Hands-on Workshop
Density-Functional Theory and Beyond: Frontiers of Advanced Electronic Structure and Molecular Dynamics Methods

Beijing, China
July 30 – August 10, 2018

Tutorial II: Periodic Systems
Manuscript for Exercise Problems

Prepared by Björn Bieniek, Johannes Hoja, William Huhn, Franz Knuth, Sebastian Kokott, Björn Lange, Lydia Nemec, Ruyi Song, Victor Yu, and Xunhua Zhao
Fritz-Haber-Institut der Max-Planck-Gesellschaft
Beijing, August 1st, 2018
Introduction

This tutorial aims to familiarize you with the basic concepts of periodic density-functional theory (DFT) calculations and with the settings necessary to run FHI-aims. Before we start working on the first problem, a short overview is provided.

The practice session consists of three parts:

Part I: Basic properties of solids and convergence tests

- Problem I: Generation and visualization of bulk structures
- Problem II: Energy convergence tests
- Problem III: Phase stability and cohesive properties
- Problem IV: Unit cell relaxation
- Problem V: Electronic band structure & density of states

Part II: Basic surface calculations

- Problem VI: Electronic structure of crystal surfaces
- Problem VII: Relaxesing surface structures

Part III: Beyond the Schrödinger equation: spin-orbit coupling

- Problem VIII: The effects of spin-orbit coupling on a free atom
- Problem IX: The effects of spin-orbit coupling on periodic materials

You should work through all the problems of Part I to learn about the basic concepts of periodic systems. Afterwards, you may attempt either Part II, where you will learn about the basic concepts of surface calculations, or Part III, which introduces spin-orbit coupling and its effects on energy levels and band structures.

In the directory `$HandsOn/tutorial_2/`, you can find all the files necessary for this tutorial. Please copy the contents of the `skel/` folder into your own working directory. Dedicated folders have been prepared in the `skel/` directory for each problem. Please use this directory hierarchy, as a few of the directories contain helpful files.

There are two important scripts introduced in the first problem, `run_diamond` and `postprocess`. Both BASH and Python variants have been provided for these scripts, with suffix `.sh` and `.py` respectively. `run_diamond` prepares and starts FHI-aims calculations on a series of geometries for the diamond structural prototype, and `postprocess` post-processes the resulting FHI-aims output. Please try to adapt these scripts to the other problems throughout the rest of this tutorial; in particular, you will need to modify `run_diamond` to work with other structural prototypes introduced in the tutorial.

If you have difficulty with a particular problem, do not hesitate to ask one of the tutors. In any case, it is perfectly fine to skip the rest of a problem and move on to the next. This also applies if your calculation takes significantly longer than the estimated CPU time for the given problem. Any intermediate results required for later problems are provided in the `reference/` folder. If you like, you may also use this folder to compare to your results.

Take your time to read every task carefully before starting calculations. Each subtask starts with a short summary (grey box) and gives details and hints afterwards. Also, feel free to consult the supplementary information presented in the Appendices.
Part I: Basic properties of solids and convergence tests

In the first Part of this tutorial, we will work on different structural phases of bulk silicon. The correct description of bulk silicon’s pressure dependence by Yin and Cohen [1] is one of the early success stories of computational materials science. In this Part, we show how to calculate basic properties of solids such as lattice constants, cohesive energies, band structures, and density of states.

Please use the basic settings given in Fig. 1 as default for Part I of this tutorial (unless specified otherwise).

```
# Physical settings
xc pw-lda
spin none
relativistic atomic_zora scalar

# k-grid settings (to be adjusted)
k_grid nkx nky nkz
```

Figure 1: Default physical and computational settings for `control.in` for Part I. This file can be found in `skel/problem_1/control_part1.in`.

The Perdew-Wang LDA (`xc pw-lda`) exchange-correlation functional will be used for all calculations. The effect of using different xc functionals has been discussed in “Tutorial 1: Basics of Electronic-Structure Theory”. Silicon is known to be non-magnetic, so no explicit spin treatment is needed. The “relativistic atomic_zora scalar” setting is not strictly necessary for silicon, since the nuclear charge of silicon ($Z = 14$) is still small enough to allow for a non-relativistic treatment. But as the correction is computationally inexpensive, it does not hurt to use it, either. However, never compare total energies from different relativistic settings, as they will differ.

The SCF settings section of `control.in` has already been discussed in detail in “Tutorial 1: Basics of Electronic-Structure Theory”. The `k_grid` setting section will be discussed in this tutorial.

For the species settings, please use the default “light” species settings for Si in: `$SPECIES_DEFAULTS/light/14_Si_default`.

Problem I: Generation and visualization of bulk structures

Our first step towards studying periodic systems with FHI-aims is to construct periodic geometries in the FHI-aims geometry input format (`geometry.in`) and visualize them. After that, we will learn how to set basic parameters in `control.in` for periodic calculations. Finally, we will compare total energies of different Si bulk geometries.

Setting up and visualizing `geometry.in`

- Construct `geometry.in` files for the Si `fcc`, `bcc`, and diamond structures. Use the approximate lattice constants $a$ of 3.8Å for `fcc`, 3.1Å for `bcc`, and 5.4Å for the diamond structure.
- Visualize them (e.g. with Jmol).
- Do not perform any calculations yet.

To set up a periodic structure in FHI-aims, all three lattice vectors as well as the atomic positions in the unit cell must be specified. The lattice vectors are specified by the keyword `lattice_vector`. There are two ways to specify the atomic positions. As in the cluster/non-periodic case, you can specify absolute Cartesian positions with the keyword `atom`. Alternatively, you can specify the atomic positions in the basis of the lattice vectors, the `fractional` (or commonly, `direct`) coordinates, with the keyword `atom_frac`. The fractional coordinates $s_i$ are dimensionless and the coefficients for the linear combination of the lattice vectors $a_i$. Written out as a formula, this linear combination reads as follows

$$\mathbf{R} = s_1 \cdot \mathbf{a}_1 + s_2 \cdot \mathbf{a}_2 + s_3 \cdot \mathbf{a}_3,$$

where $\mathbf{R}$ is the Cartesian position of the specified atom.

For example, `fcc` Si with a lattice constant $a = 4$Å is defined by
lattice_vector 0.0 2.0 2.0
lattice_vector 2.0 0.0 2.0
lattice_vector 2.0 2.0 0.0
atom_frac 0.0 0.0 0.0 Si

A full set of lattice vectors and atomic positions of primitive unit cells (the unit cell with the smallest volume, containing the bare minimum number of atoms necessary to replicate the system when all periodic images are included) for fcc, bcc, and diamond can be found in Appendix I. In Fig. 9 in Appendix I the simple cubic and primitive unit cells of fcc, bcc, and diamond are shown. Atomic positions are provided in Cartesian coordinates. Please note that the values in geometry.in must be provided explicitly and not in terms of the formulas presented in Appendix I; specifying a value of 4.0/2.0 instead of 2.0 will cause your calculation to fail.

The simplest way to check the geometry.in file is to visualize the corresponding geometry. This should always be done before any calculation, to verify that the structure is plausible (and that no atoms are extra or missing). For periodic structures in FHI-aims, we recommend Jmol, an open-source Java viewer for chemical structures in 3D. Information on the program and the source code can be obtained from http://jmol.sourceforge.net. To visualize a structure given in geometry.in with Jmol (Fig. 2), please type

jmol geometry.in &

To get periodic images, click with the right mouse button inside the Jmol drawing area and choose “Symmetry” → “Reload {444 666 1}”.

Setting up control.in and running FHI-aims

- Prepare a control.in file using 3×3×3 k-points and the settings given in the introduction of part I (see Fig. 1). These settings can be found in skel/problem_1/control_part1.in.
- Use the provided run_diamond.py script (Fig. 11) to calculate total energies of each of the different phases as a function of lattice constant $a$. For this, consider 7 different values of $a$ in steps 0.1 Å around the lattice constants given above for each structure.

[Estimated total CPU time: 3 min]

The control.in for periodic calculations looks much the same as for the cluster case, as the underlying numerics are the same. There is one important difference, though: a $k$-grid for the Brillouin zone integrations must be specified. For example, to specify a 3×3×3 $k$-grid, the following line must be included in control.in:

k_grid 3 3 3

The $k$-grid points are defined in terms of the reciprocal lattice vectors which are generated from the real-space lattice vectors as defined in geometry.in. The ordering of real-space lattice vectors in geometry.in
determines the ordering of reciprocal lattice vectors in the code. For systems with real-space lattice vectors that are not equivalent by symmetry (such as a surface calculation where one lattice vector is much longer than other lattice vectors), the ordering will matter. You will see that for a small unit cell this \( k \)-point density is never enough. But we can try to find out what happens. In the next problem, the \( k \)-point settings will be discussed in detail.

In this tutorial, we call FHI-aims with the command similar to the one we used in “Tutorial 1: Basics of Electronic-Structure Theory”. However, today we run the calculations in parallel on more than one processor by using MPI (Message Passing Interface). In order to run the calculation on the 4 available (physical) processors\(^1\) in your workstation, use the following command:

```
mpiexec -n 4 aims.x | tee aims.out
```

Make sure you use MPI, otherwise your calculations will take (roughly) 4 times longer. The estimated CPU time given with the exercises refers to the calculations with 4 processors. It is good practice to use a separate directory for every run of FHI-aims in order to preserve the exact input files along with the output files. In this tutorial, however, most of the calculations can be started using a prepared script which takes care of properly setting up calculations, and the script needs to be adjusted only slightly.

In this exercise, we compare energies of different lattice structures as a function of lattice constant. Each calculation can be prepared and started by hand, in principle, but we strongly suggest to use the python script `run_diamond.py`\(^2\) for the post-processing script provided in skel/problem_1/02_3x3x3/diamond/ and shown in Fig. 11 in Appendix III. This example script calculates the total energy of diamond Si with different lattice constants, which have to be set in the beginning of the script. Please copy this script to dedicated folders for bcc and fcc Si (we suggest problem_1/02_3x3x3/bcc/ and problem_1/02_3x3x3/fcc/), and adjust the copies according to the lattice constants given in this subtask and the lattice structures given in Appendix I. You will also need to adjust the list of lattice parameters scanned over by the script, as if you do not, you may not capture the energy minimizing lattice constants.

You will also need to provide a `control.in` you created.

To retrieve the total energies, after running the `run.sh` script, you should then use the `postprocess.py`\(^3\) script, which is provided within the same folder and printed in Fig. 12 in Appendix III. This script extracts the total energies and writes them to the file `energies.dat`, along with the lattice constants. You will need to adapt this script to the other phases of silicon. In particular, adjust the lattice constants. For the next subtask, it is advantageous to write out the total energy per atom, not per unit cell, which makes a difference for the diamond structure. (In the case of fcc and bcc, each unit cell contains one atom, so the total energy per atom and the total energy per unit cell are the same quantity.)

**Plotting total energies**

- Plot the total energy per atom of each structure as a function of the lattice constant (e.g. with xmgrace).
- What is the most stable bulk phase of Si according to your results?

After running the code, plot your data (given in `fcc/energies.dat`, `bcc/energies.dat`, and `diamond/energies.dat`) using for example xmgrace by typing:

```
xmgrace -legend load \
   fcc/energies.dat bcc/energies.dat diamond/energies.dat &
```

All curves should have minima (if not, check the list of lattice parameters in your `run.sh` script) and similar energy ranges (if not, make sure that you properly printed out the total energy per atom, and not the total energy per unit cell).

You should see that, with the current computational settings, the diamond Si phase is unfavorable compared to the other two phases by about 0.1 eV. However, the experimentally most stable phase is

---

1 Your operating system likely reports twice as many processors; this is due to a processor technology known as “hyper-threading” where each physical processor is treated as two “logical” processors. It is recommended that you run calculations with the number of physical processors (here 4).

2 In case you are more familiar with bash, there is the bash script `run_diamond.sh` that has the same functionality. You can find it in the same folder as `run_diamond.py`. If you choose to use the bash scripts, you may need to specify the bash executable explicitly, i.e. use `bash ./run_diamond.sh` to run the calculations.

3 The equivalent bash script `postprocess.sh` is provided. If you choose to use the bash version, you may need to run it via `bash ./postprocess.sh`.

5
the diamond structure. We will show in the next two problems that the too coarse 3×3×3 k-grid is the reason for this disagreement.

**Problem II: Energy convergence tests**

The results of the last problem were not quite physical. As will be shown later, this is because the calculations were not converged. Here, we will explicitly check total energy convergence with respect to the k-grid and to the basis set. In principle, each phase needs to be checked separately. Within this tutorial, however, we split the effort and everyone should only check one phase of their own choice.

**Convergence with k-grid**

- Calculate the total energies for only one of the Si phases of Problem I as a function of the lattice constant for k-grids of 8×8×8, 12×12×12, and 16×16×16. Otherwise, use the same computational settings (control.in) and the same lattice constants as in Problem I.
- Prepare a plot with all total energies drawn against lattice constant using xmgrace, as you did in Problem I. Add the 3×3×3 results from Problem I, too.
- Which k-grid should be used to achieve convergence within 10 meV?

[Estimated total CPU time: 6 min]

You should dedicate one directory for every series of these calculations. You will find empty folders in skel/problem_2/. These calculations should be done exactly as in Problem I but with the appropriate changes to control.in. In particular, use the scripts from Problem I (Fig. 11 and Fig. 12) and control.in for the phase you chose.

In the metallic fcc and bcc phases, the total energy of the 3×3×3 calculation is about 0.3 eV lower than the most accurate (16×16×16) calculation. The larger part of this error is already fixed by the 8×8×8 k-grid, which is still off by about 30 meV. The 12×12×12 grid, on the other hand, is converged within about 5 meV. For the semiconducting diamond Si phase, the total energy of the 3×3×3 calculation is about 0.3 eV too high, but the convergence is already very good for an 8×8×8 k-grid. In general, metals (like Si fcc & bcc) or small cells require a denser k-grid compared to semiconductors (Si diamond) or large cells.

In conclusion, we can use a 12×12×12 k-grid for fcc and bcc Si as a good compromise of high accuracy and reasonable computational time. For simplicity, we use the same grid also for diamond Si although 8×8×8 would be enough in that case.

**Convergence with basis set size**

- Calculate the total energies for your phase of Si as a function of the lattice constant for the minimal and the tier1 basis sets. Use the same lattice constants and computational settings as in Problem I together with the 12×12×12 k-grid.
- Again, prepare a plot with the total energies. Add the results for the minimal+spd basis set (the default for the “light” species settings) from the k-point convergence test above.

[Estimated total CPU time: 4 min]

Use your control.in file from the previous problem for the 12×12×12 k-point grid. In order to change the basis size settings, you should have a look into the species-dependent settings within control.in. There, you will find a line starting with “# First tier - ...”. Each line after this defines a basis function which is added to the minimal basis. Right now, there is one additional function for each valence function (s and p) as well as a d function to allow the atoms to polarize. This is what we call minimal+spd in the context of this tutorial. In quantum chemistry and in particular the Gaussian community, this type of basis set is often called “double zeta (ζ) plus polarization” (DZP).
To run FHI-aims with a minimal basis, simply comment out these three lines by prepending a “#” character. To run FHI-aims with a full tier1 basis set, uncomment all four lines following the statement “# “First tier” - ... ” by removing the initial “#” character.

You can see that the minimal basis gives completely unphysical results; the energetic minimum is strongly shifted towards larger lattice constants. The minimal basis lacks the flexibility to give reasonable geometries. On the other hand, the binding curve does not change drastically from minimal+spd to the full tier1 basis set whereas the computational effort increases significantly by adding the $f$ functions from minimal+spd to full tier1, as there are seven such functions ($l=-3,-2,...,2,3$). You could also check tier2. In that case, though, it would be worth using better integration grids (tighter) etc. as well.

While the total energy difference of about 60 meV between minimal+spd and tier1 is still larger than what we were aiming for in the case of the $k$-grid, we can make use of the fact that the total energy is variational so that a large part of the basis set error actually cancels nicely in energy differences between bounded structures.

**Bonus: Effect of Gaussian broadening of the Kohn-Sham occupation numbers**

**Bonus:** Please skip this subtask if you run out of time or you are inexperienced in modifying scripts.

- Calculate the total energies for fcc Si as a function of the lattice constant for a Gaussian broadening of $\sigma = 0.1$ eV. Use the same lattice constants and computational settings as before with a 12×12×12 $k$-grid and the minimal+spd basis.

- Prepare a plot with the corrected, uncorrected total energies, and the electronic “free energies” for a broadening of $\sigma = 0.1$ eV and the default value of $\sigma = 0.01$ eV from the previous calculations.

[Estimated total CPU time: 2 min]

Use your control.in file and scripts from the previous problem, with a 12×12×12 $k$-point grid and the minimal+spd basis. You can explicitly set the Gaussian broadening to $\sigma = 0.1$ eV by specifying

```
occupation_type gaussian 0.1
```

in control.in.

FHI-aims always outputs three different energies. While these energies are all the same for systems with a gap, they differ for metallic systems with finite Gaussian broadening. The “Total energy uncorrected” gives the value of the Kohn-Sham energy functional for the final self-consistent electronic structure. However, due to the Gaussian broadening, the self-consistency procedure does not minimize this total energy but a “free energy” specified right of “Electronic free energy”. From these two numbers, FHI-aims back-extrapolates to the total energy without broadening and writes the resulting number of “Total energy corrected”. For true metals, it is generally best to make use of the correction. For finite systems and in particular for isolated atoms, however, the back-extrapolation is unphysical and should not be used. You will need to modify your postprocess script to output the values for all three total energies.

For diamond Si, the Gaussian broadening makes no difference at all as long as the broadening $\sigma$ is small compared to the band gap. The first thing to notice for the metallic phases is that all of these numbers agree with each other within about 2 meV. For the default broadening of $\sigma = 0.01$ eV, the energies even agree within 0.1 meV. It can be shown using the variational principle that the total energy always increases and the electronic “free energy” always decreases for finite broadening.

For the following calculations, we will use the default value of $\sigma = 0.01$ eV because there is no benefit in convergence by increasing this parameter for the studied phases of Si. We will stick to the corrected total energy for the periodic systems in this part of the tutorial as it is the most accurate value for metals and makes no difference for semiconductors.

**Problem III: Phase stability and cohesive properties**

Having found the converged computational settings, we now revisit the phase stability of bulk silicon in Problem I. Of course in practice one should always check convergence first to avoid false conclusions as in Problem I. We will learn how to compute the basic cohesive properties and study the pressure dependence of phase stability.
Recalculation of $E(a)$ curves

- Calculate the total energies of fcc, bcc, and diamond Si as a function of lattice constant $a$. Use the settings from Problem II ($k$-grid of $12\times12\times12$, minimal+spd basis) and the same set of lattice constants as in Problem I.
- Plot the results as was done in Problem I.

[Estimated total CPU time: 4 min]

This task can be performed analogously to Problem I, but using the control.in file from Problem 2 for the $12\times12\times12$ $k$-grid and minimal+spd basis. The resulting binding curves clearly show that the experimentally observed diamond structure of silicon is most stable in LDA among the crystal structures studied in this tutorial. In the rest of this exercise, we will analyze the results obtained so far.

Cohesive energies and atomic volumes

- Calculate the total energy of a free Si atom as described in the text below.
- Figure out how to calculate the cohesive energies and the atomic volumes for all FHI-aims runs in the first subtask.
- Plot all cohesive energies of all three phases into one plot with the atomic volume on the $x$ axis.

[Estimated total CPU time: <1 min]

For the single atom energy, special care has to be taken. First, the free Si atom is spin polarized (as its valence p orbitals are not fully occupied), so you should use “spin collinear” instead of “spin none” as well as properly initialize the magnetization. For the free atom, this can be done by specifying “default_initial_moment hund”. You can use the control.in from the previous task only removing the line specifying the k-grid.

Second, we use a more converged basis. In particular, uncomment all basis functions up to and including tier 3, increase the confining potential to “cut_pot 8.3.1.”, and turn off basis dependent confining potentials with “basis_dep_cutoff 0.” in the species section of control.in.\(^4\)

Additionally, use the “Total energy uncorrected” instead of the “Total energy corrected” because the entropy correction is not physical for finite systems and in particular for atoms.

The cohesive energy ($E_{\text{coh}}$) of a crystal is the energy per atom needed to separate it into its constituent neutral atoms. $E_{\text{coh}}$ is defined as

$$E_{\text{coh}} = - \frac{E_{\text{bulk}} - NE_{\text{atom}}}{N} = - \left[ \frac{E_{\text{bulk}}}{N} - E_{\text{atom}} \right],$$

where $E_{\text{bulk}}$ is the bulk total energy per unit cell and $N$ the number of atoms in the unit cell. $E_{\text{atom}}$ is the energy of the isolated atom calculated above.

In order to compare the pressure dependence of phase stabilities, we need to express the lattice constant behavior of all phases on equal footing. One possibility to do so is to express the lattice constant in terms of the volume per atom. This atomic volume can be calculated quite easily from the lattice constant $a$ corresponding to the edge length of the simple cubic (super-)cell. To understand the difference between the primitive unit cells (which you have been using in this tutorial) and the associated simple cubic cells, please see Fig. 9 in Appendix I.

The simple cubic cell has the volume $V_{sc} = a^3$. This number should be divided by the number of atoms $N_{sc}$ in this cell $V_{\text{atom}} = a^3/N_{sc}$. Please verify that there are two, four, and eight atoms in the simple cubic supercell in the case of the bcc, fcc, and the diamond structure, respectively. Fig. 9 in Appendix I may help.

\(^4\) In most other finite-basis approaches, it is common to ensure equivalent basis sets for energy differences. The numerical atomic orbitals of FHI-aims, however, treat the free atom in principle exactly already with a minimal basis, and no such cancellation of errors is needed. We make use of this fact by using one and the same atomic reference for all basis sizes used for the compounds. This way, basis set convergence can be checked more easily as cohesive energies strictly increase with increasing basis size.
In summary, a file `energies.dat` containing the lattice constants and total energies per atom can be converted to a file `cohesive.dat` containing atomic volumes $V_{\text{atom}}$ and (negative) cohesive energies $-E_{\text{coh}}$ by the script `convert-coh.py` given in `skel/problem_3/`. You can use the script by typing:

```
python convert-coh.py --nAtoms .. --eAtom .. -i energies.dat -o cohesive.dat
```

where the “ .. ” need to be replaced by the corresponding values of $N_{\text{sc}}$ and $E_{\text{atom}}$, respectively. Please do not forget to specify the correct path to the script.

After plotting with “xmgrace -legend load */cohesive.dat ”, you see that the diamond structure is indeed the energetically most stable phase. But it is considerably more space consuming. This results from the very open structure of the diamond phase compared to the much closer packing of the bcc and fcc phases. Based on general thermodynamic principles, phases with more compact atomic volumes should be favorable at higher pressures.

**Equations of state and pressure dependence**

- Fit the cohesive energy data for the three phases to the Birch-Murnaghan equation of state using the program `murn.py`.
- Determine the lattice constant $a$, the bulk modulus $B_0$, and the cohesive energy per atom $E_{\text{coh}}$ at equilibrium.
- Compare the above quantities for the diamond phase with the experimental values of $a = 5.430$ Å, $B_0 = 98.8$ GPa, and $E_{\text{coh}} = 4.63$ eV [2].
- Plot the cohesive energies $E(V)$ with respect to the atomic volume for all three phases.
- **Bonus:** Given this data, can you estimate at what pressure a phase transition would occur? Recall the Maxwell construction (see below).

The equilibrium lattice constant $a_0$ is an important quantity which we can calculate from our data. In principle, this can be done with a quadratic ansatz for $E(a)$ or $E(V)$. Here, we will discuss and use a thermodynamically motivated and more accurate fitting function, the Birch-Murnaghan equation of state [3, 4]. The energy per atom ($E = -E_{\text{coh}}$) is expressed as a function of the atomic volume ($V = V_{\text{atom}}$)

$$E(V) = E_0 + \frac{B_0 V}{B'_0} \left[ \left( \frac{V_0}{V} \right)^{B'_0} B'_0 - 1 \right] - \frac{B_0 V_0}{B'_0 - 1}.$$  \hspace{1cm} (3)

The fitting parameters $V_0$ and $E_0$ are the equilibrium atomic volume and atomic energy, respectively, $B_0$ the bulk modulus and $B'_0$ its derivative with respect to pressure. Equation (3) can be derived by assuming a constant pressure derivative $B'_0$.

The fitting program `murn.py` is part of the set of utility scripts distributed with FHI-aims. You can get usage information by typing `$UTILITIES/murn.py --help`. If you have prepared the files `cohesive.dat` using the provided script, you can simply use the script with

```
$UTILITIES/murn.py cohesive.dat -o fit.dat
```

The program then outputs the parameters $V_0$, $E_0$, $B_0$, and $B'_0$ for the given data set as output. As a quick plausibility check of the fit, you can use the option `-p` to see a plot. The script performs no unit conversions, so the bulk modulus $B_0$ is given in units of eV/Å$^3$ because the cohesive energies and atomic volumes were provided in eV and Å$^3$, respectively. You can use “GNU units” to convert to SI units. For example, use

```
units -v "0.5 eV/angstrom^-3" "GPa"
```

to convert 0.5 eV/Å$^3$ to about 80 GPa. The optimal lattice constant can be calculated from the equilibrium atomic volume $V_0$ by $a_0 = \sqrt[3]{N_{\text{sc}}V_0}$ with $N_{\text{sc}}$ the number of atoms in the cubic unit cell.

Be sure to write down the optimal lattice constant estimated by the Birch-Murnaghan fit for the diamond phase of Si, as we will reuse this value throughout the remainder of this tutorial.

---

More precisely, in the Gibbs ensemble $g(p,T)$ which is commonly employed for thermodynamic modeling of solids, the free energy to be minimized is $g = u - TS + pv$, implying a phase with a higher volume per atom $v$ will be more strongly affected energetically by the pressure $p$ (and thus less favorable) than a phase with a lower volume per atom.
Compare the calculated results with experimental reference values given above. Note: Exact agreement between DFT and experimental data is not our goal right here – DFT-LDA is an approximation, and here we see how well (or not) it works. It is well known that LDA in general gives only slightly overbound lattice constants and cohesive energies.

After performing the Birch-Murnaghan fit for all three phases, please plot the resulting fitted curves saved in fit.dat into one figure. You should get something similar to Fig. 1 in the paper by Yin and Cohen [1].

By exposing the crystal to the right pressure (positive or negative), one can tune the atomic volume to values smaller or larger than the equilibrium value. In principle, the most stable phase for a given atomic volume $V$ can simply be deduced from the curve with the lowest $E(V)$. The corresponding pressure can be calculated from the slope of the curve by the simple thermodynamic relation $p = -\partial E/\partial V$.

However, in the regime of about $13 \text{Å}^3 < V_{\text{atom}} < 18 \text{Å}^3$ coexistence of a diamond phase at $\sim 18 \text{Å}^3$ and a $bcc$ phase at $\sim 13 \text{Å}^3$ is favorable (see Fig. 3). The fraction of atoms in the two phases then determines the average atomic volume. The resulting average atomic energy is a straight line between the corresponding points which is below both the lines of diamond and $bcc$ Si. This is called the Maxwell construction. From the slope of this line (a common tangent), we can deduce a transition pressure of about 14 GPa at which diamond and $bcc$ Si could coexist according to our calculations. This is more than $10^5$ times the ambient pressure of about 100 kPa. Note that there are additional phases for silicon which have not been calculated here. In fact, the Si $\beta$-tin phase is more stable than the $bcc$ phase at high pressures. For a more thorough discussion, please refer to [1].

Problem IV: Unit cell relaxation

You have seen in “Tutorial 1: Basics of Electronic-Structure Theory” how to optimize the positions of atoms in cluster systems. To obtain optimized periodic structures, both the atomic positions and the lattice vectors must be optimized. This will be the topic of this problem.

- Fully relax (both atomic coordinates and lattice vectors) a distorted $bcc$ Si structure (see below) with the computational settings you have used before for the Si crystals.
- Analyze the resulting structure (lengths of lattice vectors, angles between them, and atomic positions) and compare it to the equilibrium values of diamond Si obtained from the Birch-Murnaghan fit in the previous problem.

[Estimated total CPU time: 10 min]
The basic idea is to perform a structure relaxation from *bcc* to diamond Si. When performing a local structure optimization, we try not to go uphill in the energy landscape or cross energy barriers. Thus, there should be a path from the starting structure to the desired local minimum with very few barriers in between. However, this means that we have no assurance that we have found the global minimum, as this may lie over a notable energy barrier from our starting geometry. In order to cross such energy barriers, one can do, for example, variable cell-shape molecular dynamics, but this is out of the scope of this tutorial.

Since we desire a diamond structure, we must start with at least 2 atoms, as it is crucial that the starting unit cell be compatible with the primitive cell (i.e., contain integer multiples of the number of atoms in the primitive cell). Hence, our starting unit cell is a cubic *bcc* structure consisting of 2 atoms. Of course, we cannot start with the ideal cubic *bcc* structure because we will be stuck in the local minimum for this highly-symmetric structure. Even if we distort the starting geometry significantly (atomic positions & lattice vectors), we can easily end up in an unwanted local minimum. Therefore, we will provide a suitable starting geometry, but feel free to try other initial geometries when you are finished with the given one. You can find this geometry in the directory *skel/problem_4*/:

```plaintext
lattice_vector 3.1 0.4 0.4
lattice_vector 0.45 3.1 0.4
lattice_vector 0.45 0.45 3.1
atom_frac 0.0 0.0 0.0 Si
atom_frac 0.3 0.3 0.3 Si
```

To perform a full optimization of the crystal structure, add three lines to your *control.in* from the previous problem (k-grid 12×12×12, minimum+spd basis set). The line

```plaintext
relax_geometry trm 1E-2
```

requests a structure relaxation for atoms until residual forces on the atoms smaller than $10^{-2}$ eV/Å have been achieved, and the line

```plaintext
relax_unit_cell full
```

enables full optimization of the lattice vectors (lattice vector lengths and the angles between them).

The structure optimization with the provided starting geometry should take 14 relaxation steps, each relaxation step containing an SCF cycle for the geometry predicted for that step. If you take a closer look into the FHI-aims output, you will see that not only are the atomic forces calculated but also a quantity called the “stress tensor”. In essence, the stress tensor is a measure of the forces acting on the unit cell itself. The final relaxed geometry will be written to *geometry.in.next_step*.

You can visualize the geometries along the relaxation path if you run the script `$UTILITIES/create_geometry_zip.pl` (one of the utilities distributed with FHI-aims) and specify the main FHI-aims output file for your relaxation as an argument for this script. Unzip the resulting file *geometries.zip* if you want to see the individual geometries, or use Jmol and the files generated by `$UTILITIES/create_geometry_zip.pl` to watch a short movie of the relaxation by typing the command “ `jmol -s geometries.spt` ”.

The resulting geometry is a primitive diamond structure. The angles between the lattice vectors are about 60°, and the length of the lattice vectors matches approximately the result from the Birch-Murnaghan fit in the previous problem. The vector connecting the two atoms is $(0.25, 0.25, 0.25)$ in fractional coordinates, as is expected for the diamond structure. Increasing the basis set size also improves the resulting geometry\(^6\). Ideally, one should start a relaxation calculation with FHI-aims using “light” settings. Once the light relaxation has converged, the resulting geometry (e.g., from the file *geometry.in.next_step*) is used as an input for a calculation with “tighter” settings. This saves a significant amount of time, as the relaxation with (relatively) inexpensive “light” settings will draw the system into the neighborhood of the local minimum, allowing the tighter relaxation to focus on “finishing the job” of detailed convergence of the geometry.

**Problem V: Electronic band structure & density of states**

The electronic band structure describes energy levels of electrons inside a solid. It gives information about ranges of energy that electrons can occupy (the bands) and the ranges of energies where no electrons may be found (the band gaps). Many properties of a solid can be deduced from its band structure, e.g., if it is a metal, semiconductor, or insulator.

\(^6\) Comparison to the Birch-Murnaghan fit would require that the calculations entering into the Birch-Murnaghan fit are re-calculated with the same basis set.
• Calculate the electronic structure of diamond Si using the equilibrium geometry found in Problem III.

• Calculate the band structure along the high symmetry lines \( L \rightarrow \Gamma \rightarrow X \rightarrow W \rightarrow K \).

• Calculate the density of states (DOS) using an energy range of \(-18 \text{ eV} \) to \(0 \text{ eV}\), Gaussian broadening of \(0.1 \text{ eV}\), a \( k \)-grid of \(12 \times 12 \times 12\), and dos_kgrid_factors of 5 for each \( k \)-grid direction.

[Estimated total CPU time: 3 min]

Use your control.in and geometry.in for the equilibrium lattice parameter for diamond Si from Problem III. To calculate the band structure, the high symmetry \( k \)-points in reciprocal space must be supplied to FHI-aims. An example excerpt from control.in corresponding to the first part of the suggested path, with 21 points per path\(^7\), is:

```plaintext
# diamond band structure:
output band 0.5 0.5 0.5 0.0 0.0 0.0 21 L Gamma
output band 0.0 0.0 0.0 0.0 0.5 0.5 21 Gamma X
output band ...
```

In each line, the first three numbers are the coordinates of the starting point in units of the reciprocal lattice vectors. The next three numbers specify the end point. The band structure is then calculated along the path connecting these two points.

The last two entries are not mandatory for the calculation inside FHI-aims, but they provide the label of the specified \( k \)-points for the plotting script which you are going to use after the calculation. Please refer to Appendix II for the location of the other high symmetry \( k \)-points in the Brillouin zone for the materials in this tutorial.

Despite the ubiquity of band structure calculations in condensed matter physics and materials science, a systematic list of high-symmetry \( k \)-paths for all possible Brillouin zones (i.e. all possible materials) was proposed only recently in 2010 by Curtarolo and Setyawan in Ref. [5] for usage in the AFLOW materials database (website http://aflowlib.org). For comparison, the original Bloch theorem underlying the band structure formalism was proposed in 1928.

The density of states (DOS)

The density of states is one of the basic concepts in solid state physics. The DOS around the Fermi level is of particular interest as it is one of the fundamental quantities for a material, specifying whether a material is conducting, semiconducting, or insulating. Many material properties depend on the DOS, notably the electrical and thermal conductivity.

The number of states \( n \) within a given energy interval \((\epsilon_0 - \Delta \epsilon) < \epsilon < (\epsilon_0 + \Delta \epsilon)\) per unit volume \( V_{\text{cell}} \) is given by

\[
n = \int_{\epsilon_0-\Delta \epsilon}^{\epsilon_0+\Delta \epsilon} d\epsilon \ g(\epsilon)
\]  

(4)

where \( g(\epsilon) \) is the density of states (DOS). In a free atom or an isolated molecule, the DOS consists of a series of discrete energy levels (\( \delta \) peaks) and can be written as

\[
g(\epsilon) = \sum_i \delta(\epsilon_i - \epsilon).
\]  

(5)

In a periodic system, the single particle energies become \( k \)-dependent and the DOS continuous. The number of states per energy is averaged over \( k \)

\[
g(\epsilon) = \frac{1}{V_{\text{BZ}}} \sum_i \int_{BZ} d^3k \ \delta(\epsilon_i,k - \epsilon).
\]  

(6)

In order to calculate the density of states numerically, we have to replace the integral over the Brillouin zone (BZ) in Eq. (6) by a sum over \( k \)-points. In the case of infinite \( k \)-points, this replacement is exact.

\(^7\) We choose 21 points per path, instead of 20, as the endpoints are included in the number of \( k \)-points. This value breaks the \( k \)-path into 20 evenly spaced intervals.
However, to compensate the deficiency of a finite grid, we broaden the $\delta(\epsilon_{k,i} - \epsilon)$ distribution by a Gaussian function with an broadening factor $\sigma$,

$$g(\epsilon) = \frac{1}{\sqrt{2\pi\sigma}} \frac{1}{n_k} \sum_i \sum_k \exp \left[ -\frac{1}{2} \left( \frac{\epsilon - \epsilon_{k,i}}{\sigma} \right)^2 \right].$$  \hspace{1cm} (7)

To output the DOS in FHI-aims, the following lines are added to control.in:

```
output dos -18. 0. 200 0.1
```

The first two values define the energy window where the DOS should be calculated: the first value is the lower energy bound and the second value is the upper energy bound. The third value is an integer specifying how many points to include in the energy window, and the last value is the Gaussian broadening $\sigma$ used in Equation 7. All energies (both bounds and broadening) are given in eV.

Small changes in the shape of a peak in the DOS will negligibly affect the total energy. Therefore, a rather coarse $k$-grid (defined by the keyword `k_grid`) in combination with rather broad choices of $\sigma$ (given by “occupation_type gaussian”) can be used for the SCF cycle. However, to resolve fine features in the DOS, a denser $k$-grid to include more terms in the the sum over $k$-points in Eq. (7) is necessary. After self-consistency is reached, the DOS can be computed using an interpolated $k$-grid which is made denser by factors $n_1$, $n_2$, $n_3$, respectively. The factors $n_1$ to $n_3$ are given in control.in with the keyword:

```
dos_kgrid_factors 5 5 5
```

The density of states is calculated on a denser grid after the SCF cycle. The dimensions of the new $k$-point grid are $k_1 \times n_1$, $k_2 \times n_2$, $k_3 \times n_3$, where $k_i$ are dimensions of the old $k$-point grid.

Note that two DOS’s will be output into separate files: KS_DOS_total_raw.dat, where the zero of energy has not been altered, and KS_DOS_total.dat, where the zero of energy has been shifted to the computed Fermi level.

**Visualization of band structure and DOS**

- Use the python script `aimsplot.py` to visualize the band structure data and the DOS.
  How large is the LDA band gap?

In order to visualize the band structure, some postprocessing is needed after the FHI-aims run. This may be done by the script `aimsplot.py` (a utility distributed with FHI-aims) as long as the output, `geometry.in`, and `control.in` files are in the same directory. Simply run `$UTILITIES/aimsplot.py` without any arguments in this directory.

You see a band structure with an indirect band gap of about 0.5 eV. Please note that the energy zero is at the Fermi energy calculated by the code (which for an insulating system may lie anywhere within the band gap) and not at the valence band maximum, i.e. the highest occupied energy level for an insulating material\(^8\). The resulting LDA band gap is much smaller than the experimental band gap of silicon (1.17 eV \([2]\)). This disagreement is commonly called the “band gap problem” of (semi-)local functions.

---

\(^8\) The valence band maximum is chosen to be the zero of energy by convention, as it is the “true” Fermi level at $T=0$ K due to the valence band maximum being the chemical potential of electrons in the system.
Part II: Basic surface calculations

Figure 4: The hydrogen saturated ideal 2×1 Si(001) surface. The cyan (light) atoms correspond to the bulk Si atoms, the blue (dark) atoms are the surface Si atoms. The white atoms on the bottom layer are hydrogen atoms.

In the second part of this tutorial, we demonstrate standard techniques used to describe surfaces. We will model the clean Si(001) surface, which is one of the most technologically relevant surfaces. The surface reconstruction at low temperatures (driven by dangling silicon bonds at the surface) was unclear for many years. After much debate in the literature arising from differing models used to describe different feature observed in experimental (LEED, STM) and theoretical approaches, it has been shown by direct evidence in STM by Wolkow [6] that the main surface feature at low temperatures is an asymmetric Si dimer in a 2×1 reconstructed surface unit cell.

Please use the settings given in Fig. 5 of Part II of this tutorial together with the “light” species defaults for silicon. Most of these settings were also used in Part I of this tutorial, thus you can take your control.in from Part I, making sure to adjust k-grid settings. Saturating the dangling Si bonds on the bottom layer of the slab will require the addition of hydrogen to the system, so we additionally need to add the “light” species defaults for hydrogen, which you can find in $SPECIES_DEFAULTS/light/01_H_default, to our control.in file. As may be observed in Fig. 4, each silicon atom is to be saturated by two hydrogen atoms.

<table>
<thead>
<tr>
<th>Physical settings</th>
<th>xc</th>
<th>pw-lda</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>spin</td>
<td>none</td>
</tr>
<tr>
<td>relativistic</td>
<td>atomic_zora scalar</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k-grid settings</th>
<th>k_grid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12  24  1</td>
</tr>
</tbody>
</table>

Figure 5: Physical and computational settings for control.in for part II. This file can be found in skel/problem_6/control_part2.in.
Problem VI: Electronic structure of crystal surfaces

Creation and visualization of geometry.in

- Use the provided Python script (Appendix III) to construct an ideal bulk-truncated diamond Si(100) surface slab in a 2×1 cell consisting of four Si layers and an additional layer of H atoms to saturate the bottom layer. Use sufficient vacuum between the slabs.
- Adjust the Python script to add four additional layers of Si atoms.
- Visualize the surface slabs.
- Do not perform any calculations yet.

The geometry of four layers of Si(001)-(1×1) in diamond structure is given in Appendix I. Please note that by convention the surface is rotated with respect to the bulk structure by 45° around the z axis. The next four layers are just a repetition of these layers shifted by a in z direction. The (2×1) reconstructed surface is constructed by (again) repeating the atomic coordinates, this time shifting the atomic coordinate by a/\sqrt{2} in x direction and doubling the corresponding lattice vector. Please use the optimized lattice constant for diamond Si obtained in Part I.

A slab has two surfaces (by definition), a top and a bottom surface (see Fig. 4). However, in most cases the system of physical interest is essentially an isolated surface, that is, there are enough “bulk” layers between the two surfaces of the slab that they are essentially isolated from one another and are independent experimentally. Here we are interested in the electronic properties of the top surface and would like to minimize the impact of the bottom surface on our calculation. To prevent physical states on the bottom surface (arising from the dangling Si bonds) from appearing in the fundamental gap, the bottom silicon surface is saturated with hydrogen atoms in such a way as to mimic the bonding characteristics of bulk silicon. Each silicon atom in the bottom surface needs two hydrogen atoms placed at a distance of 1.5 Å in the direction where the next set of silicon atoms would have been in the bulk geometry. One can argue that the atomic environment of the hydrogen-saturated silicon atoms will then mimic the atomic environment of bulk silicon.

For your convenience, we provide a simple Python script write-geom.py (the source code is given in Fig. 13 in Appendix II) for the ideal hydrogen-saturated four layer slab. The script is provided in skel/problem_6/01_ideal_2x1_4_layers/. You should only adjust the lattice constant in line 14. The script is executed by typing “./write-geom.py > geometry.in”, and the generated structure data is written to the file geometry.in.

The eight-layer slab can be created either by hand-editing the resulting geometry.in or by editing the Python script. You can edit the Python script without being an experienced Python user. If you read through the code, either on your screen or in Appendix III, you will find in line 17 the variable n_layer. This variable gives the number of layers. In line 28, it is used to determine the slab thickness. You will need to change the value from “4” to “8” to ensure you use the same vacuum thickness as you used before.

As a next step, you will add the atoms of the next four layers. For every layer specify two atoms. This is done by using the output_atom command in the Python script, just like in line 39. The easiest way is to copy lines 39–47 and change the number of the layer in the third argument. Run the script and visualize the resulting geometry.in file.

One will need a sufficient amount of vacuum between surfaces so that the surfaces do not interact through the vacuum. In many DFT codes (especially codes implementing plane-wave basis sets), you would need to run several calculations with different vacuum thicknesses to find the smallest value which gives physical results. This is because, for many DFT codes, increasing the size of the computational cell substantially increases the basis set size and thus the runtime of a calculation, even if no additional atoms are added to the system. However, in codes implementing localized basis functions (such as FHI-aims), there is negligible computational cost in adding additional vacuum to the system9, so you may easily use an relatively large vacuum thickness of \( L_{\text{vac}} = 30 \) Å or more from the onset without a noticeable performance impact. This value is already given in the script by the variable \( L_{\text{vac}} \) in line 15.

9 In fact, this is one of the main strengths of localized basis functions.
Band structure and DOS calculation

- Prepare a control.in according to the specified settings.
- Calculate the density of states (DOS) and the surface band structures for 4 and 8 layer slab along $\overline{\Gamma} \rightarrow \overline{J} \rightarrow \overline{K}$ (see Appendix II).

[Estimated total CPU time: 3+13min]

There are two important issues to note for the optimal $k$-grid for this system. First, there should be no interaction between different periodic images of the slab in $z$ direction. Therefore, only one $k$-point is needed on that axis. (If you did need more than one $k$-point along that axis to yield converged results, this would imply that periodic images in that direction were interacting.) Second, the lattice vector in $x$ direction is twice as large as the lattice vector in $y$ direction. As this gives a shorter periodicity in $k_y$ direction, the number of $k$-points in the first direction can be half of that in the second direction. We use a well converged $k$-grid of $12 \times 24 \times 1$ (see Fig. 6).

Similar to Part I, for calculating the band structure and DOS, add the following lines in the control.in file:

```
# Si 2x1 surface band structure:
output band 0.0 0.0 0.0 0.5 0.0 0.0 21 Gamma J
output band 0.5 0.0 0.0 0.5 0.5 0.0 21 J K
```

and for the DOS

```
output dos -18. 0. 200 0.1
dos_kgrid_factors 5 5 1
```

To visualize the band structure, use the script $UTILITIES/aimsplot.py$ as was done in Problem V. Simply run the script in the directory that contains the input and output files of FHI-aims.

When plotting the band structures, note the differences between the four- and the eight-layer slabs. The bands lying within the (bulk) fundamental band gap hardly change with the number of layers. These bands are the surface bands localized to the surface, which does not noticeably change as the number of layers increases. On the other hand, the valence and conduction bands become roughly twice as dense. These bands are bulk-like; by adding two as many layers, you have increased the number of these bulk-like bands by a factor of two.
Problem VII: Relaxing surface structures

- Create a geometry.in with the two top Si atoms of the four-layer geometry in Problem VI perturbed.
- Set up a calculation to perform a structure relaxation of the top layer of the perturbed geometry.
- Specify a calculation of the DOS and the band structure along $\bar{\Gamma} \rightarrow \bar{J} \rightarrow \bar{K}$ in your control.in.
- Run the structure relaxation with the above settings.
- Compare this band structure to the ones obtained in the last Problem.

[Estimated total CPU time: $\sim 21$ min]

The DOS and the band structure can be calculated by specifying the same lines in the control.in as in the task before.

In order to perform structure relaxation such that the residual forces on atoms are smaller than $10^{-2}$ eV/Å, add the following line to control.in:

```
relax_geometry trm 1E-2
```

During a structure optimization, commonly only the surface-like parts of the structure are allowed to relax, and the bulk-like parts are kept fixed. This is done by the keyword `constrain_relaxation`, which fixes the position of the previously specified atom in geometry.in. In the FHI-aims manual, you may find different options, but for constraining all coordinates of an atom, use the flag “.true.” In the geometry.in file, please write the following line right under the atom which should be kept fixed:

```
constrain_relaxation .true.
```

An example excerpt from geometry.in reads like this:

```
atom -1.2063524529754976 0.0000000000000000 -0.8530200000000001 H
constrain_relaxation .true.
atom 1.2063524529754976 0.0000000000000000 -0.8530200000000001 H
constrain_relaxation .true.
atom ... 
```

For the sake of time, here constrain all atoms except the top layer (i.e. top two Si atoms). For publication-quality calculations, more surface layers should be relaxed. A calculation where the top three layers have been relaxed may be found in the ref/ folder for this Tutorial.

Next, the two top Si atoms must be perturbed (i.e. repositioned) in the xy-plane. For this system, a perturbation on the order of $0.5 \text{ Å}$ is suitable to ensure that the perturbed geometry is sufficiently far away from the symmetry-induced saddle point of the ideal geometry to avoid the structure re-relaxing back into the locally-optimal geometry. Also, make sure to clearly break the mirror symmetry between these two silicon atoms, e.g., by moving one of the two atoms down by several tenths of an Å. A possible choice for the top silicon atoms is:

```
atom 2.4148451634531707 0.0000000000000000 3.262 Si
atom 5.2445354903595121 0.0000000000000000 4.062 Si
```

Here, the two top Si atoms have been moved in the x-direction by $\pm 0.5 \text{ Å}$, and one Si atom has been pushed down by 0.8 Å.

The structure optimization with this starting geometry will take about 10 iterations, but feel free to try out your own guess. Depending on the starting guess, the structure optimization will take between 10 and 30 iterations and therefore 7 to 21 minutes. As in the previous Part, you can visualize geometries along the relaxation path with the script `$UTILITIES/create_geometry_zip.pl$. Unzip the resulting file geometries.zip if you want to see the individual geometries or, as before, use Jmol by typing “`jmol -s geometries.spt`”.

You can achieve the same result by replacing in `write-geom.py` the expression `output_atom(...)` with `output_constrained_atom(...)`. 

10 You can achieve the same result by replacing in `write-geom.py` the expression `output_atom(...)` with `output_constrained_atom(...).`
You will see that one dangling bond for each top silicon atom is saturated by each silicon atom forming a dimer with another silicon atom. Additionally, this dimer is asymmetric, as experimentally evidenced by Wolkow [6].

If you compare the band structures of the reconstructed and clean Si(001) surface around the Fermi level (i.e. compare the results of this Problem to the previous Problem), you find that the formation of the asymmetric dimer cause the reconstructed surface to become semiconducting, and there is a small but clear gap between the valence and the conduction band.
Part III: Beyond the Schrödinger equation: spin-orbit coupling

Relativity, the theory that ensures physics is correct for moving objects up to the speed of light $c$, creates another important set of modifications to a material’s band structure. Relativity enters into electronic structure theory via the attractive Coulomb interaction $-Z/r$ between an atom and an electron. By the simple freshman physics relationship $E = T + V$, we find that the kinetic energy $T$ of an electron as it approaches an atom must diverge as $T \approx E + Z/r$, that is, the electron becomes relativistic.

There are two relativistic effects which commonly enter into density-functional calculations\(^\text{11}\). We will present a semi-classical justification of these two relativistic effects here, as the exact mathematical derivation involves quite a few steps. Interested readers may refer to Ref. [7], where a derivation of spin-orbit coupling starting from the Dirac equation is presented, as well as the implementation of spin-orbit coupling in FHI-aims.

The first relativistic effect commonly used in DFT calculations is scalar relativity. Scalar relativity corrects the classical kinetic energy $mv^2/2$ (which predicts incorrectly that the velocity $v$ of an electron should go to infinity as its kinetic energy increases) by a “mass renormalization” factor. This renormalization factor properly limits the velocity of an electron to not exceed $c$ as the kinetic energy increases. The scalar-relativistic kinetic energy operator has the form

$$\hat{t} = \hat{p} \frac{c^2}{2} + \epsilon \hat{\sigma} \cdot \hat{p},$$

but this equation is rarely solved exactly due to an explicit dependence on the eigenvalue $\epsilon$ of the electron, one of the quantities we wish to solve for. It is an approximation for this relativistic correction that you have been including in your calculations via the relativistic atomic\_zora scalar keyword.

The second relativistic effect widely used in DFT calculations is spin-orbit coupling. Spin-orbit coupling arises from the fact that an electron has a spin $s$ and is moving relative to the nuclei of the system. While we commonly model materials as fixed (or slowly moving) nuclei alongside mobile electrons, by the principles of relativity we may also view the electron as being fixed and the nuclei moving rapidly around the electron. By the principles of electromagnetics, an accelerating charged particle induces a magnetic field $B$, and thus the electron should experience a Zeeman effect $-s \cdot B$ induced by the movement of the nuclei it observes. The spin-orbit coupling operator has the form

$$\hat{v}_{SOC} = -\frac{1}{i4c^2} \hat{\sigma} \cdot \hat{p} \times \hat{p},$$

where the Pauli matrices $\hat{\sigma}$ measure the spin of the electron. One important consequence of spin-orbit coupling is that, as its Zeeman-like form suggests, it will split individual bands of degenerate states into a set of two sub-bands in a phenomena known as “spin-orbit splitting”.

Spin-orbit coupling may be enabled in FHI-aims via the keyword

```
include_spin_orbit non_self_consistent
```

The effects of spin-orbit coupling are most visible in effective single-electron eigenvalues, that is, the band structure of Kohn-Sham theory. While relativistic effects are relatively small for light-element materials, they are noticeable already for GaAs ($Z = 31$ and $Z = 33$, respectively). The main intent of this Part is to point out that spin-orbit coupling is indeed a relevant effect that should be included in band structure calculations of real materials for all but the lightest materials.

Problem VIII: The effects of spin-orbit coupling on a free atom

- Perform a spin-orbit-coupled calculation for a free Hg atom.
- Examine the atomic spectrum to learn about the effects of spin-orbit coupling.

[Estimated total CPU time: $<1\text{ min}$]

As mentioned in the introduction to this section, the effects of spin-orbit coupling are strongest near the nuclei in a material. Thus, to understand the effects of spin-orbit coupling, it is instructive to

\(^{11}\) There are many more relativistic corrections predicted by quantum electrodynamics to higher orders in $1/c$, but these higher-order corrections have negligible effect for the majority of condensed matter physics and materials science.
Writing SOC-perturbed Kohn-Sham eigenvalues.

K-point: 1 at 0.000000 0.000000 0.000000 (in units of recip. lattice)

<table>
<thead>
<tr>
<th>State</th>
<th>Occupation</th>
<th>Unperturbed Eigenvalue [eV]</th>
<th>Eigenvalue [eV]</th>
<th>Level Spacing [eV]</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-11.106878</td>
<td>46.396812</td>
</tr>
<tr>
<td>70</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-11.106878</td>
<td>0.000000</td>
</tr>
<tr>
<td>71</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-11.106878</td>
<td>0.000000</td>
</tr>
<tr>
<td>72</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-11.106878</td>
<td>0.000000</td>
</tr>
<tr>
<td>73</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>1.884246</td>
</tr>
<tr>
<td>74</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>0.000000</td>
</tr>
<tr>
<td>75</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>0.000000</td>
</tr>
<tr>
<td>76</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>0.000000</td>
</tr>
<tr>
<td>77</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>0.000000</td>
</tr>
<tr>
<td>78</td>
<td>1.00000</td>
<td>-10.016049</td>
<td>-9.222631</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

**Figure 7:** An example set of output lines for spin-orbit coupling in FHI-aims. Here, the energy levels of 5d orbital for the free Hg atom without and with spin-orbit coupling are shown.

temporarily ignore the title of this Tutorial and return to the decidedly non-periodic example of a free atom. We here consider the Hg atom, as it is one of the “fruit fly” calculations of relativistic phenomena.

We proceed in the same manner as “Tutorial 1: Basics of Electronic-Structure Theory”, only this time including the line

```plaintext
include_spin_orbit
```

in control.in. A template control.in file is provided in your skel/problem_8/ folder.

After running the calculation, scroll down in the main FHI-aims output, past the SCF cycle, to the section that looks similar to Figure 7. This section outputs the eigenvalues without and with spin-orbit coupling for non-periodic systems. The second column is the occupation number, the third column is the eigenvalue before spin-orbit coupling is applied (i.e. the scalar-relativistic/unperturbed eigenvalue), the fourth column is the eigenvalue after spin-orbit coupling is applied (i.e. the spin-orbit-coupled/perturbed eigenvalue), and the fifth column is the difference between the perturbed eigenvalue and the previous perturbed eigenvalue (the level spacing). Spin-orbit splittings occur when the scalar-relativistic/unperturbed eigenvalues form a single band, but the associated spin-orbit-coupled/perturbed eigenvalues split into two bands; Figure 7 shows a calculated spin-orbit splitting of 1.88 eV for the Hg 5d orbital.

By looking at the multiplicity of unperturbed eigenvalues, one can identify the angular momentum values associated with the states: s states are 2-fold degenerate, p states are 6-fold degenerate, and so on. There are three trends that may be observed in the spin-orbit-coupled eigenvalues. First, we see that the s states are unaffected by spin-orbit coupling; their eigenvalues before and after spin-orbit coupling are the same. Second, we see that the effects of spin-orbit coupling becomes larger in magnitude for states with lower eigenvalues. Third, we see that p states are more strongly modified by SOC than the d states, and f states are weakly modified, even less so than d states. For a discussion of the physical origins of these observed trends, please see Ref. [7].

**Problem IX: The effects of spin-orbit coupling on periodic materials**

- Calculate the band structures, without and with spin-orbit coupling, for Si, GaAs, and CdTe.
- Record the spin-orbit splitting for the valence band maximum in Si, GaAs, and CdTe and observe how they vary as the atomic numbers of constituent species increase.

**[Estimated total CPU time: 2 min (Bonus: 3 min)]**

Returning to the periodic case, we will next observe the effects of spin-orbit orbiting on band structures. We have selected three materials with similar structural prototypes: diamond Si and (cubic) zincblende GaAs and CdTe. These materials are all tetrahedrally-bonded compound semiconductors and are most commonly used materials for semiconducting applications.

You have been provided control.in and geometry.in files for these structures in your skel/problem_9/ folder. The k-point path for the band structure calculations from Problem V is used, but we have
Figure 8: A close up of the prototypical band structure for a compound semiconductor near the valence band maximum, with and without spin-orbit coupling.

increased its density to obtain a smoother figure. For the sake of time, we provide geometry.in files with experimental lattice constants, and we will use a 12×12×12 k-grid and “light” numerical settings without checking for convergence.

FHI-aims will output both the spin-orbit-coupled and the scalar-relativistic (i.e. non-spin-orbit-coupled) band structures. We have provided two scripts to allow for easy comparison between these band structures. The first script, Separate_SR_and_SOC.sh, separates the scalar-relativistic and spin-orbit-coupled band structures into sub-directories. The second script, aimsplot_wrapper.sh, plots both band structures simultaneously for easy comparison. After running FHI-aims for each material, run the Separate_SR_and_SOC.sh script in each material’s directory, followed by aimsplot_wrapper.sh.

All three materials share the same zincblende-derived structure, and accordingly the behavior of the valence and conduction bands are similar near the Γ point. An example is reproduced in Fig. 8 with and without spin-orbit coupling. In particular, note that the valence band at the Γ point splits into two subbands when spin-orbit coupling is applied. This behavior is identical to the spin-orbit splittings observed for the free Hg atom in the previous Problem.

For each material, record the value of the spin-orbit splitting for the valence band maximum in Tab. 1 below. Note that you may click on the magnifying glass in the lower-left-hand corner of the band structure plotting window and drag out a section of the band structure to enlarge it. Also include the largest atomic number of the constituent species in the materials in this table.

How does spin-orbit coupling modify the band structures? In particular, note how the curvature of bands can change when spin-orbit coupling is applied. Physically, this corresponds to a noticeable change in the “effective masses” of charge carriers. When does spin-orbit coupling become important to model the qualitative features of a band structure?

<table>
<thead>
<tr>
<th>Structure</th>
<th>Z_{\text{max}}</th>
<th>Spin-Orbit Splitting [eV]</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>6</td>
<td>0.014</td>
</tr>
<tr>
<td>Si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GaAs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CdTe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12 The Separate_SR_and_SOC.sh script also resets the zero of energy to be the valence band maximum, allowing for easy comparison between spin-orbit-coupled and scalar-relativistic band structures.
Table 1: Table of spin-orbit splitting values for this Problem. Diamond C has already been supplied.
Appendix I: Structure information

Figure 9: Cubic (black) and primitive (grey) unit cell for bcc, fcc and diamond Si (from left to right).

<table>
<thead>
<tr>
<th>crystal structures</th>
<th>atomic coordinates</th>
<th>lattice vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcc</td>
<td>0 0 0</td>
<td>0 a/2 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a/2 0 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a/2 a/2 0</td>
</tr>
<tr>
<td>diamond</td>
<td>0 0 0</td>
<td>0 a/2 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a/4 a/4 a/4</td>
<td>a/2 0 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a/2 a/2 0</td>
<td></td>
</tr>
<tr>
<td>bcc</td>
<td>0 0 0</td>
<td>-a/2 a/2 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a/2 -a/2 a/2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a/2 a/2 -a/2</td>
<td></td>
</tr>
<tr>
<td>zincblende with A</td>
<td>A 0 0 0</td>
<td>0 a/2 a/2</td>
</tr>
<tr>
<td>atom species A and B</td>
<td>B a/4 a/4 a/4</td>
<td>a/2 0 a/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a/2 a/2 0</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Solids: Atomic coordinates and lattice vectors for different crystal structures.

<table>
<thead>
<tr>
<th>diamond(001)</th>
<th>atomic coordinates</th>
<th>lattice vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 0</td>
<td>a/√2 0 0</td>
</tr>
<tr>
<td></td>
<td>a/2√2 a/4</td>
<td>0 a/√2 0</td>
</tr>
<tr>
<td></td>
<td>a/2√2 a/2</td>
<td>0 0 L</td>
</tr>
<tr>
<td></td>
<td>0 a/2√2 3a/4</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Surfaces: Atomic coordinates of an ideal diamond(001) surface. Note: $a$ is the bulk lattice constant and $L$ is the total slab thickness ($L = a + L_{vac}$ with the vacuum size $L_{vac}$).
Appendix II: High symmetry \( k \)-points

<table>
<thead>
<tr>
<th>( fcc )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L )</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>( \Gamma )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( X )</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>( W )</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>( K )</td>
<td>0.375</td>
<td>0.375</td>
<td>0.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( fcc(001) )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{\Gamma} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \bar{J} )</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \bar{K} )</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: High symmetry points for \( fcc/diamond \) bulk and (001) surface structures given in units of the three reciprocal lattice vectors (\( k = x_1 b_1 + x_2 b_2 + x_3 b_3 \)). In the case of \( fcc/diamond \) bulk, the reciprocal lattice vectors form a \( bcc \) structure and the corresponding lattice vectors can be found in Tab. 2.

Figure 10: Brillouin zone and high symmetry points for \( fcc/diamond \) structure. The three coordinate axes (\( k_x, k_y, \) and \( k_z \)) form a Cartesian coordinate system – not to be confused with the reciprocal lattice vectors of the \( fcc/diamond \) structure.
Appendix III: Python scripts

```python
#!/usr/bin/python
import os
import shutil
import numpy as np

# Change script here

# Enter your calculated value for the minimal lattice constant
aMin = 5.1
# Enter your calculated value for the maximal lattice constant
aMax = 5.7
# Enter a value for the stepwidth
step = 0.1
# Number of calculations
n = int(np.rint((aMax - aMin)/step)) + 1

vector = np.zeros([3,3])
# Lattice vector in units of the lattice constant
vector[0] = [0.0,0.5,0.5]
vector[1] = [0.5,0.0,0.5]
vector[2] = [0.5,0.5,0.0]

# Number of basis atoms
atom_no = 2

atoms = np.zeros([atom_no,3])
# Basis atoms in fractional coordinates
atoms[0] = [0.0,0.0,0.0]
atoms[1] = [0.25,0.25,0.25]

# Change script here

for i in range(n):
aLat = aMin + step * i
print("Processing lattice constant %10.6f AA." % aLat)
# Create directory
dirname = str(aLat)
if not os.path.exists(dirname):
    os.makedirs(dirname)
# Change to directory
os.chdir(dirname)
# Write geometry.in
filename = "geometry.in"
f = open(filename,"w")
# The lattice
for lat in lattice_vector :
f.write("lattice_vector {:10.6f} {:10.6f} {:10.6f}\n", format(lat[0],lat[1],lat[2]))
# The atoms
for a in atoms:
f.write("atom_frac {:10.6f} {:10.6f} {:10.6f} Si\n", format(a[0],a[1],a[2]))
# Close file
f.close()
# Copy the control file
shutil.copyfile("./control.in","control.in")
# Run FHI-aims on 4 processes
os.system("mpirun -n 4 aims.x > aims.out")
# Change back to former directory
os.chdir("..")
```

Figure 11: Example python script for running calculations for several lattice constants used in Part I (skel/problem_1/02_3x3x2/diamond/run_diamond.py).
#!/usr/bin/python
import os
import numpy as np

# Minimal lattice constant in Å (from run script)
aMin = 5.1
# Maximal lattice constant in Å (from run script)
aMax = 5.7
# Sampling density in Å (from run script)
step = 0.1
# Number of basis atoms
atom_no = 2

# Number of calculations
n = int(np.rint((aMax - aMin)/step)) + 1

data = open("energies.dat","w")
data.write("%14s %14s
% ("aLat (Å)", "energy (eV/atom)")")

for i in range(n):
  aLat = aMin + step * i
  print("Postprocessing lattice constant %10.6f Å." % aLat)
dirname = str(aLat)
filename = dirname + "/" + "aims.out"
# Check if calculation was running
if (not os.path.isfile(filename)):
  print("%s was not processed." % filename)
else:
  # Check for convergence
  f = open(filename,'r')
  converged = False
  for line in f:
    if "Have a nice day" in line:
      converged = True
    if "Self-consistency cycle converged." in line:
      converged = True
  f.close()

if (not converged):
  print("%s is not converged." % filename)
else:
  # Grep for total energy
  f = open(filename,'r')
  for line in f:
    if "Total energy corr" in line:
      linesplit = line.split()
      energy = float(linesplit[5])/float(atom_no)
data.write("%14.6f %14.6f
% (aLat, energy))
  f.close()
data.close()
```python
#!/usr/bin/python

import sys
from math import sqrt

def output_lattice_vector(x, y, z):
    print "lattice_vector %.16f %.16f %.16f" % (x, y, z)

def output_atom(x, y, z, name):
    print "atom %.16f %.16f %.16f %s" % (x, y, z, name)

def output_constrained_atom(x, y, z, name):
    print "atom %.16f %.16f %.16f %s" % (x, y, z, name)
    print " constrain_relaxation .true."

A = 5.416 # Lattice constant
L_vac = 30. # Vacuum
A_1x1 = A/sqrt(2.) # 1x1 surface periodicity
n_layer = 4 # Number of layers in z-direction
Z = A/4. # Layer separation in z-direction
C = 0.5 * A_1x1 # Row separation in x- and y-direction

# H-saturation is put at this fraction of where the next Si atom would have been.
frac_H = 0.63

# (2x1) reconstructed lattice:
output_lattice_vector(2*A_1x1, 0., 0.)
output_lattice_vector(0., A_1x1, 0.)
output_lattice_vector(0., 0., n_layer*Z+L_vac)

# Hydrogen saturation
# The next Si would have been at (+/-C, 0., -Z).
output_atom( -frac_H*C, 0., -frac_H*Z, "H")
output_atom( +frac_H*C, 0., -frac_H*Z, "H")
# The next Si would have been at (2*C+/ -C, 0., -Z).
output_atom(2*C-frac_H*C, 0., -frac_H*Z, "H")
output_atom(2*C+frac_H*C, 0., -frac_H*Z, "H")

# Bottom Si layer
output_atom(0*C, 0., 0*Z, "Si")
output_atom(2*C, 0., 0*Z, "Si")
# Other Si layers
output_atom(0*C, C, 1*Z, "Si")
output_atom(2*C, C, 1*Z, "Si")
output_atom(1*C, C, 2*Z, "Si")
output_atom(3*C, C, 2*Z, "Si")
output_atom(1*C, 0., 3*Z, "Si")
output_atom(3*C, 0., 3*Z, "Si")
```

Figure 13: The python script used in Part III (skel/problem_6/01_ideal_2x1_4_layers/write-geom.py). The script creates a geometry output (which can be redirected to a file) for the ideal hydrogen saturated 2×1 Si(001) surface with 4 layers.
Acknowledgments

We would like to thank the testers of this tutorial for their time and feedback.

References


