used so far, you can find the following line:

```
The Pseudo was generated with a Scalar-Relativistic Calculation
```

Because `Scalar-Relativistic` implies no spin-orbit coupling, we need to find a new pseudopotential that is fully relativistic! Many different types of pseudopotentials can be downloaded from the Quantum ESPRESSO website. Go to `https://www.quantum-espresso.org/pseudopotentials/ps-library/`, and download a full relativistic ultra-soft pseudopotential (USPP) for tungsten (W) that works with the PBE functional. After that, update the line in the input files where you give the pseudopotential:

```
ATOMIC_SPECIES
  W     183.840    W.rel-pbe-spn-rrkjus_psl.1.0.0.UPF
```

Finally, the amount of valence electron per W is 14 and per Se is 6, meaning the code computes 26 electrons. If you want also to see the conduction bands, I suggest putting `nbnd = 32` or higher.

As before, run the `scf` first, followed by a `bands` calculation. You can check in the output files that the number of Kohn-Sham energies is now equal to the number of electrons. Before we included spin-orbit coupling, the spin degeneracy meant we just needed half the amount of Kohn-Sham energies.

The final band structure is shown in Fig. 4, right. As before, the band gap (here about 1.3 eV) is smaller than experimentally detected (1.7 eV in monolayers). Notably, the valence band at the K point is split due to the spin-orbit coupling, with a splitting of $\Delta^v_{\mathrm{SOC}} = 0.4$ eV, comparable to what is measured in experiments. [14]

6 Further reading

You now have learned the basics of how to compute crystal structures and electronic bands using density functional theory, implemented in the plane-wave tool Quantum ESPRESSO. But there is much more to DFT than just this. There are some additional tools that we haven't discussed, such as the possibility to compute phonon dispersions, Raman or optical spectra, and wannierization. We also haven't looked into modern developments of functionals, such as the inclusion of Van der Waals interactions or strong correlations (LDA+U, GW or DFT+DMFT).

Luckily, there are many online courses available that are more in-depth than this short ToolBox.

- The Materials Cloud webpage also contains a set of lectures, including notes, exercises and videos, on how to do DFT with Quantum ESPRESSO. You can find them here: `https://www.materialscloud.org/learn/`.
- Many universities have their classes on density functional theory online, for example MIT has `https://ocw.mit.edu/courses/materials-science-and-engineering/3-320-atomistic-computer-modeling-of-materials-sma-5107-spring-2005/labs/sections`
- The source manual of Quantum ESPRESSO named `pw_user_guide.pdf`, which comes with downloading the source package, contains a lot of information on what you can do with the code. The source package also contains examples on how to use the code, see the folder `PW/examples` for input files and ideas for `pw.x`. You can also look at examples of other parts of the code, such as `PHonon/examples`, that show you how to compute phonon dispersions.

Acknowledgements

These notes would not exist without the enthusiasm of the ToolBoX organizers João Ferreira and Michael Sonner. I would also like to thank Marco Gibertini for discussions.

Funding information L.R. acknowledges funding from the SNSF in the form of an Ambizione grant.

References

[1] P. Hohenberg and W. Kohn, *Inhomogeneous Electron Gas*, Phys. Rev. **136**, B864 (1964).

[2] W. Kohn and L. J. Sham, *Self-Consistent Equations Including Exchange and Correlation Effects*, Phys. Rev. **140**(4A), A 1133 (1965).

[3] G. Giuliani, G. Vignale and Cambridge University Press, *Quantum Theory of the Electron Liquid*, Masters Series in Physics and Astronomy. Cambridge University Press (2005).

[4] Wikipedia contributors, *List of quantum chemistry and solid-state physics software — Wikipedia, the free encyclopedia*, `https://en.wikipedia.org/w/index.php?title=List_of_quantum_chemistry_and_solid-state_physics_software&oldid=942835156` [Online; accessed 2-March-2020] (2020).

[5] G. D. Mahan, *Many-Particle Physics*, Kluwer Academic, New York, 3rd ed. edn. (2000).

[6] J. P. Perdew, J. A. Chevary, S. H. Vosko, K. A. Jackson, M. R. Pederson, D. J. Singh and C. Fiolhais, *Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation*, Phys. Rev. B **46**, 6671 (1992), doi:10.1103/PhysRevB.46.6671.

[7] J. P. Perdew, K. Burke and M. Ernzerhof, *Generalized gradient approximation made simple*, Phys. Rev. Lett. **77**, 3865 (1996), doi:10.1103/PhysRevLett.77.3865.

[8] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli *et al.*, *Quantum espresso: a modular and open-source software project for quantum simulations of materials*, Journal of Physics: Condensed Matter **21**(39), 395502 (19pp) (2009).

[9] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo *et al.*, *Advanced capabilities for materials modelling with quantum espresso*, Journal of Physics: Condensed Matter **29**(46), 465901 (2017).

[10] K. Lejaeghere, G. Bihlmayer, T. Bjorkman, P. Blaha, S. Blugel, V. Blum, D. Caliste, I. E. Castelli, S. J. Clark, A. Dal Corso, S. de Gironcoli, T. Deutsch *et al.*, *Reproducibility in density functional theory calculations of solids*, Science **351**(6280), aad3000 (2016).

[11] G. Prandini, A. Marrazzo, I. E. Castelli, N. Mounet and N. Marzari, *Precision and efficiency in solid-state pseudopotential calculations*, npj Computational Materials **4**(1), 72 (2018).

[12] H. J. Monkhorst and J. D. Pack, *Special points for Brillouin-zone integrations*, Phys. Rev. B **13**(12), 5188 (1976).

[13] N. Marzari, D. Vanderbilt, A. De Vita and M. C. Payne, *Thermal Contraction and Disordering of the Al(110) Surface*, Phys. Rev. Lett. **82**(16), 3296 (1999).

[14] Y. Zhang, M. M. Ugeda, C. Jin, S.-F. Shi, A. J. Bradley, A. Martín-Recio, H. Ryu, J. Kim, S. Tang, Y. Kim, B. Zhou, C. Hwang *et al.*, *Electronic Structure, Surface Doping, and Optical Response in Epitaxial WSe 2Thin Films*, Nano Lett. **16**(4), 2485 (2016).

[15] G.-B. Liu, W.-Y. Shan, Y. Yao, W. Yao and D. Xiao, *Three-band tight-binding model for monolayers of group-VIB transition metal dichalcogenides*, Phys. Rev. B **88**(8), 085433 (2013).