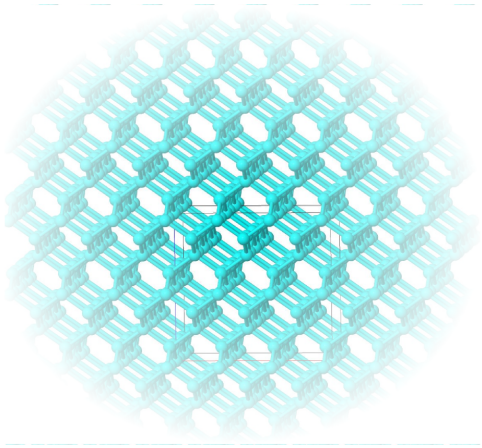
Hands-On Tutorial on *Ab Initio* Molecular Simulations

Berlin, July 12 – 21, 2011



Tutorial II: Periodic systems
Manuscript for Exercise Problems

Prepared by Jürgen Wieferink, Lydia Nemec, and Volker Blum.
Fritz-Haber-Institut der Max-Planck-Gesellschaft
Berlin, July 14, 2011

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:

1. introduces basic bulk properties 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: Electronic band structure & density of states
2. discusses surface calculations.
   - Problem V: Electronic structure of crystal surfaces
   - Problem VI: Relaxing surface structures
3. covers magnetism and collinear spin calculations on iron.
   - Problem VII: Lattice constant of non-magnetic iron
   - Problem VIII: Ferromagnetic iron
   - Problem IX: Anti-*ferro*magnetic iron

Dedicated folders have been prepared in the `skel/` directory for each of these problems. Please use this directory hierarchy, in particular because a few of the directories contain some helpful files. One of the first problems contains two scripts. The first one prepares and starts FHI-aims calculations on a series of geometries and the second one postprocesses the resulting FHI-aims output. Please try to adapt these scripts to the other problems along the rest of this tutorial.

All the problem sets are carefully designed to maximize your learning progress within the limited time of this tutorial. In case you get stuck 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 can also use this folder to compare your results to.

Take your time to read the tasks and their supplementary information carefully before proceeding to actually prepare the calculations. Each subtask starts with a short summary (gray box) and gives details and hints afterwards. The summaries are particularly imporant; read every single sentence very carefully. Also, feel free to consult the supplementary information presented in the Appendices on atomic structures and high symmetry k-points.

Part I: Basic properties of solids

In the first part of this tutorial we will work on different structural phases of bulk silicon. The correct description of their pressure dependence by Yin and Cohen [1] is one of the early success stories of first-principles DFT. In this part you will learn how to calculate basic properties of solids like lattice constants, cohesive energies, band structures, and density of states.

Please use the basic settings given in Fig. 1 as default for all problems of this session (unless specified otherwise).

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

# SCF settings
  charge_mix_param    0.2
  n_max_pulay         8
  sc_accuracy_rho     1E-4
  sc_accuracy_eev     1E-2
  sc_accuracy_etot    1E-5
  sc_iter_limit       40

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

Figure 1: Default physical settings for `control.in`. This file can be found in `skel/problem_1/control.base.in`.

For all calculations the Perdew-Wang LDA (`xc pw-lda`) exchange-correlation functional will be used. The effect of using different xc functionals has been discussed in Tutorial 1: "The 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. The nuclear charge of silicon ($Z = 14$) is still small enough to allow for a nonrelativistic treatment. But as the correction is computationally inexpensive, it does not hurt to use it, either. Just be sure to never compare total energies from different relativistic settings. The SCF settings have been discussed in detail in the first tutorial but can also be looked up in the manual. The `k_grid` setting will be discussed in the actual exercises.

Additionally, use the default "light" species settings for silicon in `$AIMSFILES/species_defaults/light/14_Si_default`. The environmental variable `$AIMSFILES` should points to the folder containing the auxiliary files of the FHI-aims distribution.

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. As a next step, we learn how to set basic parameters in `control.in` for periodic calculations. Finally, we 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 (see Appendix I). Use the experimental lattice constants a of 3.8 Å for *fcc*, 3.1 Å for *bcc*, and 5.4 Å for diamond.
- Visualize them (e. g. with GDIS).

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`. For example, *fcc* Si with a lattice constant 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   0.0   0.0   0.0   Si
```

A full set of lattice vectors and atomic positions of primitive unit cells for *fcc*, *bcc*, and diamond can be found in Appendix I.

The simplest way to check the `geometry.in` file is to visualize the corresponding geometry. For periodic structures in FHI-aims, we recommend GDIS, a free GTK-based program for the display and manipulation of isolated molecules and periodic systems, developed by Sean Fleming. Information on the program and the source code can be obtained from `http://gdis.seul.org`. The GDIS installed on the workshop machines was modified by Jörg Meyer to read the aims `geometry.in` format as well as the aims charge density output files `.cube` format).[1]

To visualize a structure given in `geometry.in` with GDIS, please type

```
gdis geometry.in &
```

Then, to repeat unit cell geometries and visualize periodic structures, choose the "Model: Images" and increase the number of cells as shown in Fig. 2.

[1] You may also try JMOL instead of GDIS. JMOL can read the aims `geometry.in` format. Just type (`jmol geometry.in &`). To get periodic images, click the right mouse button and choose "symmetry" → "reload {444 666 1}". More information about JMOL can be found at `http://jmol.sourceforge.net/`.

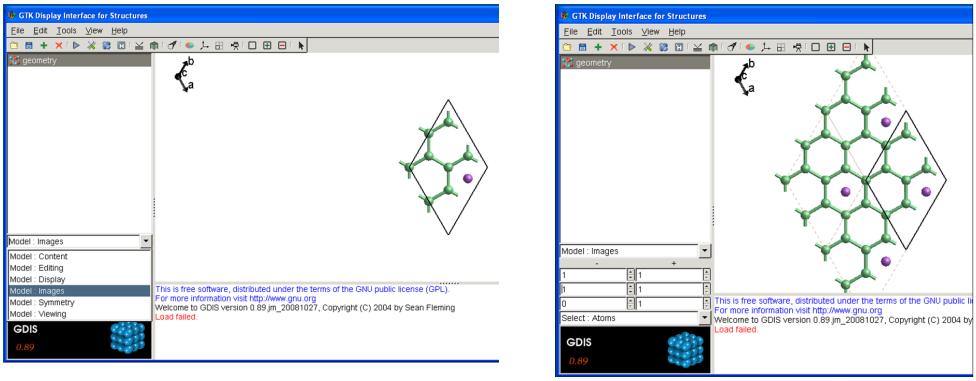


Figure 2: Show periodic replica/images using GDIS.

Setting up `control.in` und 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).
- Use the provided `run.sh` script (Fig. 3) to calculate total energies 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: 2 min]

The `control.in` for periodic calculationss looks much the same as for the cluster case (the underlying numerics are the same). There is one important difference, though, as 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 added to `control.in`:

```
k_grid 3 3 3
```

Note that the grid factors refer to the reciprocal lattice vectors corresponding to the real-space lattice vectors in `geometry.in`. If there are inequivalent lattice vectors, their order in `geometry.in` determines the ordering of reciprocal lattice vectors in the code. The k-point settings will be discussed with more detail in the next problem.

In general, you can run FHI-aims just like in the cluster case by

```
mpiexec -n 4 $AIMS_BIN </dev/null >aims.out
```

where the environmental variable `$AIMS_BIN` contains the actual FHI-aims executable. It is good practice to use a separate directory for every single run of FHI-aims in order to preserve the exact input files along with the output files.[2] In this tutorial, however,

[2] Using a separate directory for each FHI-aims run interactively is slightly simplified by using the `run_aims.sh` script. See `run_aims.sh --help` for help.

most of the calculations can be started using a prepared script which takes care of these things and 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 shell script provided in `skel/problem_1/02_3x3x3/run.sh` and shown in Fig. 3. This example script calculates the total energy of *fcc* Si with different lattice constants. Please copy this script to dedicated folders for *bcc* and diamond Si (we suggest `skel/problem_1/02_3x3x3/bcc` and `.../02_3x3x3/diamond`) and adjust the copies according to the lattice constants given in this subtask and the lattice structures given in Appendix I.

```
#!/bin/bash
set -e   # Stop on error
for A in 3.5 3.6 3.7 3.8 3.9 4.0 4.1; do
    echo "Processing␣lattice␣constant␣$A␣AA."
    mkdir $A
    # Use this construct for simple calculations. As values
    # are replaced verbatim, always put them into "(", ")".
    A2=$(python -c "print␣($A)/2.")
    # Write geometry.in
    cat <<EOF >$A/geometry.in
# fcc structure with lattice constant $A AA.
lattice_vector    0.0  $A2  $A2
lattice_vector    $A2  0.0  $A2
lattice_vector    $A2  $A2  0.0
atom  0.0   0.0   0.0 Si
EOF
    # Write control.in
    cp control.in $A/
    # Now run $AIMS_BIN with 4 processors in directory $A
    cd $A
    mpiexec -n 4  $AIMS_BIN </dev/null >aims.out
    cd ..
done
```

Figure 3: Example input shell script for running calculations for several lattice constants. This file can be found in `skel/problem_1/02_3x3x2/fcc/run.sh`.

To retrieve the total energies, you should then use the `postprocess.sh` script, which is provided within the same folder and printed in Fig. 4. This script extracts the total energies and writes them to the file `energies.dat`, along with the 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. You can use the idiom `Eatom=$(python -c "print␣($E)/2.")`, which calculates the

```
#!/bin/bash
for A in 3.5 3.6 3.7 3.8 3.9 4.0 4.1; do
    # Check for problems and errors marked by stars aims.out.
    grep --with-filename '*' $A/aims.out
    # Get 6th column from the line
    # containing "Total energy corr".
    E=$(gawk '/Total energy corr/ {print $6}' $A/aims.out)
    # Get 7th column out of the line with "Total time   "
    time=$(gawk '/Total time    / {print $7}' $A/aims.out)
    # Write results to data files.
    echo $A $E >>energies.dat
    echo $A $time >>times.dat
done
```

Figure 4: Example shell script to retrieve information from the FHI-aims output file. This file can be found in `skel/problem_1/02_3x3x2/fcc/postprocess.sh`.

energy per atom and puts it into the bash variable `$Eatom`.

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 &
```

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. As you might or might not already have heard of, the experimentally most stable phase is the diamond structure. We will show in the next two problems that the very coarse $3\times3\times3$ k grid is the reason of 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 convergence was not properly checked. 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 his 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 and the same lattice constants as in Problem I.
- Prepare one plot with all total energies and another with the computational times drawn against lattice constant. 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: 2 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 the last problem but with the appropriate changes to `control.in`. In particular, use the scripts provided for the Problem I (Fig. 3 and Fig. 4).

In the metallic *fcc* and *bcc* phases, the total energy of the 3×3×3 calculation differs by about 0.3 eV from 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. The semiconducting diamond Si phase, however, shows a 0.6 eV error for the 3×3×3 calculation and very good convergence already 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.

Looking at the computational times you see two general trends in FHI-aims. First, the times strongly decrease towards larger lattice constants. This is because there is less overlap between atoms and so less integrations are needed. The approach of FHI-aims is particularly efficient for "open" structures where the atoms occupy more space and thus have less neighbors. Second, increasing the number of k-points does not affect the calculational times significantly up to comparably dense k-grids. A total energy calculation with a 12×12×12 grid is not so much more expensive than a 3×3×3 calculation. Only with even denser k-grids computational times increase noteworthy.

In conclusion, we should 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 one plot with the total energies and another with the computational times. Add the results for the `minimal+spd` basis set from the k-point convergence test above.

[Estimated total CPU time: 2 min]

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 polarisation" (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 "*`# "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 significantly 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`.

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.

Bonus: Effect of Gaussian broadening

Bonus: Please skip this subtask if you run out of time.

- Calculate the total energies for *fcc* Si as a function of the lattice constant for a Gaussian broadening of $\sigma = 0.1\,\text{eV}$. Use the same lattice constants and computational settings as before with a $12\times12\times12$ 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 $\sigma = 0.1\,\text{eV}$ and the default value of $\sigma = 0.01\,\text{eV}$ from the previous calculations.

[Estimated total CPU time: 1 min]

You can explicitly set the Gaussian broadening to $\sigma = 0.1\,\text{eV}$ 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" (not to be confused with the true free energy of Tutorial 5) specified right of "`Electronic free energy`". From these two numbers, FHI-aims backextrapolates to the total energy without broadening and writes the resulting number right 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 correction is unphysical and may not be used.

For diamond Si the Gaussian broadening of course makes no difference at all as long as the broadening σ is small compared to the band gap. The first thing to notice for the metallic phases is that all of these numbers agree to each other within about 2 meV. For the default broadening of $\sigma = 0.01\,\text{eV}$ 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 smearing of $\sigma = 0.01\,\text{eV}$ 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 tutorial as it is the most accurate value for metals and makes no difference for semiconductors.

Problem III: Phase stability and cohesive properties

After finding "converged" computational settings, we now revisit the phase stability of bulk silicon in Problem I. Please note that in practice you 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 energy of *fcc*, *bcc*, and diamond Si as a function of lattice constant a. Use the settings from Problem II (k-grid of 12×12×12, `minimal+spd` basis) and the same lattice constants as in Problem I.
- Plot the results as done in Problem I.

[Estimated total CPU time: 2 min]

Problem I. 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.[3]

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 atom is of course spin polarized and you should use "`spin collinear`" instead of "`spin none`" as well as properly initialize the magnetization with "`default_initial_moment hund`".

Second, we use a more converged basis. In particular, uncomment all basis functions up to and including "tier 3", increase the cutting potential to "`cut_pot 8. 3. 1.`",

[3] If you happen to use different lattice constants from what is specified in Problem I, you might run into trouble with *bcc* Si at $a = 2.6$ Å. The one-particle energies differ that strongly for different k-points that the default number of states calculated per k-point is not large enough. Look up the keyword `empty_states` in the manual to fix the problem, then.

and turn off basis dependent confining potentials with "`basis_dep_cutoff 0.`" in the `species` section of `control.in`.[4]

Additionaly, 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_{coh}) of a crystal is the energy per atom needed to separate it into its constituent neutral atoms. E_{coh} is defined as

$$E_{\mathrm{coh}} = -\frac{E_{\mathrm{bulk}} - N E_{\mathrm{atom}}}{N} = -\left[\frac{E_{\mathrm{bulk}}}{N} - E_{\mathrm{atom}}\right], \tag{1}$$

where E_{bulk} is the bulk total energy per unit cell and N the number of atoms in the unit cell. E_{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. The simple cubic (super-)cell has the volume $V_{\mathrm{sc}} = a^3$. This number has to be divided by the number of atoms N_{sc} in this cell $V_{\mathrm{atom}} = a^3/N_{\mathrm{sc}}$. 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.

In summary, a file `energies.dat` containing the lattice constants and and total energies per atom can be converted to a file `cohesive.dat` containing atomic volumes V_{atom} and (negative) cohesive energies $-E_{\mathrm{coh}}$ by

```
../convert-coh.awk Nsc=.. Eatom=.. <energies.dat >cohesive.dat
```

where the "`..`" need to be replaced by the corresponding values of N_{sc} and E_{atom}. The heart of the script, which is given in `skel/problem_3/convert-coh.awk` is the following line:

```
gawk '!/#/ {printf "%.8f  %.8f\n", $1**3/Nsc, $2 - (Eatom)}'
```

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. It is plausible that upon high pressure, phases with more compact atomic volumes might be favorable.

[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 exact 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.

Equation of states and pressure dependence

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

An important equilibrium quantity we can calculated from our data is the equilibrium lattice constant a_0. 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 states [3, 4]. The energy per atom ($E = -E_{\mathrm{coh}}$) is expressed as a function of the atomic volume ($V = V_{\mathrm{atom}}$)

$$E(V) = E_0 + \frac{B_0 V}{B_0'}\left[\frac{(V_0/V)^{B_0'}}{B_0' - 1} + 1\right] - \frac{B_0 V_0}{B_0' - 1}. \tag{2}$$

The fitting parameters V_0 and E_0 are the equilibrium atomic volume and atomic energy, B_0 the bulk modulus and B_0' its derivative with respect to pressure. Equation (2) can be derived by assuming a constant pressure derivative B_0'.

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

```
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 output in units of eV/Å^3 because the cohesive energies and atomic volumes were provided in eV and Å^3. 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 by $a_0 = \sqrt[3]{N_{\mathrm{sc}} V_0}$.

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 we here 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 one can enforce many different atomic volumes smaller than the equilibrium ones. 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. 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 we can deduce a transition pressure of 15 GPa at which diamond and *bcc* Si could coexist according to our calculations. This is about $1.5 \cdot 10^5$ times the ambient pressure of about 100 kPa. Note that there are additional phases for silicon which have not been calculated here. For a more thorough discussion, please refer to [1].

Problem IV: Electronic band structure & density of states

- Calculate the electronic structure of diamond Si using the equilibrium geometries found in Problem III
- Choose the band structures along the high symmetry lines
  $L \to \Gamma \to X \to W \to K$.
- For the DOS use the energy range of -18 eV to 0 eV, Gaussian broadening of 0.1 eV, a k-grid of $12 \times 12 \times 12$, and `dos_kgrid_factors` of 5 for each k-grid.

[Estimated total CPU time: 1 min]

In order to calculate the band structure, we have to specify the high symmetry points to FHI-aims. An example excerpt from `control.in` corresponding to the first part of the suggested path with 50 points per path reads like this:

```
# diamond band structure:
output band  0.5   0.5   0.5    0.0   0.0   0.0    50    L  Gamma
output band  0.0   0.0   0.0    0.0   0.5   0.5    50    Gamma  X
output band  ...
```

Please refer to Appendix II: for the location of these high symmetry points.

The density of states (DOS)

The density of states is one of the basis concepts in solid state physics. Particularly, the DOS around the Fermi level is of interest as it will give you information about the characteristic of your system, for example, whether it is conducting, semi-conducting or insulating. Many material properties depend on the DOS for instance the conductivity.

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

$$n = \int_{\epsilon_0 - \Delta\epsilon}^{\epsilon_0 + \Delta\epsilon} g(\epsilon)\ d\epsilon \tag{3}$$

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 (δ peaks) and can be written as

$$g(\epsilon) = \sum_i \delta(\epsilon_i - \epsilon). \tag{4}$$

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

$$g(\epsilon) = \frac{1}{V_{BZ}} \sum_i \int_{BZ} d^3k\ \delta(\epsilon_{i,k} - \epsilon). \tag{5}$$

In order to calculate the density of states numerically, we have to replace the integral over the Brillouin zone in Eqn. (5) by a sum over k-points. In the case of infinite k-points this replacement is exact. 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 σ.

$$g(\epsilon) = \frac{1}{\sqrt{2\pi}\sigma}\frac{1}{n_k}\sum_i\sum_k \exp\left[-\frac{1}{2}\left(\frac{\epsilon-\epsilon_{k,i}}{\sigma}\right)^2\right] \tag{6}$$

Within FHI-aims, we have to specify in the `control.in`:

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

The first two values define the energy window of interest, the first value is the lower energy bound and the second value is the upper energy bound. The third value is an integer specifying the number of energy data points, and the last value gives the Gaussian broadening σ. All energies (bounds and broadening) are given in eV.

In the case of independent particles (ip) the total energy of the system (E_{ip}) is given by

$$E_{ip} = \int_{-\infty}^{\epsilon_F} \epsilon g(\epsilon)\, d\epsilon, \tag{7}$$

where ϵ_F is the Fermi-energy. Also in the DFT this integral contributes to the total energy. Now Eqn. 6 replaces $g(\epsilon)$ in Eqn. 7

$$E_{ip} = \frac{1}{\sqrt{2\pi}\sigma}\frac{1}{n_k}\int_{-\infty}^{\epsilon_F} \epsilon \sum_i\sum_k \exp\left[-\frac{1}{2}\left(\frac{\epsilon-\epsilon_{k,i}}{\sigma}\right)^2\right] d\epsilon. \tag{8}$$

Small changes in the shape of a peak barely affect the integral and comparably coarse k-grids for the sum in Eqn. 8 can be used. The same argument applies for the self-consistent field procedure. There a rather coarse k-grid (defined by the keyword `k_grid`) in combination with rather broad choices of σ (given by `occupation_type gaussian`) can be used. For a well resolved DOS, however, a denser k-grid to determine the sum over k-points in Eqn. 6 is necessary. After self-convergence is reached, the DOS is computed using an auxiliary k-grid that is made denser by factors $n1$,$n2$, $n3$, respectively. The factors $n1$ to $n3$ have to be given in the `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 `kn(1)*n1`, `kn(2)*n2`, `kn(3)*n3`, where `kn(`i`)` are dimensions of the old k point grid.

- After each calculation, use the python script `aimsplot.py`, which can be found in `$AIMSFILES/utilities/`, 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. Fortunately, the script `aimsplot.py` is smart enough to do so as long as the `geometry.in` and `control.in` files are in the same directory. Simply run `aimsplot.py` without any arguments in this directory.

You see a band structure with an indirect band gap of about 0.6 eV. Please note that in contrast to convention the energy zero is at the Fermi energy and not at the valence band maximum. This is of course much smaller than the experimental band gap of silicon. This disagreement is commonly called the "band gap problem" of (semi-)local functions.

Part II: Basic surface properties

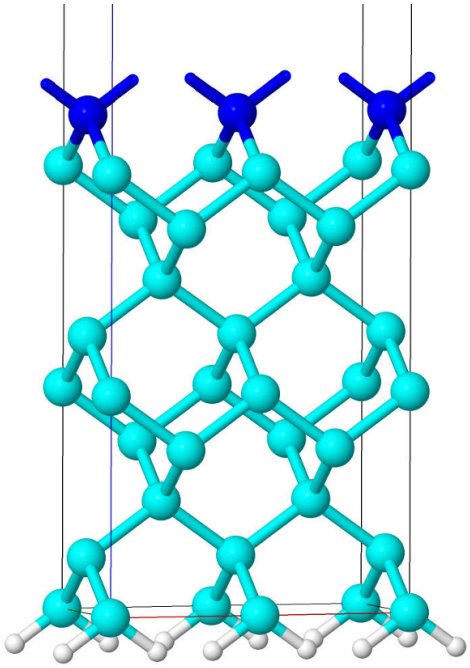


Figure 5: The hydrogen saturated 2×1 Si(001) surface. The cyan (light) atoms correspond to the bulk Si atoms, the blue (dark) atoms are the top Si atoms and the white atoms on the bottom layer are hydrogen atoms.

In the second part of this tutorial, we come to standard techniques needed to describe surfaces. As a physical system, we use the clean Si(001) surface, one of the technological most relevant surfaces. The surface reconstruction at low temperatures were unclear for many years. There was a long discussion and different models were used to understand the differences between various experimental (LEED, STM) and theoretical approaches. Finally, it has been shown by direct evidence in STM by Wolkow [5] that the main surface feature at low temperatures is the asymmetric dimer in a 2×1 reconstructed surface unit cell.

Please use the same `control.in` settings as given in the introduction to Part I. For saturating the bottom layer of the slab we additionally need the *light* species defaults for hydrogen. As you can see in Fig. 5 each silicon atom has to be saturated by two hydrogen atoms.

Problem V: Electronic structure of crystal surfaces

Creation and visualization of `geometry.in`

- Use the provided Python script (Fig. 11) 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.

The geometry of four layers of Si(001)-(1×1) in diamond structure is given in the 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 shifted by $a/\sqrt{2}$ in x direction and doubling the corresponding lattice vector. Please use the optimized lattice constant obtained in the last problem.

Obviously, a slab always has two surfaces, a top and a bottom one (see Fig. 5). In most cases one is only interested in one of these surfaces. In order to avoid physical states from the bottom layer within the fundamental gap, the bottom silicon layer is saturated with hydrogen atoms. Additionally one can argue that the atomic environment of the hydrogen saturated silicon atoms is closer to bulk silicon atoms.

Each bottom layer silicon atom needs two hydrogen atoms placed about 1.5 Å in the direction where the next silicon atoms would have been in the bulk geometry.

For your convenience, we provide a simple python script (the source code is given in Appendix III) for the ideal hydrogen saturated four layer slab. You should only adjust the lattice constant in line 14 and execute the script by typing `./write-geom.py`.

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

To determine a sufficient amount of vacuum between slabs, you would actually need to run several calculations with different vacuum thickness. In FHI-aims vacuum is cheap, you can easily use a vacuum of $L_{\mathrm{vac}} = 30$ Å without noticeable performance impact. This value is already given in the script by the variable `L_vac` in line 15.

Band structure and DOS calculation

- Prepare a `control.in` and use "safe" k-point settings.
- Calculate the density of states (DOS) and the surface band structures for these systems along $\bar{\Gamma} \to \bar{J} \to \bar{K}$.

[Estimated total CPU time: 2+7 min]

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. 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. As k-points are particularly cheap for this system, use a "safe" grid of $12 \times 24 \times 1$.

To visualize the band structure, use the script `aimsplot.py` just like before. Simply run the script in the directory that contains the input and output files of FHI-aims.

When plotting the band structures as before, note the differences between the four- and the eight-layer slabs. While the bands within the fundamental band gap, which stem from the surface, hardly change with the number of layers, the bands get about twice as dense in the valence and the conduction bands. These bands are bulk-like; the more layers you add, the more of these bands occur. If you have time, generate a 16 layer slab and calculate its band structure.

Problem VI: Relaxing surface structures

- Create a `geometry.in` with the two top Si atoms of the four-layer geometry in Problem V perturbed.
- Perform a structure relaxation of the top three layers of the resulting geometry.
- Calculate the DOS and the band structure of the resulting system along $\bar{\Gamma} \to \bar{J} \to \bar{K}$ (should be specified within the same `control.in`).
- Compare this band structure to the ones obtained in the last Problem.

[Estimated total CPU time: 15–30 min]

For structure relaxations to forces smaller than 10^{-2} eV/Å, add the following line to `control.in`:

```
relax_geometry trm 1d-2
```

During a structure optimization, we like only parts of the structure to relax and the rest kept at a fixed position. This is done by the keyword `constrain_relaxation`, which fixes the position of the last specified atom in `geometry.in`. In the FHI-aims manual you find different options, but for constraining all coordinates of an atom, we use the flag `.true.`. In the `geometry.in` file please write:[5]

```
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 ...
```

As a next step, the top Si atoms have to be perturbed. The perturbation is best chosen in the order of 0.5 Å. This ensures that the initial geometry is sufficiently far away from the symmetry induced saddle point of the ideal geometry. Also make sure to clearly break the mirror symmetry among these two silicon atoms, e. g. by moving down one of the two atoms 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
```

[5] You can achieve the same result by replacing `output_atom(...)` by `output_constrained_atom(...)` in `write-geom.py`.

The structure optimization with this starting geometry will take about 10 iterations. 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 15 to 45 minutes.

You can have a look at geometries along the relaxation path if you run the script `create_geometry_zip.pl` *`aims.out`* and unzip the resulting file `geometries.zip` (or using `jmol -s geometry.spt`). You will see that one dangling bond per top silicon atom is saturated by forming a dimer among them. Additionally, this dimer gets asymmetric as experimentally evidenced by Wolkow [5].

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    50    Gamma  J
output band  0.5   0.0   0.0    0.5   0.5   0.0    50    J  K
```

and for the DOS

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

If you compare the band structure of the reconstructed and clean Si(001) surface around the Fermi level you find that by means of the asymmetric dimer the surface gets semiconducting and there is a small but clear gap between the valance and the conduction band.

Part III: Magnetism

Magnetic materials show a variety of unique physical properties. In this part, we focus on the most common magnetic solid, namely iron. Iron is the most common element in the whole earth. In particular, it is not only abundant in its crust, it also forms much of the earth's outer and inner core. Iron can be found in many different structures and magnetizations reaching from the ambient (low pressure and low temperature) ferromagnetic *bcc* phase up to more complex structures under earth's core conditions (360 GPa, 5700 K), as sketched in the phase diagram in Fig. 6.

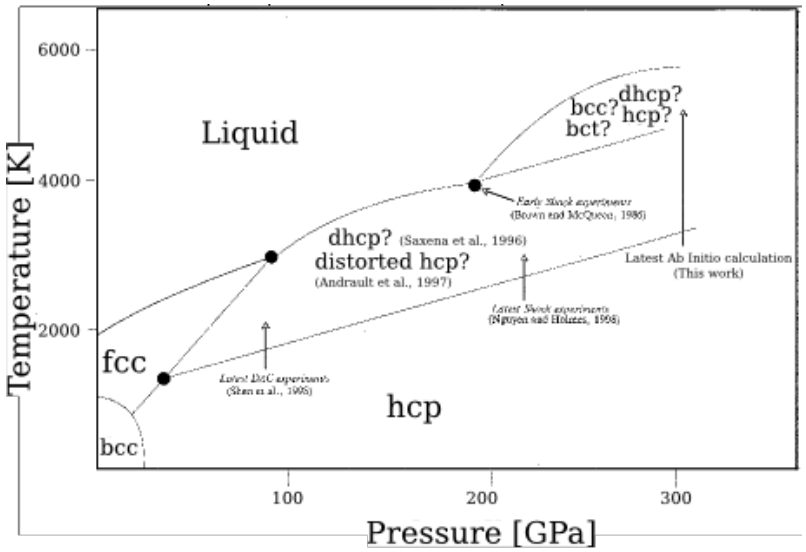


Figure 6: Phase diagram of iron ranging from the ambient (low temperature and low pressure) *bcc* phase up to pressure and temperatures present in earth's inner core. Adapted from [6].

At ambient conditions, iron crystallizes in a ferromagnetic *bcc* phase. This will be the main focus of this part. Later on, in Problem IX we will actually consider an anti-ferromagnetic *fcc* phase. and compare the relative stability of three phases in LDA.

Convergence tests

In order to safe precious tutorial time, the convergence tests have already been performed beforehand. Figure 7 shows a logarithmic plot of the differences between the total energy calculated at different k-grid densities to the converged energy (calculated with a $48 \times 48 \times 48$ grid). A grid of $16 \times 16 \times 16$ gives a good compromise of computational time and physical accuracy. This grid is significantly denser than what is needed in the case of silicon. As stated earlier in this tutorial, smaller band gaps as well as smaller unit cells call for finer integrations grids in reciprocal space. Metals

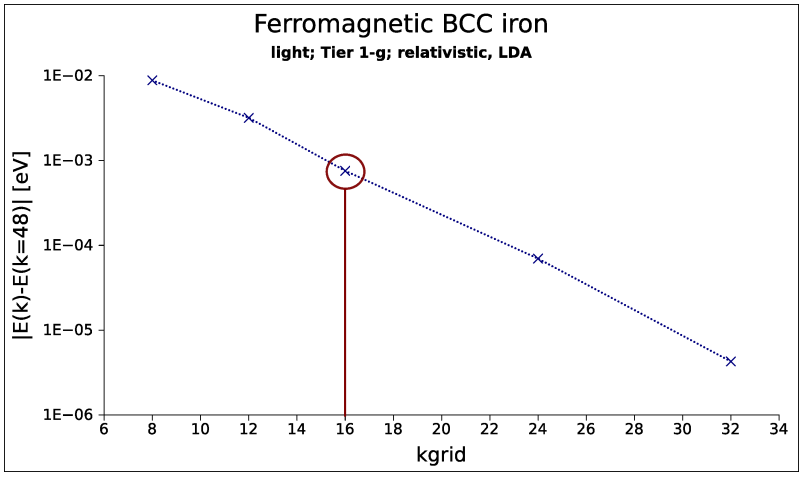


Figure 7: Convergence tests for the k-grid.

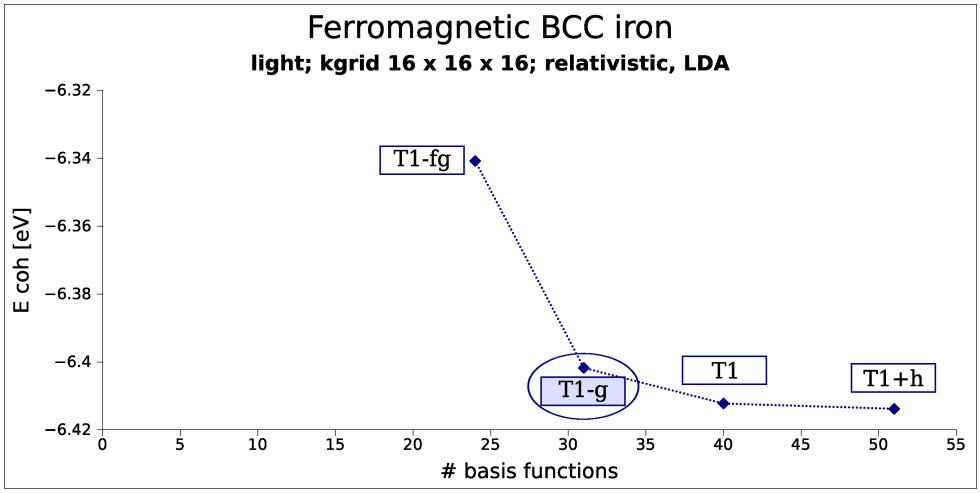


Figure 8: Convergence tests for the cohesive energy with respect to the basis set.

are particularly demanding. Throughout the iron part we ask you to use a grid of $16 \times 16 \times 16$. Please specify in the `control.in`:

```
k_grid   16 16 16
```

As stated before, basis set convergence is particularly important. Figure 8 shows cohesive energies for different basis sets. As you can see, the "T1-fg" basis set (tier 1 without f and g functions) is clearly underconverged. By giving the valence electrons the possibility to polarize by partially occupying f functions the cohesive energy can be significantly lowered. This basis size is also used in the default `light` settings of iron. Please use these settings, given in `$AIMSFILESspecies_defaults/light/26_Fe_default`, throughout this part of the tutorial.

For a efficient convergence the linear mixing parameter should be chosen carefully as metals, in particular magnetic ones, are harder to converge. For the same reason, the smearing should be broader:

```
charge_mix_param  0.1
occupation_type  gaussian  0.1
```

Problem VII: Lattice constant of non-magnetic iron

Find the equilibrium lattice constant of non-magnetic *bcc* iron

- Calculate the equilibrium lattice constant of non-magnetic *bcc* iron by performing five total energy calculations in steps of 0.15 Å around the experimental lattice constant of $a = 2.87$ Å. You find the corresponding lattice vectors in Tab. 1 of Appendix I.

[Estimated total CPU time: 2 min]

The lattice constant can be calculated completely along the lines of Problems III and IV (you may adapt the bash script given in Part I Fig. 3) but with the `control.in` adjusted as stated in the introduction of this Part. First, we are interested in the non-magnetic state, leave the `spin` setting to `none`. As you can see, the lattice constant is underestimated in LDA, which is related to the typical overbinding of this functional.

Calculate the energy of a free Fe atom

- Perform a free atom calculation along the lines of Problem III. In particular, use `spin collinear` for the free atom. Calculate the cohesive energy of non-magnetic *bcc* iron.

[Estimated total CPU time: <1 min]

Just as explained in the first part of this tutorial for the free atom calculation, special care has to be taken. First, the free atom is spin polarized and instead of "`spin none`" you have to use "`spin collinear`". The magnetization has to be initialized properly with "`default_initial_moment hund`".

Second, we use a more converged basis. In particular, uncomment all basis functions up to and including "tier 3", increase the cutting potential to "`cut_pot 8. 3. 1.`", and turn off basis dependent confining potentials with "`basis_dep_cutoff 0.`".

Calculate the density of states

- For the equilibrium lattice constant in LDA of $a = 2.69$ Å calculate and plot the DOS.

[Estimated total CPU time: 4 min]

We have already discussed the density of states of silicon in the first part of the tutorial. For iron, a smaller smoothening factor σ should be chosen. Please use for your DOS calculation the settings given below:

```
output dos -18.  0.  200  0.05
dos_kgrid_factors 5 5 5
```

If you have time left, you may like to increase `dos_kgrid_factors` to a higher value. The quality of the DOS plot will improve.

As before, you can use the script `aimsplot.py` to plot the DOS. The syntax of the DOS output is rather straight-forward so that you can also use Xmgrace (`xmgrace KS_DOS_total.dat`) for this task. The DOS shows a sharp peak at the Fermi energy.

Electron bands can spontaneously split into up and down spins. This happens if the relative gain in exchange energy is larger than the loss in kinetic energy. To understand when a material becomes *ferro*magnetic, E. C. Stoner formulated a model [7] from which he could derive a criteria for ferromagnetism

$$2\mu_B^2 I N(\varepsilon_\mathrm{F}) > 1, \tag{9}$$

where I is the Stoner parameter which is a measure of the strength of the exchange correlation of the electrons in the system, $N(\varepsilon_\mathrm{F})$ is the DOS at the Fermi energy ε_F. The factor $2\mu_B^2$ is a constant prefactor. According to the Stoner criterion, a very high DOS at the Fermi energy $N(\varepsilon_\mathrm{F})$ strongly favors ferromagnetism. This will be checked next for our particular case.

Problem VIII: Ferromagnetic iron

Find the equilibrium lattice constant of ferromagnetic *bcc* iron

- Calculate the equilibrium lattice constant of *ferro*magnetic *bcc* iron by performing five total energy calculations in 0.1 Å steps around a lattice constant of $a = 2.8$ Å.
- Extract the magnetic moment from the calculations above and plot.

[Estimated total CPU time: 7 min]

You should now explicitly take spin into account by specifying

```
spin collinear
```

in `control.in`.

It is much harder to converge the SCF for magnetic materials compared to non-magnetic ones. For one thing, convergence strongly depends on a sensible initialization of the charge and spin densities. Therefore, FHI-aims will refuse to run if you do not specify the initial magnetic moments of the atoms. You can do so with `default_initial_moment` *`value`* in `control.in`, where *`value`* is either a real number (specifying $N_\uparrow - N_\downarrow$) or "`hund`" which means to initialize the atom in a high-spin state. Alternatively, you can add `initial_moment` *`value`* after the `atom`s in `geometry.in` to specify the initial moments of each atom separately.

Please initialize the magnetic moment of the iron atom with $N_\uparrow - N_\downarrow = 2$:

```
atom  0.  0.  0.  Fe
  initial_moment 2.
```

Also note that convergence is more sensitive to all kinds of other settings. So please be careful to use all specified settings or be prepared for comparably slow convergence (the number of necessary SCF cycles might easily increase by an order of magnitude).

If you compare the lattice constant of *bcc* Fe to the one of *fcc* Fe, you will see that the equilibrium lattice constant increases from 2.69 Å to 2.75 Å.

You can see that the magnetic moments (from the line containing `N = N_up - N_down`) get smaller with decreasing lattice constant. This is quite a general trend among magnetic materials and is a consequence of the Pauli principle.

Calculate and plot the cohesive energy of ferromagnetic *bcc* iron

- Calculate the cohesive energy of ferromagnetic *bcc* iron and compare it to the non-magnetic one. Reuse the free atom reference energy from Problem VII.
- Plot the cohesive energy versus the volume per atom of non-magnetic and ferromagnetic *bcc* iron in one plot (for example use xmgrace).

To convert into cohesive energies and atomic volumes, you can reuse the `gawk` script given in Problem III.

The cohesive energy of the ferromagnetic phase is more favorable by some 300 meV. Please note that energy differences between different magnetic states are often even much smaller than this.

Calculate the density of states for ferromagnetic *bcc* iron

- For the equilibrium lattice constant in LDA of $a = 2.75\,\text{Å}$ calculate and plot the DOS using the same settings as in the last Problem.

[Estimated total CPU time: 7 min]

As you can see from the density of states, the magnetization leads to a shift of the d-band peak away from the Fermi energy. In the majority spin channel, the d bands are shifted towards lower energy and mostly occupied whereas in the minority channel, the shift is in the opposite direction, giving mainly unoccupied d bands. By this reduction of symmetry, the instability mentioned in the last problem is resolved and the total energy decreases.

Problem IX: Anti-*ferro*magnetic iron

Of course, in a careful study, you should not stop your search after you found the expected results; so we now turn towards *fcc* iron.

Cohesive energy and DOS of non-magnetic *fcc* iron

- Calculate the cohesive energy of non-magnetic *fcc* iron for the given LDA lattice constant of $a = 3.41$ Å (single point calculation). Within the same FHI-aims run, also calculate the density of states using the same parameters as before.

[Estimated total CPU time: 10 min]

So far we have calculated two different phases of iron, namely non-magnetic and ferromagnetic *bcc* iron. Thin films of iron grown on top of a metallic surface e.g. copper, show anti-ferromagnetic order in *fcc* structure. Regrettably, we will not have the time to calculate thin films, but an important preparation for any thin film calculation is a bulk calculation.

Please use the lattice parameters given in Appendix I and the same parameters as in Problem VII (in particular `spin none`.

First you should note that the non-magnetic *fcc* iron phase is actually favored by about 50 meV compared to ferromagnetic *bcc* iron in LDA. Additionally, the DOS is still comparably high at the Fermi level, but there is not such a pronounced peak as in Problem VIII.

Writing and visualizing anti-ferromagnetic `geometry.in`

- Construct `geometry.in` files for anti-ferromagnetic Fe *fcc* at the minimum lattice constant (a=3.41 Å) with an anti-ferromagnetic order in (001) planes. As a first guess for the `initial_moment` use ± 0.5.

In an anti-ferromagnetic system there are spatially varying moments which average to zero by symmetry. Anti-ferromagnetism breaks the translational symmetry to some extent because atoms with a majority of spin-up electrons are not equivalent to atoms with a majority of spin-down electrons. Therefore, the atomic basis twice as large as before.

Often there are several options to realize an anti-ferromagnetic order. For example you can have the spin-up atoms ordered in sheets. The most important anti-ferromagnetic orders for an *fcc* lattice entail alternating spins along the [111] or the [001] directions, giving either (111) or (001) planes with atoms of equivalent moments. The hard task is to find the right primitive lattice vectors for each of these cases.

There are many different ways on finding the right unit cell, one is the recipe given below:

1. Find the atoms representing the spin up and spin down layer.

2. Find two primitive vectors spanning a plane. Within a layer in this plane the atoms should be ordered *ferro*magnetically

3. Now find a primitive vector out of this plane.

In Appendix I you will find the geometry for anti-ferromagnetic *fcc*(001) iron. As stated in the subtask description, use `initial_moments` of `0.5` and `-0.5`.

In order to check the geometry with a visualization program you can temporarily rename one of the two atoms in the atomic basis to get them drawn in different colors.

Cohesive energy and DOS of anti-ferromagnetic *fcc* iron

- Calculate the cohesive energy of anti-ferromagnetic *fcc* iron and the density of states (for the DOS use the same parameters as before). Additionally, plot the atom-projected DOS.
- Study the resulting magnetic moments.
- Plot the two points of non-magnetic and anti-ferromagnetic Iron in your plot from the first subtask of Problem VIII. Where does your point fall in the plot? Do you believe these points?

[Estimated total CPU time: 20 min]

In an anti-ferromagnetic system convergence is even harder than for a ferromagnetic one. If you start from a physical reasonable point, your calculation will converge, but it may take some time. Therefore we provide a `control.in` file in `skel/problem_9/control.base.in`. You will find, that we changed the default value (`2.0`) of the `preconditioner kerker` to `1.5` and the Pulay mixer setting `n_max_pulay` to `4`. The Gaussian broadening is reduced significantly[6] to `0.01` and the convergence criteria is further loosened for `sc_accuracy_rho` to `5E-4`. For the analysis of the final magnetic moment we add `output hirshfeld` to get a Hirshfeld partition of the charge and spin density and `output atom_proj_dos ...` for an atom-projected DOS. Please note that the latter does not respect the `dos_kgrid_factors` settings. Therefore, you should use a larger smearing of 0.2 eV.

The magnetic moment cannot be obtained as easily as in the ferromagnetic case because the total moment $N_\uparrow - N_\downarrow$ vanishes. There are several ways of partitioning the magnetic moments to the atoms. The Mulliken analysis (done because of the atom-projected DOS) gives a moment of ± 0.44 and the Hirshfeld partitioning results in a very similar moment of ± 0.41. There is an additional output of the quantity

$$\langle |n_\uparrow - n_\downarrow| \rangle = \int \mathrm{d}^3 r \;\; |n_\uparrow(\boldsymbol{r}) - n_\downarrow(\boldsymbol{r})| \tag{10}$$

[6] Please note that while in general a larger broadening enhances convergence, in rare cases a small broadening is more "lucky" to reach self-consistency.

after the line "`Integration grid: ...`". Its final value is 0.886 or about 0.44 per atom. This is a comparably small magnetization. However, the atomic magnetization increases with increasing lattice constant.

The atom-projected dos, e. g., for the first atom, can simply be plotted using

```
xmgrace -legend load atom_proj_dos_spin_*Fe0001.dat
```

However, the result is not as easily interpreted as in the case of ferromagnetic *bcc* iron in Problem VIII.

In the last part you performed calculations for different phases of bulk iron with LDA. In the final plot of Problem IX you find anti-ferromagnetic *fcc* iron as the most stable phase (closely followed by non-magnetic *fcc* iron). The phase diagram (Fig. 6) of [6] clearly shows *bcc* iron in the low temperature and low pressure regime.

This is a known deficiency of LDA with iron. LDA describes the electronic structure and geometry within a given phase correctly, but the energy hierarchy of different phases and the geometry is not well captured in LDA. In the case of iron, the deficiency is lifted by switching from LDA to PBE, as can be seen in Fig. 9.

For magnetic systems some functionals catch the physics better than others. For example in the case of palladium PBE fails to predict the most stable phase. The important message is, with DFT you can explore exciting physics of magnetic materials, but you have to be careful.

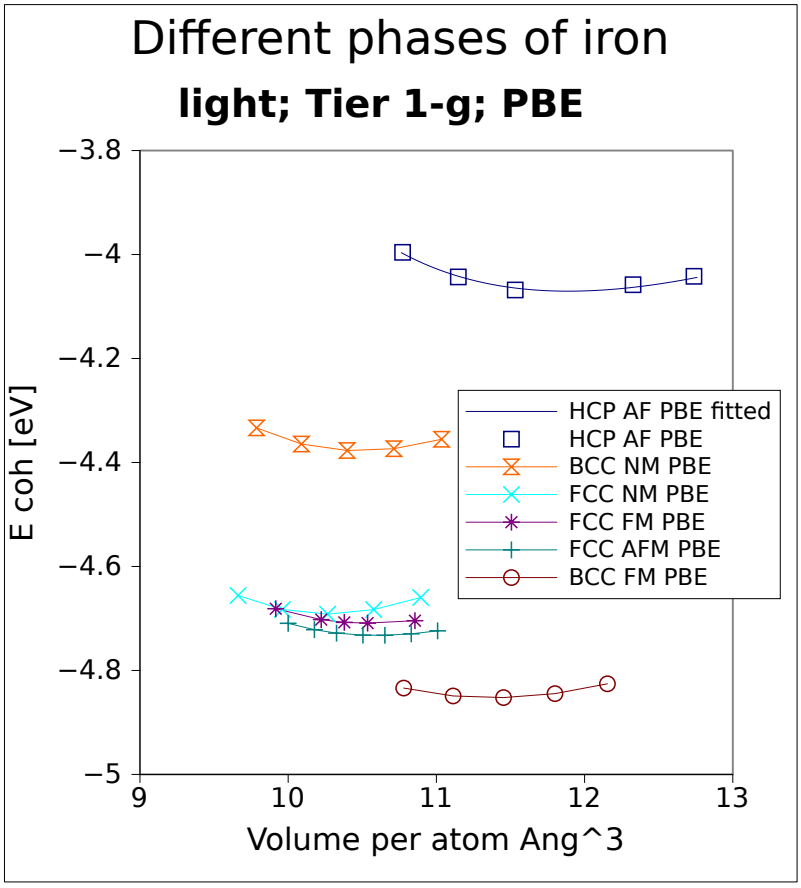


Figure 9: The cohesive energy is plotted against the volume using PBE for the xc-functional. Ferromagnetic *bcc* iron is the most stable phase at a zero temperature zero pressure.

Appendix I: Structure Information

Space group	atomic coordinates	lattice vectors
fcc	$0 \quad 0 \quad 0$	$0 \quad a/2 \quad a/2$ $a/2 \quad 0 \quad a/2$ $a/2 \quad a/2 \quad 0$
diamond	$0 \quad 0 \quad 0$ $a/4 \quad a/4 \quad a/4$	$0 \quad a/2 \quad a/2$ $a/2 \quad 0 \quad a/2$ $a/2 \quad a/2 \quad 0$
bcc	$0 \quad 0 \quad 0$	$-a/2 \quad a/2 \quad a/2$ $a/2 \quad -a/2 \quad a/2$ $a/2 \quad a/2 \quad -a/2$
hcp	$0 \quad 0 \quad 0$ $0 \quad -a/\sqrt{3} \quad c/2$	$a/2 \quad -a\sqrt{3}/2 \quad 0$ $a/2 \quad a\sqrt{3}/2 \quad 0$ $0 \quad 0 \quad c$
fcc anti-ferromagnetic (001)	$0 \quad 0 \quad 0$ $a/2 \quad 0 \quad a/2$	$a/2 \quad -a/2 \quad 0.0$ $a/2 \quad a/2 \quad 0.0$ $0.0 \quad 0.0 \quad a$

Table 1: Solids: Atomic coordinates and lattice vectors for different space groups. *Note*: $c/a = \sqrt{8/3} \approx 1.633$ for ideal *hcp*.

diamond(001)						*hcp*(0001)					
atomic coordinates			lattice vectors			atomic coordinates			lattice vectors		
0	0	0	$a/\sqrt{2}$	0	0	0	$a/2\sqrt{3}$	0	$a/2$	$-a\sqrt{3}/2$	0
$a/2\sqrt{2}$	0	$a/4$	0	$a/\sqrt{2}$	0	0	$-a/2\sqrt{3}$	$\frac{c}{2}$	$a/2$	$a\sqrt{3}/2$	0
$a/2\sqrt{2}$	$a/2\sqrt{2}$	$a/2$	0	0	L				0	0	L
0	$a/2\sqrt{2}$	$3a/4$									

Table 2: Surfaces: The atomic coordinates of an ideal diamond (001) and ideal *hcp* (0001) surfaces. *Note: a* is a 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

fcc	x_1	x_2	x_3
W	0.25	0.5	0.75
L	0.5	0.5	0.5
Λ	0.25	0.25	0.25
Γ	0	0	0
Δ	0	0.25	0.25
X	0	0.5	0.5
Z	0.125	0.5	0.625
W	0.25	0.5	0.75
K	0.375	0.375	0.75

fcc(001)	x_1	x_2	x_3
$\bar{\Gamma}$	0.0	0.0	0.0
$\bar{J}$	0.5	0.0	0.0
$\bar{K}$	0.5	0.5	0.0

Table 3: High symmetry points for *fcc*/diamond bulk and (001) surface structures given in units of reciprocal lattice vectors ($\boldsymbol{k} = x_1\boldsymbol{b}_1 + x_2\boldsymbol{b}_2 + x_3\boldsymbol{b}_3$).

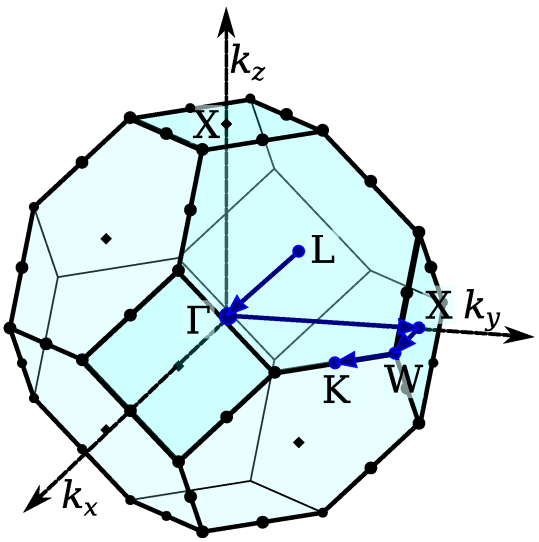


Figure 10: Brillouin zones and high symmetry points for *fcc* and diamond structures.

Appendix III: Si(001) geometry generation python script

```
#!/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)
print

# 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 11: The python script used in Part II (`skel/problem_5/01_ideal_2x1/write-geom.py`). The script creates a geometry.in file for the ideal hydrogen saturated 2×1 Si(001) surface with 4 layers.

Acknowledgments

We like to thank testers of this tutorials for their invaluable feedback. In particular, we thank Yong Xu for many helpful discussions.

References

[1] M. T. Yin and M. L. Cohen, *Microscopic Theory of the Phase Transformation and Lattice Dynamics of Si*, Phys. Rev. Lett. **45**, 1004 (1980).

[2] C. Kittel, *Introduction to Solid State Physics* (John Wiley & Sons Inc, 1986), 6 sub edition.

[3] F. Murnaghan, *The compressibility of media under extreme pressures*, in *Proc. Nat. Acad. Sci.*, volume 30, page 244 (1944).

[4] F. Birch, *Finite Elastic Strain of Cubic Crystals*, Phys. Rev. **71**, 809 (1947).

[5] R. A. Wolkow, *Direct observation of an increase in buckled dimers on Si(001) at low temperature*, Phys. Rev. Lett. **68**, 2636 (1992).

[6] L. Vocadlo, J. Brodholt, D. Alfè, M. J. Gillan and G. D. Price, *Ab initio free energy calculations on the polymorphs of iron at core conditions*, Physics of The Earth and Planetary Interiors **117**, 123 (2000).

[7] E. C. Stoner, *Collective Electron Ferromagnetism*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **165**, 372 (1938).