Physics and Astronomy, School of Physical and Chemical Sciences,
University of Canterbury, Christchurch, New Zealand

Computing Crystal Band Structures with CP2K

Joseph Wilson

PHYS493 Project 2019

Supervisor: R. Reeves

Abstract

The band structure of a crystal contains a wealth of information about its electrical and material properties, and is of great interest in material science. However, generating a band structure diagram is a complex task and relevant software and documentation remain relatively inaccessible. This report aims to make band structure computations immediately accessible by 1) presenting an overview of the relevant theory and 2) outlining a work-flow using the open-source molecular dynamics software `CP2K`. The band structure of monocrystalline silicon is computed with `CP2K` for various choices of exchange-correlation functional, and the resulting band structure diagrams of Si and GaAs are qualitatively compared to the literature. The project is done in the hope that it may jump-start other investigations which are interested in or which could benefit from accessible band structure calculation.

Contents

1 Introduction

The band structures of crystalline solids, including semiconductors and minerals, encode information about many of the materials' optical, mechanical and electronic properties. For instance, the band structure of monocrystalline silicon reveals that it is a semiconductor and that the bandgap is indirect. These properties are of great interest when designing semiconductors for use in transistors, for example. Information about materials' absorption and emission spectra can be obtained from the band structure, which are of great interest when designing LEDs and lasers [5].

This report serves as a guide to performing band structure calculations using the open-source software package `CP2K` for quantum chemistry and solid state physics. An overview of the theory underlying band structure calculations, *density functional theory* (DFT), is presented in relation to `CP2K` in chapter 2. A guide to configuring `CP2K` input files and running them is given in appendix A. Provided with this report is a USB drive which contains two Python scripts which were developed to 1) read `CP2K` band structure output and 2) generate band structure diagrams. A working example `CP2K` input file for monocrystalline silicon is also present, and all three files are reproduced in appendix B.

1.1 Background Band Theory

Quantum mechanics predicts that a bound electron exists in (a superposition of) discrete orbitals of definite energy. For simplicity, consider a one-dimensional analogue of the hydrogen atom, whose discrete energy states are $\psi_n(x)$, where n is the *principle quantum number*. If another hydrogen nucleus is introduced far away from the first, its associated orbitals $\psi'_n(x)$ will be identical to $\psi_n(x)$ except translated in space. If the two nuclei are brought sufficiently close that their orbitals overlap, the electron will exist in *molecular orbitals* which are a combination of $\psi_n(x)$ and $\psi'_n(x)$. For a given n, two molecular orbitals will form; one of lower energy from an in-phase combination $\Psi_n^{(+)}(x) \propto \psi_n(x) + \psi'_n(x)$, and one of higher energy from an anti-phase combination, $\Psi_n^{(-)}(x) \propto \psi_n(x) - \psi'_n(x)$. If a third hydrogen nucleus is introduced, a similar mixing effect will occur, resulting in a triplet of molecular orbitals of slightly differing energies for each n.

If N hydrogen nuclei are arranged in a one dimensional lattice, there will be N distinct molecular orbitals with closely spaced energies. Furthermore, the molecular orbitals will begin to exhibit a definite periodicity (see figure 1.1). More precisely, they will begin to take the form $\Psi_n^{(k)}(x) = e^{ikx}u_n(x)$ for some k, where $u_n(x)$ is some function with the periodicity of the lattice. As N becomes large, the N distinct values of k become ever more closely spaced, so that in the limit k varies continuously. Thus, we may identify each molecular orbital by the principle quantum number n (now called the band number) and

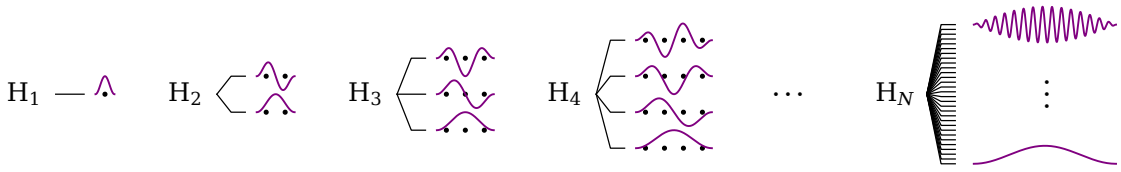


Figure 1.1: Schematic of band phenomena arising in a one-dimensional array of hydrogen nuclei. The vertical axis represents higher and lower energies of the purple molecular orbitals (not to scale). As N increases, the molecular orbitals (shown here for $n = 1$) form a dense energy band.

a wavenumber k.

This is the essence of Bloch's theorem, which states that the energy eigenstates of an electron in a periodic potential $V(\boldsymbol{r}) = V(\boldsymbol{r} + \boldsymbol{R})$, will have the form

$$\psi_{n\boldsymbol{k}}(\boldsymbol{r}) = e^{i\boldsymbol{k}\boldsymbol{r}} u_n(\boldsymbol{r}),$$

where $u_n(\boldsymbol{r}) = u_n(\boldsymbol{r} + \boldsymbol{R})$ shares the periodicity of the potential [5].

We may consider the energies associated with each eigenstate $\psi_{n\boldsymbol{k}}(\boldsymbol{r})$ as a function of the wavevector $\boldsymbol{k}$ to obtain the band structure.

$$\hat{H}\psi_{n\boldsymbol{k}}(\boldsymbol{r}) = E_{n\boldsymbol{k}}\psi_{n\boldsymbol{k}}(\boldsymbol{r}) \qquad \longrightarrow \qquad E_n(\boldsymbol{k}) \tag{1.1}$$

Plotting $E_n(\boldsymbol{k})$ against $\boldsymbol{k}$ for many band numbers n results in a *band structure diagram*, as in figure 1.2. This, $E_n(\boldsymbol{k})$ is the central object of interest concerning a crystal's band structure, and finding it boils down to solving the Schrödinger equation (1.1) for the entire crystal. However, this proves to be difficult for real materials, since (1.1) as it stands quickly becomes intractable for many electrons.

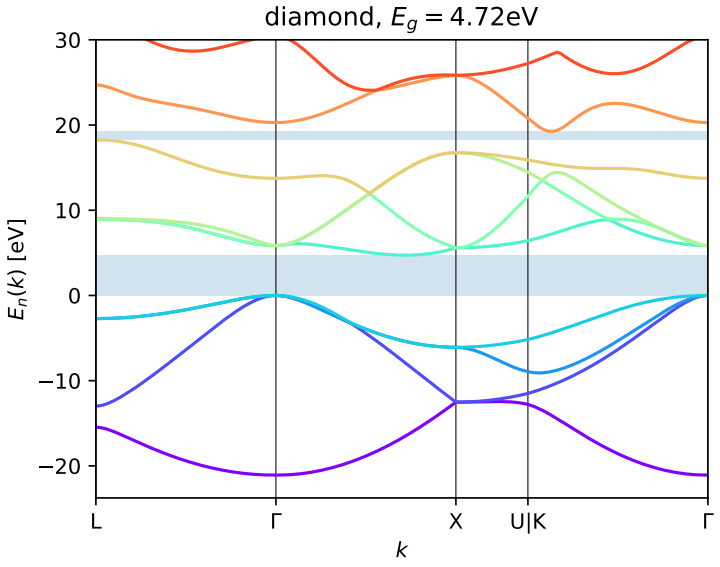


Figure 1.2: An example band structure diagram of diamond generated with `CP2K`.

2 Overview of Density Functional Theory

The software CP2K is capable of computing band structures by using a numerical *Gaussian plane wave* (GPW) method of solving the *Kohn–Sham* equations. This chapter will outline the prerequisite theory and present a brief first-look of the Kohn–Sham equations and the GPW method. The level of theoretical introduction is intended to be enough to enable comfortable use and comprehension of the CP2K software.

2.1 The Quantum Many-Body Problem

To determine the properties of a general many-body quantum system, we are interested in solving the Schrödinger equation $\hat{H}\Psi = E\Psi$ with the system's corresponding Hamiltonian $\hat{H}$. Treating the nuclei as point charges, this many-body Hamiltonian consists of terms for the kinetic energies of the system's N electrons and M nuclei, and the inter-electron, electron–nuclear, and inter-nuclear Coulomb interactions (written here respectively, in atomic units):

$$\begin{aligned}\hat{H} = &- \underbrace{\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2}_{\text{electron E.K.}} - \underbrace{\sum_{j=1}^{M} \frac{1}{2m_j}\nabla_j^2}_{\text{nuclear E.K.}} \\ &+ \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{i-1} \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}}_{\text{inter-electron C.I.}} - \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} \frac{Z_j}{|\boldsymbol{r}_i - \boldsymbol{R}_j|}}_{\text{electron–nuclear C.I.}} + \underbrace{\sum_{i=1}^{M}\sum_{j=1}^{i-1} \frac{Z_i Z_j}{|\boldsymbol{R}_i - \boldsymbol{R}_j|}}_{\text{inter-nuclear C.I.}}\end{aligned} \tag{2.1}$$

In the non-relativistic regime, this Hamiltonian is exact—but complex. To simplify the problem, we note that the nuclei are $> 10^3$ times more massive than—and experience interactions of similar strength to—the electrons, and therefore are able to be considered stationary on the electronic time-scale [8]. This decoupling of the nuclear and electronic dynamics is known as the Born–Oppenheimer approximation, under which the system's Hamiltonian (2.1) reduces to:

$$\hat{H} = - \underbrace{\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2}_{\text{electron E.K.}} + \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{i-1} \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}}_{\text{inter-electron C.I.}} + \underbrace{\frac{1}{2}\sum_{i=1}^{N} \overbrace{\sum_{j=1}^{M} \frac{-Z_j}{|\boldsymbol{r}_i - \boldsymbol{R}_j|}}^{V(\boldsymbol{r})}}_{\text{electron–nuclear C.I.}} \tag{2.2}$$

However, the problem still remains practically intractable, for the following reasons [9]:

- The stationary wavefunction $\Psi(\boldsymbol{r}_1, ..., \boldsymbol{r}_N)$ is a function of the positions of all N electrons, or of the $3N$ coordinates. For bulk materials, $N \sim 10^{26}$ and so an explicit representation of Ψ is simply unwieldy.

- Since the Hamiltonian contains an inter-electron interaction term, the electrons' wavefunctions are correlated, rendering it impossible to separate (2.2) into N tractable, single-body problems.

- The inter-electron interaction is too strong to be regarded as a perturbation treatable with perturbation theory.

2.2 The Electron Density

A significant conceptual leap was made towards approximating the many-body problem by Thomas and Fermi in the late 1920s, when the wavefunction $\Psi(\boldsymbol{r}_1, \dots, \boldsymbol{r}_N)$ was replaced by the *electron density* $\rho(\boldsymbol{r})$ as the central unknown variable. The electron density measures the electron occupancy of the infinitesimal volume of space at $\boldsymbol{r}$, and can be defined in terms of the normalised N-electron wavefunction as follows:

$$\begin{aligned}\rho(\boldsymbol{r}) &= \sum_{i=1}^{N} \int \prod_{j \neq i} \mathrm{d}\boldsymbol{r}_j \left|\Psi(\boldsymbol{r}_1, \dots, \boldsymbol{r}_{i-1}, \boldsymbol{r}, \boldsymbol{r}_{i+1}, \dots, \boldsymbol{r}_N)\right|^2 \\ &= N \int \mathrm{d}\boldsymbol{r}_2 \cdots \mathrm{d}\boldsymbol{r}_N \left|\Psi(\boldsymbol{r}, \boldsymbol{r}_2, \dots, \boldsymbol{r}_N)\right|^2 \end{aligned} \tag{2.3}$$

The ith term in the sum is to be interpreted as the total probability density of the ith electron being observed at location $\boldsymbol{r}$. Since electrons are indistinguishable, each term in this sum is identical, and hence the definition simplifies to the second line. This change of perspective is beneficial because the electron density is a much simpler object than the wavefunction, in the sense that it is only a function of space (3 d.o.f.) whereas the Ψ is a function of the system's configuration ($3N$ d.o.f.).

At the same time, Thomas and Fermi proposed an approximate Hamiltonian in the form of a functional of electron density, where they derived the kinetic energy term by crude analogy with a uniform electron gas.

$$H_{\mathrm{TF}}[\rho] = \underbrace{C_{\mathrm{KE}} \int \mathrm{d}\boldsymbol{r}\, \rho(\boldsymbol{r})^{\frac{5}{3}}}_{\text{kinetic energy}} + \underbrace{\frac{1}{2} \int \mathrm{d}\boldsymbol{r}\ \mathrm{d}\boldsymbol{r}' \frac{\rho(\boldsymbol{r}) n(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|}}_{\substack{\text{Hartree energy} \\ \text{(inter-electron C.I.)}}} + \underbrace{\int \mathrm{d}\boldsymbol{r}\, \rho(\boldsymbol{r}) V(\boldsymbol{r})}_{\substack{\text{external potential} \\ \text{(electron–nuclear C.I.)}}} \tag{2.4}$$

However, in this form, Thomas–Fermi theory fails to predict any type of bonding between atoms. Most of the error is due to the kinetic energy term, and because of the negligence of quantum phenomena such the *exchange and correlation interactions* between electrons.

2.3 The Hohenberg–Kohn Theorems

In 1964, another leap was made when Hohenberg and Kohn showed that an exact formulation of the energy functional in terms of electron density was possible, by proving two salient theorems relating to the electron density.

Theorem 1 (Hohenberg–Kohn) *The groundstate electron density $\rho_0(\boldsymbol{r})$ uniquely determines the external potential $V(\boldsymbol{r})$ to within a constant.*

Since both the potential $V[\rho_0]$ and number of electrons $N = \int \mathrm{d}\boldsymbol{r}\, \rho_0(\boldsymbol{r})$ are determined by the groundstate electron density, the system in question is totally specified by $\rho_0(\boldsymbol{r})$. Thus, the groundstate expectation value of any observable—including the total energy—may be viewed as a functional of $\rho_0(\boldsymbol{r})$, allowing us to write $H[\rho] \equiv \langle \Psi[\rho] | (\hat{T} + \hat{V}[\rho]) | \Psi[\rho] \rangle$.

While this theorem asserts the existence of an exact total energy functional (which Thomas and Fermi sought to approximate), its explicit form remains unknown [3]. However, we may split the exact energy functional into two terms; one an unknown functional, and the other the contribution of the external potential:

$$H[\rho] = F[\rho] + \underbrace{\int \mathrm{d}\boldsymbol{r}\, \rho(\boldsymbol{r}) V(\boldsymbol{r})}_{\text{external functional, } V[\rho]} \tag{2.5}$$

The functional $F[\rho]$ is unknown, but universal: it is the same for all N-electron systems. That which distinguishes the unique physical system is contained within $V[\rho]$.

The second Hohenberg–Kohn theorem states a very powerful result concerning the total energy functional.

Theorem 2 (Hohenberg–Kohn) *The true groundstate electron density $\rho_0(\boldsymbol{r})$ uniquely minimises the energy functional, i.e., $H[\rho_0] \leq H[\rho]$ for all physically-attainable $\rho(\boldsymbol{r})$.*

Therefore, the groundstate properties of a many-electron system can be determined by minimising $H[\rho]$ with respect to $\rho(\boldsymbol{r})$ subject to the constraint that $\int \mathrm{d}\boldsymbol{r}\, \rho(\boldsymbol{r}) = N$. This minimisation can be accomplished through the method of Lagrange multipliers, yielding an Euler-Lagrange equation involving $\rho(\boldsymbol{r})$ and a constant Lagrange multiplier μ.

$$\mu = \frac{\delta F[\rho]}{\delta \rho} + V(\boldsymbol{r}) \tag{2.6}$$

If $F[\rho]$ was known, then (2.6) could be solved for $\rho(\boldsymbol{r})$ using an iterative numerical method. However, approximations to $F[\rho]$ have had little success (e.g., (2.4) from Thomas-Fermi theory) because the kinetic energy functional is problematic.

2.4 The Kohn–Sham Scheme

One year after the Hohenberg–Kohn theorems were published, Kohn and Sham [6] devised a cunning method for approximating the groundstate density which avoided having to estimate the kinetic energy functional. The method involves translating the system of many interacting electrons into a fictitious auxiliary non-interacting system whose potential term

is chosen such that the auxiliary system's solution yields the same groundstate density (and thus exhibits the same properties) as the physical system. This is desirable because the non-interacting kinetic energy functional is known.

More concretely, the method involves transforming (2.6), whose solution is the groundstate density, into a system of single-electron Schrödinger-like equations, known as the Kohn–Sham equations, which can be solved iteratively.

To transform the Euler–Lagrange equation (2.6), the universal functional is partitioned into three terms:

$$F[\rho] = T_\mathrm{s}[\rho] + E_\mathrm{H}[\rho] + E_\mathrm{XC}[\rho] \tag{2.7}$$

The first two are known exactly, and the last is unknown but constitutes a small amount of the total energy.

- $T_\mathrm{s}[\rho]$ is the exact kinetic energy of a *non*-interacting electron gas of density $\rho(\boldsymbol{r})$,

$$T_\mathrm{s}[\rho] = -\frac{1}{2}\sum_{i=1}^{N}\int \mathrm{d}\boldsymbol{r}\,\psi_i^*(\boldsymbol{r})\nabla^2\psi_i(\boldsymbol{r}), \tag{2.8}$$

  where $\psi_i(\boldsymbol{r})$ are the wavefunctions of the N non-interacting electrons.

- $E_\mathrm{H}[\rho]$ is the Hartree energy, which accounts for the majority of the inter-electron interaction energy:

$$E_\mathrm{H}[\rho] = \frac{1}{2}\int \mathrm{d}\boldsymbol{r}\,\mathrm{d}\boldsymbol{r}'\,\frac{\rho(\boldsymbol{r})\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}$$

- $E_\mathrm{XC}[\rho]$ is an unknown *exchange-correlation functional*, which can be implicitly defined as the difference between the exact kinetic energy $T[\rho]$ and inter-electron interaction $E_\mathrm{ee}[\rho]$ and the two previous terms.

$$E_\mathrm{XC}[\rho] = \underbrace{(T[\rho] + E_\mathrm{ee}[\rho])}_{\text{exact, unknown}} - \underbrace{(T_\mathrm{s}[\rho] + V_\mathrm{H}[\rho])}_{\text{approximate, known}}$$

  In practice, this term is approximated—and with better success than direct approximation of the kinetic energy, due to it accounting for less of the total energy.

Substituting (2.7) into (2.6) yields

$$\mu = \frac{\delta T_\mathrm{s}[\rho]}{\delta\rho} + \overbrace{\int \mathrm{d}\boldsymbol{r}'\,\frac{\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}}^{\frac{\delta E_\mathrm{H}[\rho]}{\delta\rho}} + \frac{\delta E_\mathrm{XC}[\rho]}{\delta\rho} + V(\boldsymbol{r}) = \frac{\delta T_\mathrm{s}[\rho]}{\delta\rho} + V_\mathrm{KS}(\boldsymbol{r}), \tag{2.9}$$

where we define the Kohn–Sham potential:

$$V_\mathrm{KS}(\boldsymbol{r}) = \int \mathrm{d}\boldsymbol{r}'\,\frac{\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} + \frac{\delta E_\mathrm{XC}[\rho]}{\delta\rho} + V(\boldsymbol{r})$$

The new Euler–Lagrange equation (2.9) now resembles (2.6)—and it is important to keep in mind that it *is* the same equation, only rearranged, so that both equations' solutions are the groundstate density $\rho_0(\boldsymbol{r})$.

Equation (2.9) is the minimised solution to the following total energy functional of the auxiliary system,

$$H_{\mathrm{KS}}[\rho] = T_{\mathrm{s}}[\rho] + \int \mathrm{d}\boldsymbol{r}\, \rho(\boldsymbol{r}) V_{\mathrm{KS}}(\boldsymbol{r}), \tag{2.10}$$

in the same way that (2.6) is the minimisation of (2.5).

At this point, we note that since the auxiliary system consists of non-interacting fermionic electrons, its solution takes the form of a totally-antisymmetric *Slater determinant*,

$$\begin{aligned}\Psi_{\mathrm{KS}} &= \frac{1}{\sqrt{N!}} \det\left[\psi_1(\boldsymbol{r}_1)\psi_2(\boldsymbol{r}_2)\cdots\psi_N(\boldsymbol{r}_N)\right] \\ &\equiv \frac{1}{\sqrt{N!}} \prod_{\sigma\in S_N} (-1)^{\sigma} \psi_{\sigma(1)}(\boldsymbol{r}_1)\psi_{\sigma(2)}(\boldsymbol{r}_2)\cdots\psi_{\sigma(N)}(\boldsymbol{r}_N),\end{aligned}$$

where S_N is the permutation group of N objects, and $(-1)^{\sigma} = 1$ if the permutation σ is even and -1 if σ is odd. The Kohn-Sham wavefunction may be thought of as the antisymmetrisation of a wavefunction of non-interacting particles, $\psi_1(\boldsymbol{r}_1)\psi_2(\boldsymbol{r}_2)\cdots\psi_N(\boldsymbol{r}_N)$. The electron density is therefore related to the Kohn–Sham wavefunction through

$$\rho(\boldsymbol{r}) = N \int \mathrm{d}\boldsymbol{r}_2 \cdots \mathrm{d}\boldsymbol{r}_N |\Psi_{\mathrm{KS}}(\boldsymbol{r}, \boldsymbol{r}_2, ..., \boldsymbol{r}_N)|^2 = \sum_{i=1}^{N} |\psi_i(\boldsymbol{r})|^2. \tag{2.11}$$

We may now write (2.10) in full, expanding $T_{\mathrm{s}}[\rho]$ with reference to (2.8) and $\rho(\boldsymbol{r})$ with reference to (2.11), to find a Schrödinger–like equation take form.

$$\sum_{i=1}^{N} \int \mathrm{d}\boldsymbol{r}\, \psi_i^*(\boldsymbol{r}) \hat{H}_{\mathrm{KS}} \psi_i(\boldsymbol{r}) = \sum_{i=1}^{N} \int \mathrm{d}\boldsymbol{r}\, \psi_i^*(\boldsymbol{r}) \left[-\frac{1}{2}\nabla^2 + V_{\mathrm{KS}}(\boldsymbol{r})\right] \psi_i(\boldsymbol{r})$$

Thus, we are left with N independent Kohn–Sham equations.

$$E_i \psi_i(\boldsymbol{r}) = \left[-\frac{1}{2}\nabla^2 + V_{\mathrm{KS}}(\boldsymbol{r})\right] \psi_i(\boldsymbol{r}) \tag{2.12}$$

Keep in mind that the these equations depend on the Kohn–Sham potential $V_{\mathrm{KS}}(\boldsymbol{r})$, which itself depends on the groundstate density $\rho_0(\boldsymbol{r})$. The wavefunctions and groundstate density must be solved for self-consistently, using sophisticated numerical methods.

$$V_{\mathrm{KS}}(\boldsymbol{r}) = V(\boldsymbol{r}) + \int \mathrm{d}\boldsymbol{r}' \frac{\rho_0(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} + V_{\mathrm{XC}}[\rho_0](\boldsymbol{r}) \longleftarrow \rho_0(\boldsymbol{r}) = \sum_{i=1}^{N} |\psi_i(\boldsymbol{r})|^2$$

$$\downarrow \qquad\qquad \uparrow$$

$$E_i \psi_i(\boldsymbol{r}) = \left[-\frac{1}{2}\nabla^2 + V_{\mathrm{KS}}(\boldsymbol{r})\right] \psi_i(\boldsymbol{r}) \longrightarrow$$

2.5 The Gaussian Plane Wave Implementation

We have encountered the Kohn–Sham scheme for solving the quantum many body problem. In order to solve the self-consistent Kohn–Sham equations numerically, computationally efficient representations of the density $\rho(\boldsymbol{r})$ and wavefunctions $\psi_i(\boldsymbol{r})$ are needed. The Gaussian plane wave (GPW) representation that `CP2K` implements uses plane waves and contracted Gaussian-type orbitals to represent $\rho(\boldsymbol{r})$ and $\psi_i(\boldsymbol{r})$, respectively. The method employs two different kinds of basis sets in order to benefit from the advantages of both, resulting in an efficient implementation [10].

2.5.1 Plane waves and pseudopotentials

Since we are interested in regular crystals, the external potential is periodic, i.e.,

$$V(\boldsymbol{r}) = V(\boldsymbol{r} + \boldsymbol{R}),$$

where $\boldsymbol{R}$ is any integer combination of primitive lattice vectors. By Bloch's theorem, the solutions to Schrödinger's equation for electrons in such a periodic potential have the form

$$\psi_{n\boldsymbol{k}}(\boldsymbol{r}) = e^{i\boldsymbol{k}\boldsymbol{r}} u_{n\boldsymbol{k}}(\boldsymbol{r}),$$

where $u_{n\boldsymbol{k}}(\boldsymbol{r})$ has the translational symmetry of the lattice. Each solution $\psi_{n\boldsymbol{k}}(\boldsymbol{r})$ is identified by two subscripts: the discrete band index n; and the wavevector $\boldsymbol{k}$, with which $\psi_{n\boldsymbol{k}}(\boldsymbol{r})$ varies quasi-continuously.

Since $|\psi_{n\boldsymbol{k}}(\boldsymbol{r})|^2 = |u_{n\boldsymbol{k}}(\boldsymbol{r})|^2$ is periodic, the electron density is also periodic. Therefore, plane waves of the form $\phi_{\boldsymbol{G}}(\boldsymbol{r}) = e^{i\boldsymbol{G}\boldsymbol{r}}$, where $\boldsymbol{G}$ is a k-vector in the principle Brillouin zone, serve as a natural basis for the periodic density $\rho(\boldsymbol{r})$.

$$\rho(\boldsymbol{r}) = \int \mathrm{d}\boldsymbol{G}\, \rho_{\boldsymbol{G}} \phi_{\boldsymbol{G}}(\boldsymbol{r})$$

We may discretise the Brillouin zone so that there are only finitely many coefficients $\rho_{\boldsymbol{G}}$ to determine. Thus, we may represent the density $\rho(\boldsymbol{r})$ by its Fourier coefficients $\rho_{\boldsymbol{G}}$.

$$\rho(\boldsymbol{r}) = \sum_{\boldsymbol{G}} \rho_{\boldsymbol{G}} \phi_{\boldsymbol{G}}(\boldsymbol{r}) \tag{2.13}$$

The fewer the coefficients, the lesser the computational cost—but at the expense of accuracy.

A practical problem with the plane wave representation arises near atomic nuclei. In the regions of the core (inner) electrons, $\rho(\boldsymbol{r})$ is large and forms a sharp cusp. In these regions, many terms are required to achieve numerical convergence. However, the inner shells of atoms in a lattice are tightly bound, and are hardly influenced by the presence of neighbouring atoms. These core electron states can therefore be assumed to be known from calculations of isolated atoms [5]. In fixing the core electron states, the valence electrons are found to be moving in an effective potential (a.k.a., a *pseudopotential*) which is a sum of the nuclear potentials and a contribution from the core electrons. The pseudopotential method requires fewer plane waves to accomplish numerical convergence, and is therefore more computationally efficient. However, the method relies on the existence of accurate pseudopotentials, which must be computed and specified in `CP2K` input files separately.

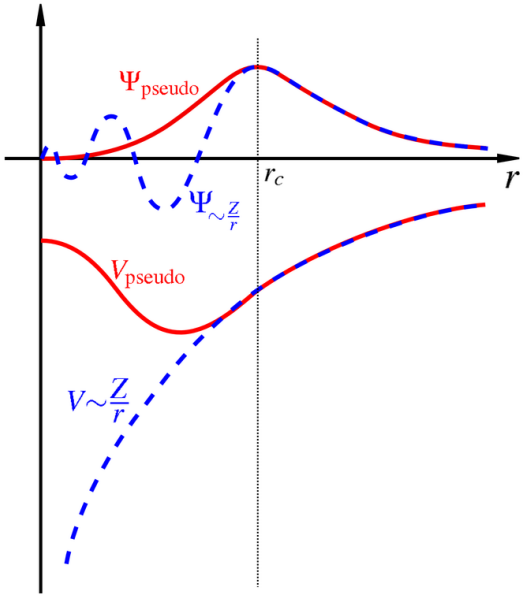


Figure 2.1: Illustration of the effect of the pseudopotential on the complexity of the solution. The dashed blue lines show the true potential and true wavefunction, while the red lines show the pseudopotential and its "simpler" solution, Ψ_{pseudo}. This requires fewer terms of its Fourier series than $\Psi_{\sim\frac{Z}{r}}$ to be represented accurately. (Image in public domain.)

2.5.2 Gaussian basis sets

The plane wave basis is not well-suited for representing electron wavefunctions. Instead, CP2K uses contracted Gaussian-type orbitals (GTOs) as a basis. GTOs have the form

$$\psi_{\text{GTO}}(\boldsymbol{r};\sigma,l,\boldsymbol{A}) = r^l e^{-\sigma(\boldsymbol{r}-\boldsymbol{A})^2},$$

where l is a shape factor and $\boldsymbol{A}$ is the centre of the distribution. Contracted GTOs, $\varphi_i(\boldsymbol{r})$, are constructed as follows,

$$\varphi_i(\boldsymbol{r};\boldsymbol{A}) = \sum_{j=1}^{N} c_{ij}\psi_{\text{GTO}}(\boldsymbol{r};\sigma_j,l_i,\boldsymbol{A}),$$

where the four coefficients c_{ij}, σ_j, l_i, and N collectively define each $\varphi_i(\boldsymbol{r};\boldsymbol{A})$. Gaussian basis functions are preferable for constructing the electron wavefunctions because they can be localised at the atomic nuclei, like the wavefunctions themselves [11]. Contracted Gaussian basis sets can be optimised to each atomic element and pseudopotential. CP2K requires basis sets to be specified for each atomic kind when using the Gaussian plane wave method to perform DFT computations.

3 Comparison to the Literature

3.1 Band Structure of Si

Monocrystalline silicon has a diamond-cubic structure, with a lattice constant $a_{\mathrm{Si}} = 5.431\,\text{Å}$, primitive lattice vectors

$$\boldsymbol{a} = \frac{a_{\mathrm{Si}}}{2}\begin{bmatrix}0\\1\\1\end{bmatrix}, \qquad \boldsymbol{b} = \frac{a_{\mathrm{Si}}}{2}\begin{bmatrix}1\\0\\1\end{bmatrix}, \qquad \boldsymbol{c} = \frac{a_{\mathrm{Si}}}{2}\begin{bmatrix}1\\1\\0\end{bmatrix},$$

and unit basis $\mathrm{Si}\,(0,0,0)\left(\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$. The band structure of mono-Si is well very studied, and its bandgap is known to be 1.12 eV [4, 7]. Its band structure was computed using the Kohn–Sham scheme as implemented in `CP2K`, along the standard k-path L–Γ–X–U|K–Γ. The computation was repeated for different choices of exchange-correlation functional, and a qualitative comparison of the results is shown in figure 3.1. See `mono-Si.inp` in appendix A for the `CP2K` input file that was used for the computation with the Perdew–Burke–Ernzerhof (PBE) functional. All original band structure diagrams were generated with the scripts provided in appendix B.2.

The most successful exchange-correlation functional of the ones that were trialled was determined to be the PBE functional, which belongs to the class of generalized gradient approximation (GGA) functionals. GGA functionals are generally the best for applications in solid state physics [8].

A mono-Si band structure diagram published in Chelikowsky, 1974 [1] was replicated with `CP2K` using the PBE functional and the results are shown side-by-side in figure 3.3. The four-electron pseudopotential `GTH-PBE-q4` and basis set `DZVP-GTH-PBE` were used, both of which are optimised for the PBE functional. The resemblance is remarkable; there is a clear correspondence between equivalent bands in the two diagrams, and all qualitative features are reproduced. Both diagrams underestimate the bandgap by $\sim 21\%$.

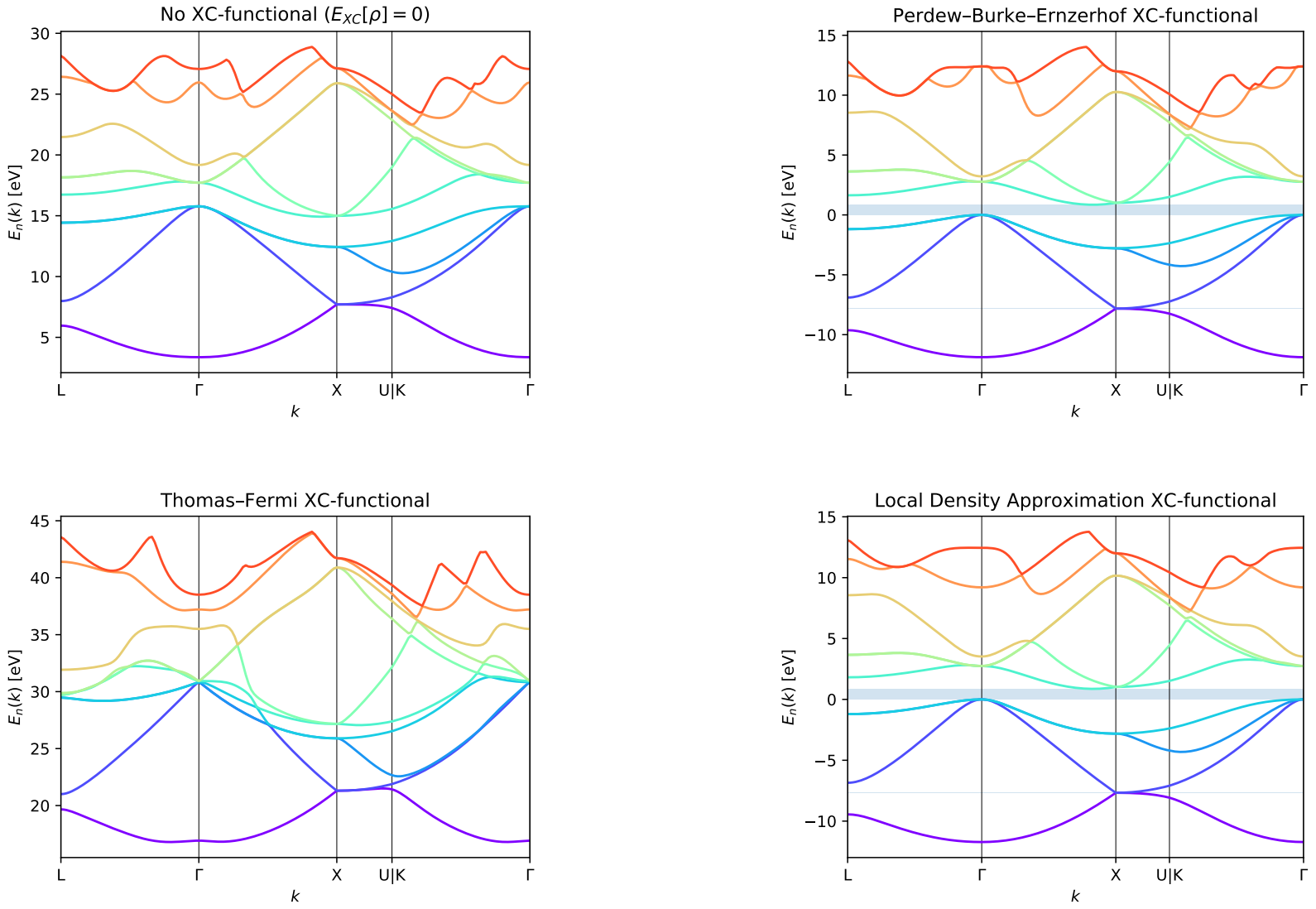


Figure 3.1: Comparison of band structures for monocrystalline silicon arising from different choices of $E_{\mathrm{XC}}[\rho]$. The local density approximation (LDA) functional predicts a bandgap of 0.793 eV, and Perdew–Burke–Ernzerhof (PBE) predicts 0.896 eV. No bandgap is predicted in the Thomas–Fermi model or if the exchange-correlation term is simply omitted from the Kohn–Sham potential.

3.2 Band Structure of GaAs

Gallium arsenide, a III–V direct bandgap semiconductor, has the same lattice vectors as silicon, but with lattice constant $a_{\mathrm{GaAs}} = 5.6535\,\text{Å}$ and unit basis Ga $(0,0,0)$ As $\left(\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$. The band structure of GaAs was computed using various exchange-correlation functionals, and again, the PBE functional yielded the best results. For gallium, the three-electron `GTH-PBE-q3` pseudopotential was used, and for arsenide, the five-electron `GTH-PBE-q5` pseudopotential was used. The basis sets were not necessarily optimised for the PBE functional and these pseudopotentials: different basis sets were simply tried until `CP2K` was able to successfully solve the system.

Figure 3.2 compares the resulting diagram to Krüger, 1993 [7]. The similarity is striking—however, there are some features that `CP2K` failed to reproduce.

1. The predicted fundamental bandgap is $\sim 50\%$ the size of the bandgap quoted by Krüger,
2. The predicted gap between the first and second bands is $\sim 37\%$ larger than quoted by Krüger,
3. There is no predicted gap between the second and third bands at the Γ point and between the fifth and sixth bands at the X point,
4. There is no clear splitting of the third and fourth bands near the L point.

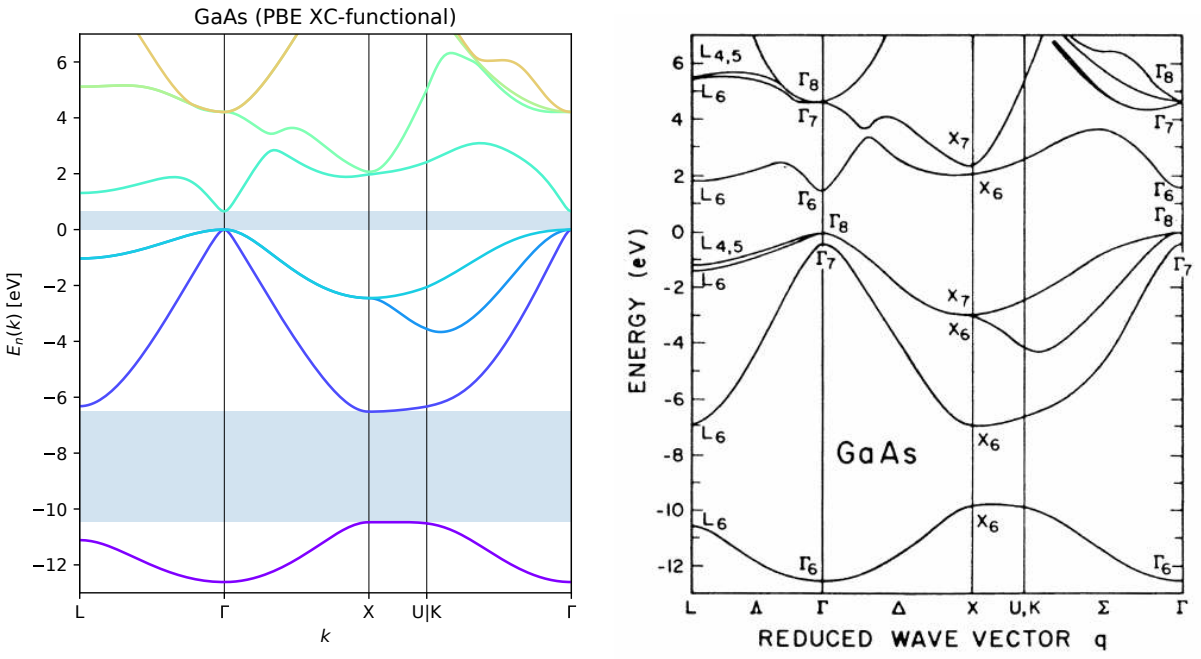


Figure 3.2: *(Left)* Band structure of GaAs compared to *(right)* published result [7]. The predicted bandgaps are 0.724 eV *(left)* and 1.45 eV *(right)* compared to the true value of 1.521 eV.

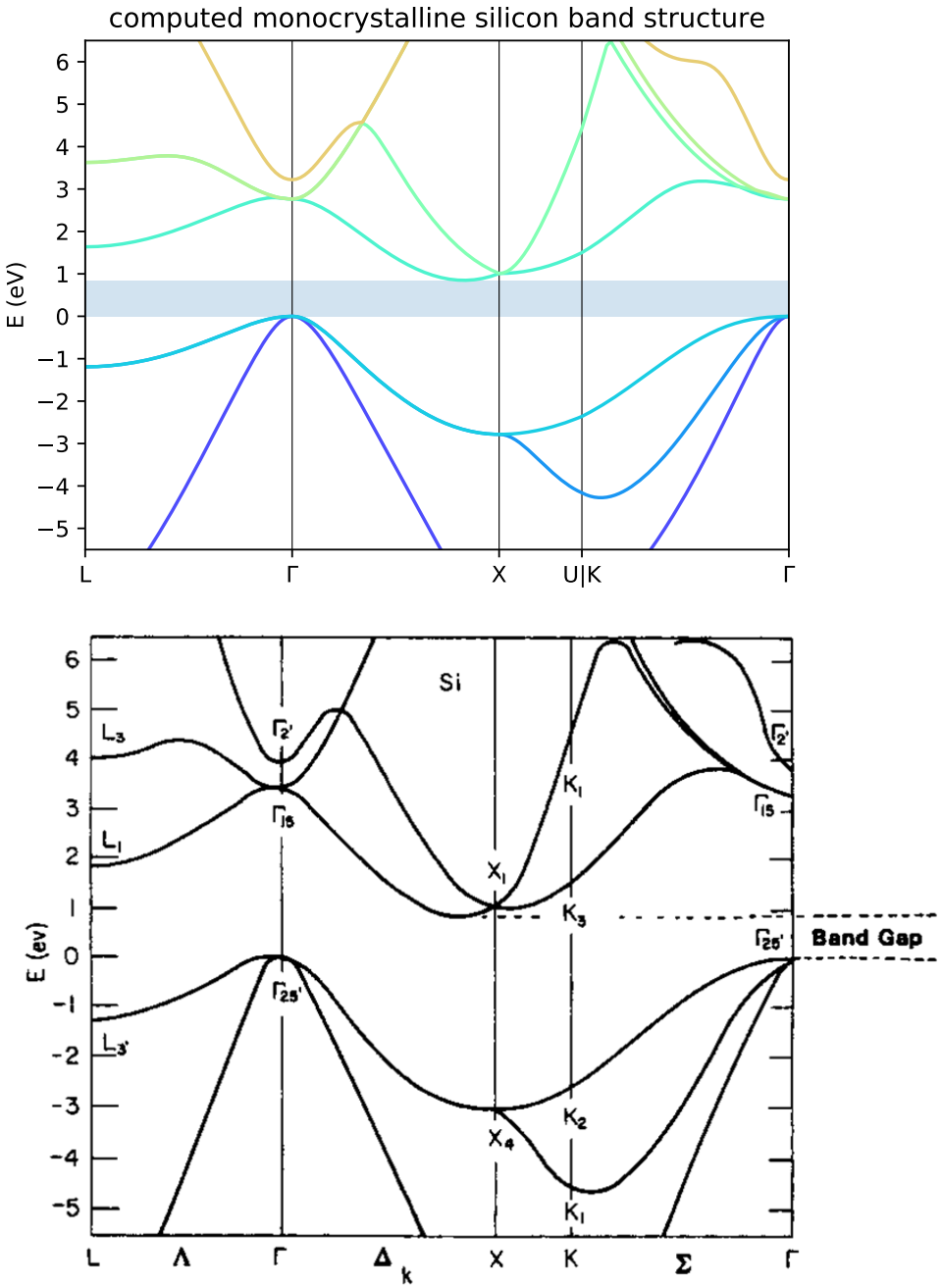


Figure 3.3: *(Top)* Band structure of mono-Si computed with `mono-Si.inp` compared to *(bottom)* published semi-empirical band structure [1]. The predicted bandgaps are 0.896 eV *(top)* and ~ 0.88 eV *(bottom)* compared to the true value of 1.12 eV.

4 Conclusions

Band diagrams for silicon and gallium arsenide were successfully produced with CP2K yielding close agreement to published literature. The most notable discrepancy was systematic underestimation of the fundamental bandgap—which is a well known problem of density functional theory. The Perdew–Burke–Ernzerhof exchange-correlation functional proved to be the most accurate for both these simple semiconductors; however, not all exchange-correlation functionals implemented in CP2K were compared, and only their default configurations were considered.

A deeper understanding of the advantages and disadvantages of the various pseudopotentials, basis sets and exchange-correlation functionals would benefit the quality of results obtained with CP2K. Further directions of inquiry could include how to specify crystal defects or medium boundaries in CP2K, making possible information-rich band structure computations for many more interesting systems.

Acknowledgements

I owe special thanks to Roger Reeves of the School of Physical and Chemical Sciences, University of Canterbury for trusting me with a self-directed a project and for being willing to supervise it. I would also like to thank Orlon Petterson for kindly installing the CP2K software on the physics department servers.

Bibliography

[1] James R. Chelikowsky and Marvin L. Cohen. Electronic structure of silicon. *Physical Review B*, 10(12):5095–5107, 1974.

[2] Peter Draper. Condensed matter electronic structure theory. University Zürich, 2016. Section 7.1.

[3] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871, Nov 1964.

[4] Andreas Hössinger. *Simulation of Ion Implantation for ULSI Technology*. PhD thesis, Technische Universität Wien Institute for Microelectronics, 7 2000.

[5] T. Ihn. *Semiconductor Nanostructures: Quantum States and Electronic Transport*. OUP Oxford, 2010.

[6] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, Nov 1965.

[7] Michael Rohlfing, Peter Krüger, and Johannes Pollmann. Quasiparticle band-structure calculations for c, si, ge, gaas, and sic using Gaussian-orbital basis sets. *Physical Review B*, 48(24):17791–17805, 1993.

[8] Philip Peter Rushton. Towards a non-local density functional description of exchange and correlation. 2002.

[9] Paul Robert Tulip. Dielectric and lattice dynamical properties of molecular crystals via density functional perturbation theory: Implementation within a first principles code. 2002.

[10] Joost VandeVondele, Matthias Krack, Fawzi Mohamed, Michele Parrinello, Thomas Chassaing, and Jürg Hutter. Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. *Computer Physics Communications*, 167(2):103 – 128, 2005.

[11] R. Vuilleumier. CP2K: the Gaussian plane wave (GPW) method. Lecture notes, Département de chimie, Ecole normale supérieure, 2018.

A Using the CP2K Software Package

CP2K is an open source software package which can be found at https://www.cp2k.org. A working copy of CP2K version 7.0 (development version) was installed on the Physics Department server physhpc2 in December, 2018. The program may be accessed via ssh remote login from any machine connected to the university's network. Consult the Physics and Astronomy departments' I.T. manager (room 322 of the Ernest Rutherford Building as of early 2019) for further instructions on how to set up a remote login.

Once connected to physhpc2, the CP2K program is located at /usr/local/bin/cp2k and should be able to be invoked with the command CP2K. Run the command cp2k -v to verify the installation; the versions description should be printed to the console. Official documentation can be found at https://manual.cp2k.org/, although it is very terse and not helpful as an introduction to the software. More helpful unofficial documentation (including lecture slides and theses) can be found at https://www.cp2k.org/docs.

A.1 Running CP2K

CP2K is a command-line program which takes an input file (conventionally ending with .inp) and writes to an output file (conventionally ending with .out) and zero or more other output files. To perform the computations specified by the input file project.inp and write the output to project.out, the following command may be used:

```
cp2k -i project.inp -o project.out
```

Alternatively, to simultaneously write the output to a file and monitor it in the console, one may use the following:

```
cp2k project.inp | tee project.out
```

The tee command pipes output from cp2k project.inp both to the console and to project.out. This pattern of commands is better suited to interactive use.

Some CP2K programs reference other source files from within the main .inp file. Relative file-paths can be used, but for frequently used source files (such as basis sets and pseudopotential definitions for various elements), CP2K will also search in the directory specified by the environment variable $CP2K_DATA_DIR. To test whether the environment variable is set, use the command echo $CP2K_DATA_DIR and its value (if any) will be shown. Set this environment variable by inserting a line similar to

```
export CP2K_DATA_DIR='~/cp2k_data'
```

in the user's ~/.bashrc to keep common CP2K source files organised in a single directory irrespective of each .inp file's location.

A.2 Performing Band Calculations

`CP2K` input files contain keywords (e.g., `KEYWORD`) and parameters (e.g., `PARAMETER value1 value2 ...`) organised into sections (delimited with `&SECTION` and `&END SECTION`). Single-line comments begin with a hash or exclamation point (`# comment` or `! comment`).

A fully-commented minimal working example input file `mono-Si.inp` which computes the band structure of monocrystalline silicon, is given in appendix B.1. The file includes the following main sections:

- `&GLOBAL`: In which the project name and computation type are specified,
- `&FORCE_EVAL`: For parameters relating to computing energies and forces. This section contains the following subsections:
  - `&SUBSYS`: Contains the definition of the atomic system. In the example, this is where the primitive cell of monocrystalline silicon is defined.
  - `&DFT`: Section for performing density functional theory calculations and processing the results, which includes the following subsections:
    - `&PRINT`: Section specifying which data to output, containing one subsection:
      - `&BAND_STRUCTURE`: Defines the k-path along which to sample and output the band structure.

Other sections may be added to enable more features or to refine computations. A complete index tree of all sections and their parameters can be found at `https://manual.cp2k.org/trunk/`.

A.2.1 Defining the cell structure: The `&SUBSYS` section

With reference the example file `mono-Si.inp` in appendix B.1, the `&SUBSYS` section contains a `&CELL` section which defines the primitive cell of monocrystalline silicon.

```
11      &CELL                      ## Unit cell parameters
12         A [angstrom]  0.000000  2.715475  2.715475
13         B [angstrom]  2.715475  0.000000  2.715475
14         C [angstrom]  2.715475  2.715475  0.000000
15         PERIODIC XYZ            # Periodicity of the unit cell; in most cases, this should be
           ↪ the same as &FORCE_EVAL/&DFT/&POISSON/PERIODIC
16      &END CELL
```

The parameters `A`, `B`, and `C` define the primitive lattice vectors, and `PERIODIC XYZ` enables periodicity in all directions. The unit option `[angstrom]` can be substituted for `[nm]` or `[bohr]`. The primitive cell is then populated with silicon atoms in the `&COORD` section:

```
22      &COORD                     ## Coordinates of atoms in primitive cell
23         SCALED                   # Relative to primitive lattice vectors
24         Si  0.00  0.00  0.00
25         Si  0.25  0.25  0.25
26      &END COORD
```

It is important that the crystal is defined in terms of the primitive unit cell when doing band calculations, as opposed to a diamond-cubic unit cell like the following:

```
      &CELL
         ABC [angstrom] 5.43095 5.43095 5.43095 # Simple cubic cell with mono-Si lattice
          ↪ constant
         PERIODIC XYZ
      &END CELL
      &COORD
         SCALED
         Si  0.00  0.00  0.00
         Si  0.00  0.50  0.50
         Si  0.50  0.00  0.50
         Si  0.50  0.50  0.00
         Si  0.25  0.25  0.75
         Si  0.25  0.75  0.25
         Si  0.75  0.25  0.25
         Si  0.75  0.75  0.75
      &END
```

This is because `CP2K` defines the reciprocal lattice vectors in terms of the given lattice vectors `A`, `B`, and `C`; they are not reduced to primitive lattice vectors internally. Thus, if a the silicon crystal is defined as above, then any k-points which are defined in terms of the reciprocal lattice vectors (in, for example, `&KPOINT_SET` sections) will not correspond to standard k-points which are seen in the literature, leading to confusing results. The computation time is also reduced significantly when symmetry is taken the most advantage of in using the primitive cell definition (for this example, the computation time is ~ 6 times shorted).

The `&KIND` subsection of `&SUBSYS` in which atom kinds are defined has not yet been discussed; its role will be explained shortly.

A.2.2 Configuring the solver: The `&DFT` section

The `Quickstep` algorithm implemented by `CP2K` uses the mixed Gaussian plane wave method to perform DFT calculations. The method relies on having suitable pseudopotentials in order to achieve numerical convergence without the need of an unreasonably large basis of plane waves. The method also requires that basis sets be defined.

Pseudopotentials and basis sets can be specified in the `CP2K` input file for each atomic kind. However, it is generally easier to refer to a file containing many named pseudopotentials or basis sets for various elements, and to specify by name which one to use for each atomic kind. The parameters `POTENTIAL_FILE_NAME` and `BASIS_SET_FILE_NAME` located in the `&DFT` section specify the path to such files (which should be placed in the `$CP2K_DATA_DIR` directory).

A typical pseudopotential file contains many entries in a format similar to the following:

```
Si GTH-PBE-q4
    2    2
     0.44000000    1    -6.26928833
    2
```

```
      0.43563383    2     8.95174150    -2.70627082
                                          3.49378060
      0.49794218    1     2.43127673
```

A typical basis set entry looks similar, containing coefficients for each basis orbital in the set (see 2.5.2). The exact format for `CP2K` basis sets is described at `https://www.cp2k.org/basis_sets`.

The particular pseudopotential and basis set is selected for each atomic kind in a `&SUBSYS/&KIND` section. For instance, in the example file `mono-Si.inp`, the atomic kind of silicon is defined like so:

```
      &KIND Si                ## Define a species of atom (repeated for each atom kind in cell)
         # Selected basis set and pseudopotential from files specified in &FORCE_EVAL/&DFT
         BASIS_SET DZVP-GTH-PBE
         POTENTIAL GTH-PBE-q4
      &END KIND
```

The "`PBE`" in the name `DZVP-GTH-PBE` specifies that the basis set is optimised for the Perdew–Burke–Ernzerhof (PBE) exchange-correlation functional. The name `GTH-PBE-q4` is given to the silicon pseudopotential which has four valence electrons (`q4`) and is also optimised for PBE.

The form of the exchange-correlation functional $E_{\mathrm{XC}}[\rho]$ to be implemented is specified in the `&DFT/&XC` section. Since it is the only approximation used in the Kohn–Sham scheme, this choice of exchange-correlation functional is what determines accuracy in practice. There are many different exchange-correlation functionals programmed into `CP2K` with different strengths and weaknesses, and many of them accept extra parameters so they may be fine-tuned for each use case.

It is advisable that, in order to maximise accuracy in computations, the choice of pseudopotential is one which is optimised for the current exchange-correlation functional, and vice versa [2].

A.2.3 Getting band information: The `&PRINT` section

The `&PRINT` section is used to specify which data is to be printed to the output file or saved to an external file. It contains one subsection, `&BAND_STRUCTURE`, which defines a path in k-space along which the band structure is to be evaluated. The `ADDED_MOS` parameter specifies how many conduction bands to include in the output, in addition to the occupied valence bands. The `FILE_NAME` parameter specifies a file path to append the band structure data to.

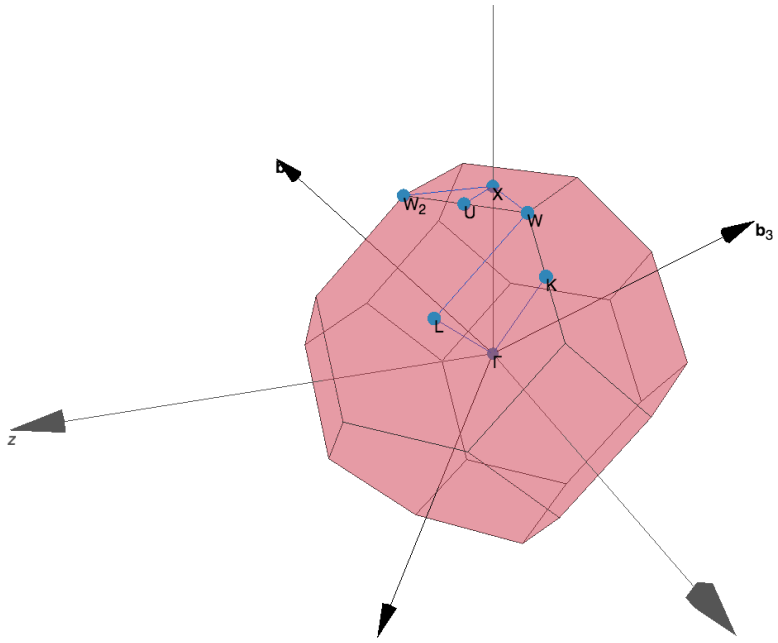
In addition to these, there are one or more `&KPOINT_SET` sections.

```
            &KPOINT_SET         ## Define a k-path linearly interpolating between defined
             ↪ k-points
               UNITS B_VECTOR    # Coordinates are multiples of reciprocal lattice vectors
                ↪ (1/Å)
               SPECIAL_POINT L  0.500  0.500  0.500
               SPECIAL_POINT Γ  0.000  0.000  0.000
               SPECIAL_POINT X  0.500  0.000  0.500
```

```
48                NPOINTS 50        # Number of interpolation points between each special k-point
49              &END KPOINT_SET
```

Each `&KPOINT_SET` section defines a k-path which linearly interpolates between particular k-points defined in order by `SPECIAL_POINT` parameters, each of which may take three coordinates, or a k-point label and three coordinates. The `NPOINTS` parameter defines the number of k-points to sample in each segment of the k-path. Multiple `&KPOINT_SET` sections can be defined in order to vary the number of interpolation points between k-points.

An extremely helpful tool for finding and visualising k-paths and generating `&KPOINT_SET` sections automatically is See$k$-path, which is free to use at `https://tools.materialscloud.org/seekpath`. For example, after inputting the primitive lattice vectors and unit basis for GaAs, See$k$-path suggests the k-path Γ—X—U|K—Γ—L—W—X—W_2 and displays the following visualisation of the path in the Brillouin zone:



B Code

B.1 Minimal Working CP2K Input File

Shown below is the example CP2K input file mono-Si.inp:

```
# You may define pre-compiler variables for organisational purposes like so:
@SET PROJECT mono-Si
&GLOBAL                          ## Global parameters
   PROJECT ${PROJECT}             # Project name; output files are prefixed with this
   RUN_TYPE ENERGY                # Kind of calculation; band structure calculations involve computing energies
   PRINT_LEVEL LOW                # Level of detail of output
&END GLOBAL
&FORCE_EVAL                      ## Parameters for energy and force calculations
   METHOD Quickstep               # DFT calculations with the Gaussian plane wave method
   &SUBSYS                       ## Define the atomic system
      &CELL                      ## Unit cell parameters
         A [angstrom]  0.000000  2.715475  2.715475
         B [angstrom]  2.715475  0.000000  2.715475
         C [angstrom]  2.715475  2.715475  0.000000
         PERIODIC XYZ             # Periodicity of the unit cell; in most cases, this should be the same as &FORCE_EVAL/&DFT/&POISSON/PERIODIC
      &END CELL
      &KIND Si                ## Define a species of atom (repeated for each atom kind in cell)
         # Selected basis set and pseudopotential from files specified in &FORCE_EVAL/&DFT
         BASIS_SET DZVP-GTH-PBE
         POTENTIAL GTH-PBE-q4
      &END KIND
      &COORD                     ## Coordinates of atoms in primitive cell
         SCALED                   # Relative to primitive lattice vectors
         Si  0.00  0.00  0.00
         Si  0.25  0.25  0.25
      &END COORD
   &END SUBSYS
   &DFT                          ## Parameters for computations using density functional theory
      # Files containing basis set and pseudopotential data (usually located in $CP2K_DATA_DIR)
      BASIS_SET_FILE_NAME BASIS_SET
      POTENTIAL_FILE_NAME POTENTIAL
      &POISSON
         PERIODIC XYZ             # Periodic boundary conditions for electrostatic calculations; should be the same as &FORCE_EVAL/&SUBSYS/&CELL/PERIODIC
      &END POISSON
      &XC                        ## Parameters to configure the exchange-correlation functional
```

```
36          &XC_FUNCTIONAL PBE      # Perdew–Burke–Ernzerhof exchange-correlation functional
37          &END XC_FUNCTIONAL      # This section contains no parameters; default settings for
             ↪  the PBE XC-functional are used
38       &END XC
39       &PRINT                    ## Defines which parts of the DFT computation are printed or
         ↪  saved to file
40          &BAND_STRUCTURE        ## Sample the band structure along a k-path, defined below
41             ADDED_MOS 6          # Number of conduction bands to include
42             FILE_NAME ${PROJECT}.band # File in which to *append* band structure output
                ↪  (note: does not replace or override file. Remove file before recomputing)
43             &KPOINT_SET         ## Define a k-path linearly interpolating between defined
                ↪  k-points
44                UNITS B_VECTOR    # Coordinates are multiples of reciprocal lattice vectors
                   ↪  (1/Å)
45                SPECIAL_POINT L  0.500  0.500  0.500
46                SPECIAL_POINT Γ  0.000  0.000  0.000
47                SPECIAL_POINT X  0.500  0.000  0.500
48                NPOINTS 50        # Number of interpolation points between each special k-point
49             &END KPOINT_SET
50             &KPOINT_SET
51                UNITS B_VECTOR
52                SPECIAL_POINT X  0.500  0.000  0.500
53                SPECIAL_POINT U  0.625  0.250  0.625
54                NPOINTS 20
55             &END KPOINT_SET
56             &KPOINT_SET
57                UNITS B_VECTOR
58                SPECIAL_POINT K  0.375  0.375  0.750
59                SPECIAL_POINT Γ  0.000  0.000  0.000
60                NPOINTS 50
61             &END KPOINT_SET
62          &END BAND_STRUCTURE
63       &END PRINT
64    &END DFT
65 &END FORCE_EVAL
```

B.2 Scripts for Creating Band Structure Diagrams

Two `python3` scripts are provided which assist with

1. (`cp2k-band2csv.py`) parsing band structure output produced by `CP2K`, and
2. (`band-plotter.py`) plotting the band structure in a band diagram using `matplotlib`.

The first script was adapted from `https://github.com/dev-zero/cp2k-tools/blob/develop/scripts/cp2k_bs2csv.py`, and the latter was written for this project.

To use these utilities, ensure that the scripts have execute permissions (`chmod +x *.py` adds execute permissions to all Python scripts in the current working directory) and then run them as follows.

```
$ cp2k -i mono-Si.inp -o mono-Si.out    # creates `mono-Si.band` file
$ ./cp2k-band2csv.py mono-Si.band       # produces a set of .csv files
$ ./band-plotter.py mono-Si.band        # plots an interactive band diagram
```

B.2.1 cp2k-band2csv.py

```
#!/usr/bin/env python3
"""
Convert the CP2K band structure output to CSV files.

This script was adapted by Joseph Wilson under PHYS493, 2019 from:
https://github.com/dev-zero/cp2k-tools/blob/develop/scripts/cp2k_bs2csv.py
It is free to use and modify.
"""

import re
import argparse


SET_MATCH = re.compile(r'''
[ ]*
  SET: [ ]* (?P<setnr>\d+) [ ]*
  TOTAL [ ] POINTS: [ ]* (?P<totalpoints>\d+) [ ]*
  \n
(?P<content>
  [\s\S]*?(?=\n.*?[ ] SET:|$)  # match everything until next 'SET' or EOL
)
''', re.VERBOSE)

SPOINTS_MATCH = re.compile(r'''
[ ]*
  POINT [ ]+ (?P<pointnr>\d+) [ ]+ (?P<label>not[ ]specified|\S+) [ ]+ (?P<a>\S+) [ ]+
 ↪  (?P<b>\S+) [ ]+ (?P<c>\S+)
''', re.VERBOSE)

POINTS_MATCH = re.compile(r'''
[ ]*
  Nr\. [ ]+ (?P<nr>\d+) [ ]+
  Spin [ ]+ (?P<spin>\d+) [ ]+
  K-Point [ ]+ (?P<a>\S+) [ ]+ (?P<b>\S+) [ ]+ (?P<c>\S+) [ ]*
  \n
[ ]* (?P<npoints>\d+) [ ]* \n
(?P<values>
  [\s\S]*?(?=\n.*?[ ] Nr\.|$)  # match everything until next 'Nr.' or EOL
)
''', re.VERBOSE)


def convert_band2csv(file, basename):
    """
    Convert the input from the given input file handle and write
    CSV output files based on the given pattern. Returns a list of
    the special points.
    """
    special_points = []
    for kpoint_set in SET_MATCH.finditer(file.read()):
        filename = f'{basename}.set-%s.csv' % kpoint_set.group('setnr')
        set_content = kpoint_set.group('content')

```

```
53          with open(filename, 'w') as csvout:
54             print(f"writing point set {filename!r}"
55                   " (total number of k-points: {totalpoints})"
56                   .format(**kpoint_set.groupdict()))
57
58             print("  with the following special points:")
59
60             for point in SPOINTS_MATCH.finditer(set_content):
61                point = point.groupdict()
62                if point['label'] == 'not specified':
63                   point['label'] = 'nolabel'
64                print("    {pointnr}) {label}\t < {a}, {b}, {c} >".format(**point))
65                special_points.append(point)
66
67             for point in POINTS_MATCH.finditer(set_content):
68                results = point.groupdict()
69                results['values'] = " ".join(results['values'].split())
70                csvout.write('{a} {b} {c} {values}\n'.format(**results))
71
72       return special_points
73
74
75    if __name__ == '__main__':
76       parser = argparse.ArgumentParser(description=__doc__)
77       parser.add_argument('bandfilename', metavar='bandstructure-file', type=str,
78                           help="the band structure file generated by CP2K")
79
80       parser.add_argument('-k', action='store_true',
81                           help="exports kpath with special point labels to CSV")
82
83       args = parser.parse_args()
84
85       kpath_filename = f'{args.bandfilename}.kpath.csv'
86       with open(args.bandfilename, 'r') as pointset_file:
87          if args.k:
88             # write special k-points to file
89             with open(kpath_filename, 'w') as kpath_file:
90                print(f"writing k-path to {kpath_filename!r}")
91                special_points = convert_band2csv(pointset_file, args.bandfilename)
92                for special_point in special_points:
93                   kpath_file.write("{label} {a} {b} {c}\n".format(**special_point))
94          else:
95             convert_band2csv(pointset_file, args.bandfilename)
```

B.2.2 band-plotter.py

```
1  #!/usr/bin/env python3
2  """
3  Plot a band structure diagram from CSV point sets. See `cp2k-band2csv.py`
4  for how to the generate the CSVs from CP2K output.
5
6
7  This script was written by Joseph Wilson under PHYS493, 2019.
8  It is free to use and modify.
```

```
"""

import argparse
from glob import glob
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages


def get_plottable_data(pointsets, labelled_kpoints={}):
    """
    Convert the CSV data from point sets generated by `cp2k-band2csv.py`
    into an easy-to-plot form, complete with special k-point x-labels.

    Returns
    """

    xlabels = {}
    xcoord = 0
    xcoords = []
    data = []
    for pointset in pointsets:
        for kcoord, bandvals in zip(pointset[:,:3], pointset[:,3:]):

            # find kcoord label
            for labelled_kcoord, label in labelled_kpoints.items():
                # see if kcoord is equal to a labelled kcoord to within floating point precision
                if np.allclose(kcoord, labelled_kcoord):
                    # kcoord is labelled
                    if xcoord in xlabels and xlabels[xcoord] != label:
                        xlabels[xcoord] += '|' + label
                    else:
                        xlabels[xcoord] = label
                    break

            xcoords.append(xcoord)
            data.append(bandvals)

            xcoord += 1
        xcoord -= 1       # have no horizontal gap between point sets

    bands = np.array(data).T
    return bands, xcoords, xlabels


def find_bandgaps(bands):
    """
    Finds the (finite) ranges of energy which contain no bands.
    Uses the fact that the bands are ordered so that their
    successive minima and maxima are non-decreasing.
    """
    gaps = []
    upper = np.inf
    for band in bands:
        lower = band.min()
        if upper < lower:
```

```
65              gaps.append((upper, lower))
66          upper = band.max()
67      return gaps
68
69
70  def save_as_pdf(filename):
71      """
72      Export the figure as it appears to a PDF.
73      """
74      if not filename.lower().endswith('.pdf'):
75          filename += '.pdf'
76      pdf = PdfPages(filename)
77      pdf.savefig(fig)
78      pdf.close()
79      print(f"saved figure to {filename!r}")
80
81
82  if __name__ == '__main__':
83      parser = argparse.ArgumentParser(description=__doc__)
84      parser.add_argument('bandfilename', metavar='bandstructure-file', type=str,
85                          help="the band structure file generated by CP2K. "
86                               "This file isn't read, but used as the basename "
87                               "to find associated csv files.")
88
89      args = parser.parse_args()
90
91      # load pointsets
92      pointset_pattern = f'{args.bandfilename}.set-*.csv'
93      pointsets = [np.loadtxt(f) for f in sorted(glob(pointset_pattern))]
94      if not pointsets:
95          raise parser.error(f"no point sets found: {pointset_pattern}")
96
97      # load kpath file, if it exists
98      try:
99          kpath_filename = f'{args.bandfilename}.kpath.csv'
100         kpath = np.loadtxt(kpath_filename,
101                            dtype=[('label', '<U20'), ('a', float), ('b', float), ('c', float)])
102         labelled_kpoints = {tuple(kpoint): label for label, *kpoint in kpath}
103     except OSError:
104         print(f"kpath file {kpath_filename!r} not found. Plot will have no x-labels")
105         labelled_kpoints = {}
106
107     # convert data
108     bands, xcoords, xlabels = get_plottable_data(pointsets, labelled_kpoints)
109     bandgaps = find_bandgaps(bands)
110
111     if bandgaps:
112         # set zero level to the bottom of the highest band gap
113         highest_bandgap = bandgaps[-1]
114         print(f"highest bandgap: {highest_bandgap[1] - highest_bandgap[0]:5g} eV")
115         yoffset = -highest_bandgap[0]
116     else:
117         print(f"no bandgaps found")
118         yoffset = 0
119
120     # create figure window and axes
```

```
121     fig, ax = plt.subplots()
122
123     # show k-path along x axis
124     sortedxticks = sorted(xlabels.keys())
125     ax.set_xticks(sortedxticks)
126     ax.set_xticklabels([xlabels[x] for x in sortedxticks])
127     for x in sortedxticks:
128        ax.axvline(x, c='k', lw=0.5)
129
130     # plot each band
131     for i, band in enumerate(bands):
132        ax.plot(xcoords, band + yoffset, c=plt.cm.rainbow(i/len(bands)))
133
134     # shade in the bandgap region
135     for lower, upper in bandgaps:
136        ax.axhspan(lower + yoffset, upper + yoffset, alpha=0.2)
137
138     ax.set_title(args.bandfilename)
139     ax.set_ylabel("$E_n(k)$ [eV]")
140     ax.set_xlabel("$k$")
141
142     ax.autoscale(tight=True, axis='x')
143     plt.tight_layout()
144
145     plt.show()
```