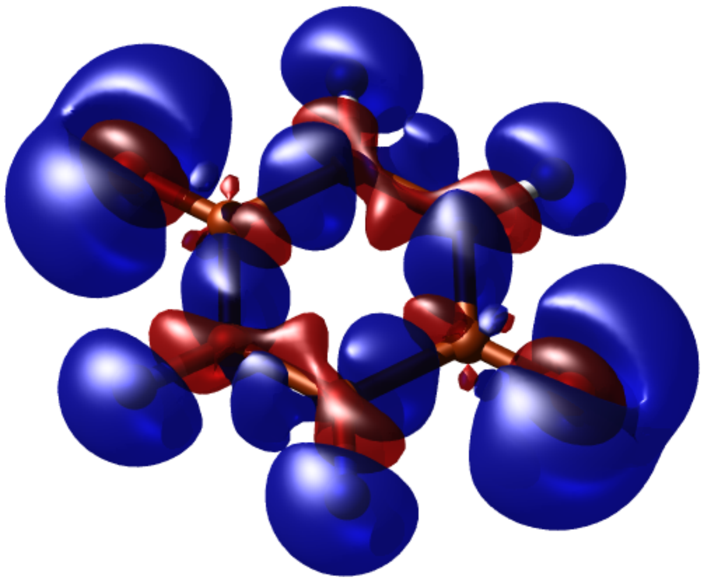
FHI-aims Tutorial

Basics of Electronic-Structure Theory with FHI-aims: Atoms and Molecules



Tutorial I and II: Basics of Electronic-Structure Theory
Manuscript for Exercise Problems

Prepared by Debalaya Sarker, Zhong-Kang Han, Maria K. Pogodaeva and Sergey V. Levchenko
Advanced Materials Modeling
Skoltech, April 22, 2021

A quick summary of the exercises

A guideline through the tutorial

This tutorial aims to give a basic introduction to electronic structure calculations for very simple systems. As every DFT code has its own philosophy, this tutorial should also familiarize you with fundamental aspects of using FHI-aims. The goal of the first section is to explain the basic inputs of FHI-aims and to demonstrate that DFT calculations can have predictive power for many observable quantities. The second part introduces geometric optimization of a molecule and how to assess the reliability of the result. Some exercises are marked with a red exclamation mark (!). These exercises demonstrate pitfalls or limitations of the approach.
The practice session consists of three parts:

Part I: Basic electronic structure with FHI-aims

Problem I: Total energy of free atoms
The hydrogen atom

Problem II: Hydrogen Molecule(H_2): bond length !

Part II: Local structure optimization

Problem III: Hydrogen Molecule H_2

This tutorial is located in your `$ home` directory

FHI-aims tutorial is already in your home directory:

- **`cd  $home/HandsOn/tutorial_2`**

For every exercise, we also provide solutions and sample input files. They can be found in $HandsOn/tutorial_2/solutions and $HandsOn/tutorial_2/skel/exercise_XX/templates, respectively. However, we strongly recommend to use the provided input files only in case of time shortage. You will maximize your learning progress by trying to generate the input files on your own. In case you get stuck with a particular problem, do not hesitate to ask one of the tutors. For the tutorials, an executable of FHI-aims will be provided on your workstation at `/usr/local/bin/aims.200926.parallel.x`. However, you have to submit the jobs to queue with submission script `/usr/local/bin/run_parallel.aims`

Note: Please do not copy and paste the description in this pdf into your input files. Typically, invisible characters from the formatting are copied, too. FHI-aims will reject these characters and your calculations will not start.

The very basics of FHI-aims

Each calculation should be done in a separate directory containing the two mandatory input files `control.in` and `geometry.in`. FHI-aims is then called in this directory.

In short:

- **Each calculation in a separate directory**
- **2 input files:**
  - `control.in`
  - `geometry.in`
- **copy submission script to working directory**
  - `cp /usr/local/bin/run_parallel.aims Path_to_Your_working_directory/`
  - **Start calculation**
    `sbatch run_parallel.aims`

The above starts a calculation on a single processor, shows the main output on the screen and, at the same time, pipes it to the `output` file. The `output` file contains the basic information and results of the calculation such as the total energy, atomic forces, and so forth. Additional output files might be generated according to the specified settings.
NOTE:

:

- Depending on the job in hand, you can modify the wall time in the submission script `job.sh` in each calculation directory.

```
#!/bin/bash

#SBATCH -J test                 # Job name
#SBATCH -o %j.out           # Name of stdout output file (%j expands to jobId)
#SBATCH -e %j.err            # Name of stdout output file (%j expands to jobId)
#SBATCH -N 1                    # Total number of nodes requested
#SBATCH -n 4                   # Total number of mpi tasks requested
#SBATCH -t 00:05:00        # Run time (hh:mm:ss) - 12 hours

module load Compilers/GNU/9.3.1

mpirun /usr/local/bin/aims.200926.parallel.x > aims.parallel.out
```

Figure 1: Sample submission script. You may alter the wall time # SBATCH -t, number of nodes # SBATCH -n depending on the job in hand.

Nuclear positions: geometry.in

The `geometry.in` file contains all information concerning the atomic structure of the system. This includes the nuclear coordinates, which are specified by the keyword `atom`, followed by cartesian coordinates (in units of Å) and the element symbol, called "species" in FHI-aims. In addition, comments can be added using a preceding hash symbol. Fig. 2 shows an example `geometry.in` file for a hydrogen atom.

```
#The hydrogen atom
atom 0.0 0.0 0.0 H
```

Figure 2: An example `geometry.in` file for a hydrogen atom positioned at the origin.

For periodic calculations, lattice vectors can also be given in this file. This will be covered in the next tutorial. In the present tutorial, however, we will stick to non-periodic systems.

Choosing the method: control.in

This file contains all physical and computational settings for the calculation. Fig. 3 shows a minimal example of a `control.in` file, which can be used as a template during the tutorial.

In this example, the following options are set:

- `xc`
  This keyword sets the method to be used. For example you can choose the option `hf`, which requests a Hartree-Fock calculation.

- `charge`
  Set the total charge of the system in units of $|e|$. For a neutral system, this is zero.

- `sc_iter_limit`
  Determines the limit of self-consistency steps allowed in the calculation.

```
#Simplest input settings
##################################################

xc  hf
charge 0.0

#####Iteration limit####################################

sc_iter_limit 300

#######Species###################################
```

Figure 3: Simple physical and computational settings for `control.in`.

Additional remark:
Please note that you also can change the convergence criteria of the self-consistent field (s.c.f) cycle using the keyword `sc_accuracy_rho` for the electron density. If this keyword is not set, FHI-aims uses a default that is rather stringent for most production calculations. The value chosen by the code for a given system can be found in the output file of each run. There are additional criteria that can be required for s.c.f. convergence. However, checking expensive quantities such as forces or stresses directly for s.c.f. convergence can be **very** costly and should be avoided unless there is a definite need.

In addition to these keywords, the `control.in` file must contain the definition of the computational parameters for each species that is specified in `geometry.in`. The order of the species in the listing is irrelevant. FHI-aims is shipped with pre-defined settings for all species which govern the key parameters regarding the numerical accuracy. They include, *inter alia*, the specification of all real-space integration grids, the accuracy of the Hartree potential and, of course, the basis set. For all elements, defaults are provided for three different levels of accuracy: *light*, *tight*, and *really tight*. Additional *intermediate* settings are provided for several much-used elements. The pre-defined settings can be found in the directory

```
$home/HandsOn/species_defaults
```

and should be copied and pasted into `control.in`, e.g. via the command

```
cat $home/HandsOn/species_defaults /really_tight/01_H_default » control.in
```

which pastes the really_tight settings of the H-atom into the `control.in` file. Already the *tight* species_defaults are rather safe and *really tight* settings are overconverged for most purposes. In addition the number of basis functions can be varied, as well as the basis functions themselves. The basis functions associated with a given species are tabulated at the end of these default settings, as shown in Fig. 4

The idea of keeping the species defaults out in the open is that, even if they are not modified, these are the critical accuracy parameters which one might look at to ensure numerical convergence. Each line denotes a specific basis function. They can be read as follows: The first keyword denotes the "type" of the basis function. *hydro* means that this is a hydrogen-like basis function. Some basis functions are of the type *ionic*. They are described in more detail in the manual. The next two symbols correspond to the first two quantum numbers (principal quantum number n, orbital quantum number l) of the basis functions, and the final number corresponds to the "effective nuclear charge" for which this basis function is created. `hydro 1 s 0.85` corresponds to the exact solution for the 1 s basis function of a hydrogen atom if it had a nuclear charge of only 0.85.

The basis functions are classified in "tiers" (i.e., levels of importance). Not all basis functions are enabled by default. Rather, some are commented out using the " # " symbol. They can be included in the calculation by removing the hash symbol from the corresponding lines. Systematically improved calculations can be performed by enabling additional tiers one after another (however, the computational cost for routinely overconverged calculations also increases rapidly).

```
########################################################################
#
#  FHI-aims code project
#  VB, Fritz-Haber Institut, 2007
#
#  Suggested "safe" defaults for H atom
#  (to be pasted into control.in file)
#
########################################################################
  species        H
#     global species definitions
    nucleus             1
    mass                1.00794
#
    l_hartree           8
#
    cut_pot             4.0  2.0  1.0
    basis_dep_cutoff    0.d0
#
    radial_base         24 7.0
    radial_multiplier   2
    angular_grids       specified
      division   0.2783  110
      division   0.3822  194
      division   0.5626  302
      division   0.5922  434
      division   0.6227  590
#      division   0.7206  770
#      outer_grid  770
      outer_grid  590
########################################################################
#
#  Definition of "minimal" basis
#
########################################################################
#     valence basis states
    valence      1  s   1.
#     ion occupancy
    ion_occ      1  s   0.5
########################################################################
#
#  Suggested additional basis functions. For production calculations,
#  uncomment them one after another (the most important basis functions
#  are listed first).
#
#  Basis constructed for dimers: 0.5 A, 0.7 A, 1.0 A, 1.5 A, 2.5 A
#
########################################################################
#  "First tier" - improvements: -1014.90 meV to -62.69 meV
     hydro 2 s 2.1
     hydro 2 p 3.5
#  "Second tier" - improvements: -12.89 meV to -1.83 meV
#     hydro 1 s 0.85
#     hydro 2 p 3.7
#     hydro 2 s 1.2
#     hydro 3 d 7
#  "Third tier" - improvements: -0.25 meV to -0.12 meV
#     hydro 4 f 11.2
#     hydro 3 p 4.8
#     hydro 4 d 9
#     hydro 3 s 3.2
```

Figure 4: Tabulated basis functions for hydrogen. The basis functions are classified in "`tiers`". In this example only the `tier 1` and minimal basis functions are enabled.

Free atoms and simple spin-polarized systems: Additions to control.in

Part I of this tutorial will deal with free atoms and very simple diatomic molecules. For a physically correct treatment, their spin will have to be considered. To do so, modify your `control.in` file as follows:

```
#Sample input file for the calculation of a H atom
##################################################

xc  hf
charge 0.0
spin collinear
default_initial_moment 1

#####Iteration limit####################################

sc_iter_limit 300

########Species###################################
```

Figure 5: Default physical and computational settings for simple spin-polarized systems in `control.in`.

These basic keywords should be used as default for part I of this tutorial, unless specified otherwise. However, **do not** use these modifications routinely in other calculations if you do not need them – see below for more information:

- `spin`
  This keyword governs the spin treatment. It can be set to `none`, which requests a spin-restricted calculation, or to `collinear`, which requests a spin-unrestricted (polarized) calculation. In a spin-restricted calculation, α and β spins are assumed to be equal. Only one spin-channel is treated and hence, the number of electrons is effectively halved. This accelerates the calculations significantly, typically a factor 2 or more.
  Important: In systems that are safely unpolarized, always use `spin none`.

- `default_initial_moment 1`
  Sets the initial spin of the atoms. The value 1 requests an initial spin moment of 1, i.e., one more spin-up electron than there are spin-down electrons. Only necessary for `spin collinear` calculations.
  Warning: Only ever use the `default_initial_moment` keyword for very simple systems, where all atoms are expected to behave identically. In more complex systems, where some atoms carry spin and others don't, never use `default_initial_moment`. Instead, modify the `geometry.in` file and add individual `initial_moment` keywords to specific atoms there. Initializing the s.c.f. cycle with an unphysical spin state can greatly slow down the calculation and can lead to physically incorrect results.

  Tip:
  In principle, one never needs to use `default_initial_moment` at all. It is much simpler (and less error-prone) to place `initial_moment` keywords after any atom to be spin-initialized in `geometry.in` in Figure 6:

```
#The correctly spin-initialized hydrogen atom
atom 0.0 0.0 0.0 H
  initial_moment 1.0
```

Figure 6: `geometry.in` file with spin initialization of a single H atom. Each atom can get its very own initial moment by placing an `initial_moment` tag in the next line for any atom that might carry a spin.

Additional tools and programs

Bash shell:
A short list of the basic bash (command line) commands is given in Appendix II.

All scripts you will need for this tutorial can be found in
`$HandsOn/tutorial_2/utilities`

Visualization tools:
To visualize structures, you may use jmol. A short jmol tutorial video can be found in $HandsOn/tutorial_2/jmol_tutorial.ogv. In case you are not familiar with jmol, use any of your favourite softwares viz. VESTA etc.

Plotting and Editing:
Matplotlib and xmgrace can be used to visualise and plot data. Also, you can use Origin, Gnuplot or any software of your choice that you are already familiar with.

Part I: Basic electronic structure with FHI-aims

Problem I: Total energy of free atoms: The hydrogen atom

In this exercise, we aim to convey the basics of FHI-aims using the hydrogen atom. The hydrogen atom is the simplest non-trivial system possible and the only one for which the exact analytic solution is known. By the end of the first exercise, we will see how various computational methods compare to each other and to the exact solution. From a technical perspective, we will learn how to generate input files, read the standard FHI-aims output, and perform basis set convergence tests.

Getting started - the hydrogen atom

Educational Objectives:

- Become aquainted with running FHI-aims calculations
- Learn how to do systematic basis set convergence

Tasks

1. Generate a simple `geometry.in` file by hand, which contains only a single hydrogen atom, using the example shown in Fig. 2. This corresponds to a single hydrogen atom in a hypothetical ideal gas phase. It is located at the origin of the coordinate system, although its position does not matter here.

2. Generate a simple `control.in` file by hand, using the example `control.in` file given in Fig. 5. Systems with only a single electron can be solved exactly (within the Born-Oppenheimer approximation) using Hartree-Fock. Finally, append the "*really_tight*" species data of H to the end of the `control.in` file, e.g. via the command
   ```
   cat $SPECIES_DEFAULTS/really_tight/01_H_default >> control.in
   ```

3. copy `HandsOn/run_parallel.aims` to working directory

4. Now, run FHI-aims:
   ```
   sbatch run_parallel.aims
   ```

   Once the calculation has finished, open the `output` file. If you find the line "`Self-consistency cycle converged.`" near the end, then your calculation is converged. We are now interested in the total energy. Search for the block

   ```
   | Total energy uncorrected   :  -0.136053823214315E+02  eV
   | Total energy corrected     :  -0.136053823214315E+02  eV
   | Electronic free energy     :  -0.136053823214315E+02  eV
   ```

   In this special case, all energies are equal, but this will not be the case if fractionally occupied orbitals were found! For non-metallic systems, as in this tutorial, always use the **Total energy uncorrected** value. (You will learn what the other two values mean in a future lecture.) Compare it with the exact result for the hydrogen atom (0.5 Hartree ≈ 13.6057eV).

   TIP:
   In later exercises, to find this value fast and efficiently, use the command
   ```
   grep 'Total␣energy␣uncorrected' output
   ```

5. Redo the calculation with different basis sets (`minimal` , `tier1`, `tier2`, `tier3`) by (un)commenting the basis functions at the end of the control.in file. Calculations with minimal basis set are performed by removing all basis functions that are listed in the file. Search the output file to find out how many basis functions are actually used in the calculations. Then, plot the total energy as function of the basis set size. At which tier does total energy converge?

   TIP:
   To plot the results, simply create a text file (e.g., *results.dat*) with two columns, the number of basis functions and the obtained total energy. This file can be plotted directly using xmgrace with the command
   ```
   xmgrace results.dat
   ```

In principle you can use any plotting software of your choice if you dont́ have xmgrace installed viz. Gnuplot, Origin etc.

Method performance

To learn about the performance of difference exchange-correlation functionals, repeat for different methods. Replace `hf` in control.in with

- `pw-lda`
- `pbe`
- `pbe0`

Do all methods converge with basis set size? Do all converge to the same result?

Optional: An optimal basis set

If you browse through the hydrogen basis set, you will note that the hydrogen 1s function is not included. Change that by adding the line

```
hydro 1 s 1
```

at the end of the `control.in` file. Comment out all other basis functions and run the Hartree-Fock calculation again. How close does it get to the exact result?

Problem II: Hydrogen molecule (H_2): bond length

Hydrogen Molecule (H_2)

One of the most influential papers in chemistry for systematic investigation of the performance of DFT was Johnson et al.[1], in which several properties of a large number of diatomic systems were consistently computed and compared to experimental values. In the style of this work, we will calculate the binding curve and atomization energy (ΔH_{at}) for hydrogen molecule (H_2) with two methods.

Eductational Objectives:

- Find the equilibrium bond distance of a simple diatomic molecule.

Tasks:

1. The first task of this exercise will be to find the equilibrium bond distance of H_2 from a series of calculations. Start by creating a `geometry.in.template` file which contains two H atoms, as shown in Fig. 7.

```
#H2 at variable bond distances
atom 0.0 0.0 0.0 H
atom 0.0 0.0 Dist H
```

Figure 7: The geometry data for a H_2 molecule. One H is put on the origin and the other H is located `Dist` Å away from the origin along the z-axis.

   In this example, One H is put on the origin and the other H is located `Dist` Å away from the origin along the z-axis. Hereby, `Dist` is a placeholder which has to be replaced by the actual distance for every calculation.

2. Create a `control.in.template` file, and specify a `hf` calculation for a neutral system. Feel free to copy contents from `control.in` file in the first exercise, but remove the *really tight* species setting for hydrogen and paste the *tight* setting for H into the `control.template` file.

3. Next, run calculations in separate folders for different bond distances (ideally between 0.5Å and 1.2Å with 0.1Å steps, and a denser step width of 0.02 Å between 0.65 Å and 0.85 Å).

For each distance, you should

- create a unique directory
- create the `control.in` and `geometry.in` file from templates
- replace the bond distance place holder `Dist` with the bond distance and
- start FHI-aims.

Run FHI-aims for all bond lengths and plot the total energies vs. the bond length.

Which bond length corresponds to the lowest energy? How does the bond length compare to the experimental bond length of 0.74Å?

4. To compare with experimental values, we compute the atomization energy (ΔH_{at}). In order to calculate ΔH_{at}, we will also need the total energy of the isolated H atom. Use the total energies for the single atom from previous exercise. Check the result carefully

   Calculate the atomization energy (ΔH_{at}) of H_2 by subtracting the free-atom energies from the predicted total energy of H_2 (i.e. the minimum total energy found when varying bond distances).

$$\Delta H_{at} = E_{tot}^{H_2} - 2E_{atom}^{H} \tag{1}$$

   How does this compare to the experimental value of $\Delta H_{at} = 103.3\text{kcal mol}^{-1}$ (4.479 eV)?

Optional: Method Performance.

Repeat the bond length determination using `pbe0`. How does the optimal bond length change? How much does the total energy and atomization energy change?

Part II: Local structure optimisation

Problem III: H_2

This exercise covers how to perform geometry optimizations. Specifically, we will relax the H_2 molecule of our previous exercise starting from an initial guess for the geometry and do avoid the equilibrium bond distance.

Educational Objectives

- Learn how to perform a geometry optimization in FHI-aims
- Visualize the relaxation.

1. Fig. 8 shows a `geometry.in` file for starting the relaxation calculationPlease use this as the starting point for your structure relaxation.

```
#H2 at starting bond distance for relaxation
atom 0.0 0.0 0.0 H
atom 0.0 0.0 0.5 H
```

Figure 8: This `geometry.in` file gives a starting point for the molecular H_2.

2. Create a `control.in` file, using the `control.in` file from the previous exercises as a template. For your `xc` functional (i.e. `method`), specify `pw-lda`. This time, use `spin none` Finally, add the keyword `relax_geometry trm 1E-3` to request structural relaxation. Your `control.in` should look similar to the example shown in fig. 9.

```
#This is a sample input file for execise 2 - relaxation calculation of H2
#####################################################################################################

##### Method #####################

  xc pw-lda
  charge 0.0
  spin none

#####  Iteration limit  #####
 sc_iter_limit     100

#####  Relax Initial Geometry ###########
 relax_geometry trm 1E-3
```

Figure 9: The `control.in` file used for the structure relaxation of the cation H_2.

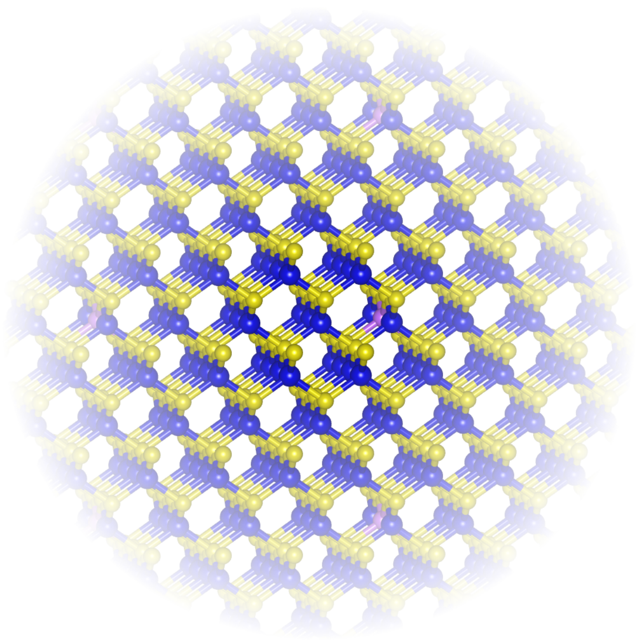
As in the exercises before, the basis set must be included in the `control.in` file. Use the "*tight*" species defaults for H atoms. **FHI-aims control file does not require repeated entries of species defaults.**
`cat $SPECIES_DEFAULTS/light/01_H_default  >> control.in`

3. Run FHI-aims.
`sbatch job.sh`

4. To visualize the results, copy the tool `create_relax_movie.py` from
ls `$HandsOn/tutorial_2/utilities` into the working directory. Apply it to the output and pipe the result to a new file using the command
`python create_relax_movie.py output > H2.xyz`
Open the file `H2.xyz` with a visualizer, e.g. jmol.
`jmol H2.xyz`
(You may use any viewer of your choice and can see jmol tutorial video $HandsOn/tutorial_2/jmol_tutorial.ogv)

Important:
Realize that the chosen relaxation criteria of `relax_geometry trm 1E-3` is rather harsh and should only be used for high accuracy and finite-difference calculations.

FHI-aims Tutorial: Density-Functional Theory and Beyond Advanced Materials Modelling



Tutorial III: Periodic Systems Manuscript for Exercise Problems

Prepared by Debalaya Sarker,
Zhong-Kang Han, Sergey V. Levchenko

CEST, Skoltech

What does the fully relaxed structure look like?
Do you believe that this could be the actual total energy minimum?
How is the optimized bond length different from the one you have found in the previous problem?

- Check how the final bond length changes with the change of functionals, tiers and relaxation conditions.

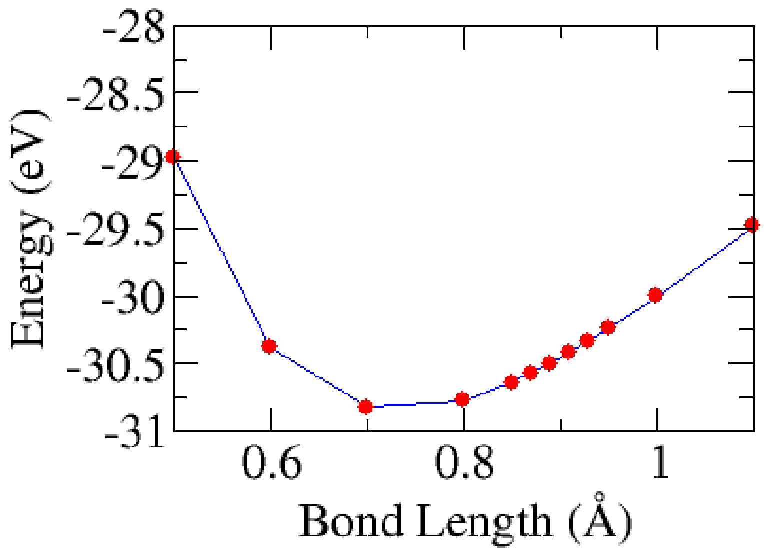


Figure 10: Bond length vs. Total energy plot for a H_2 molecule calculated with hf.

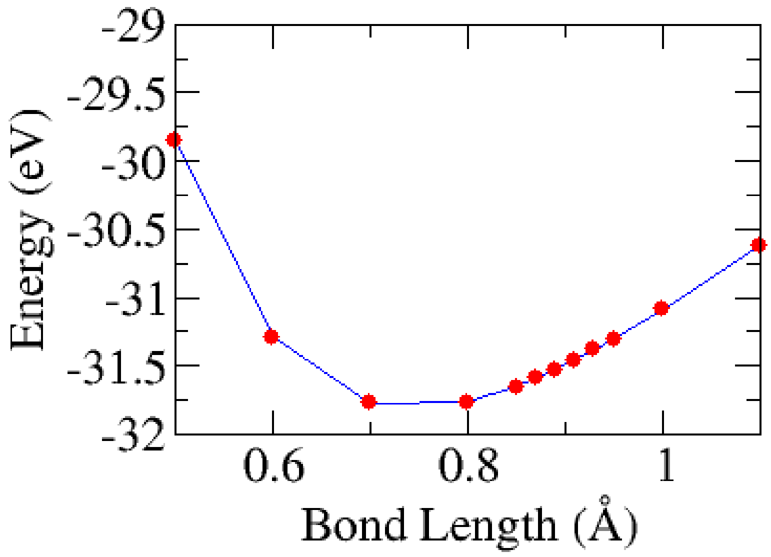


Figure 11: Bond length vs. Total energy plot for a H_2 molecule calculated with pbe0.

Results

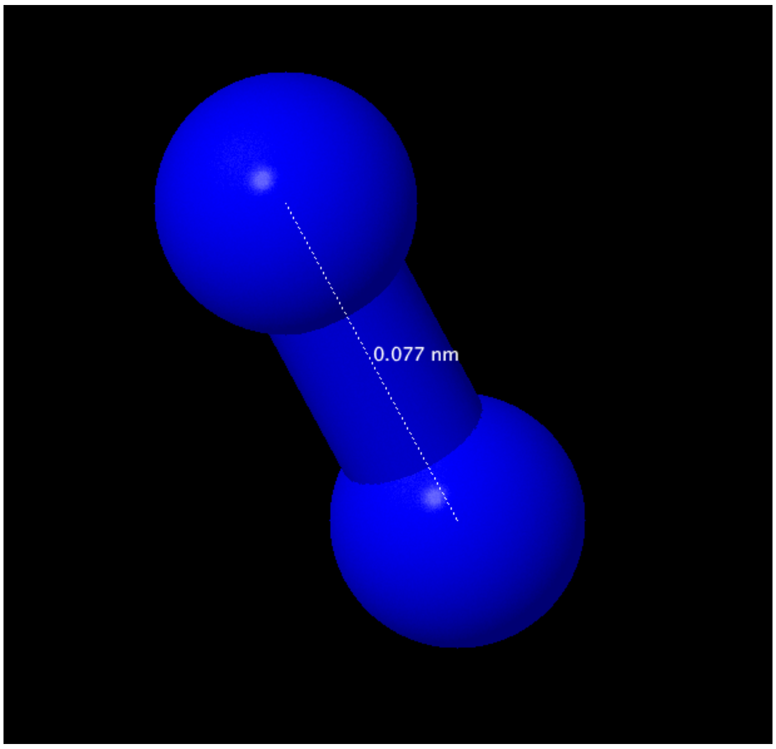


Figure 12: Bond length in H_2 molecule calculated with pw-lda.

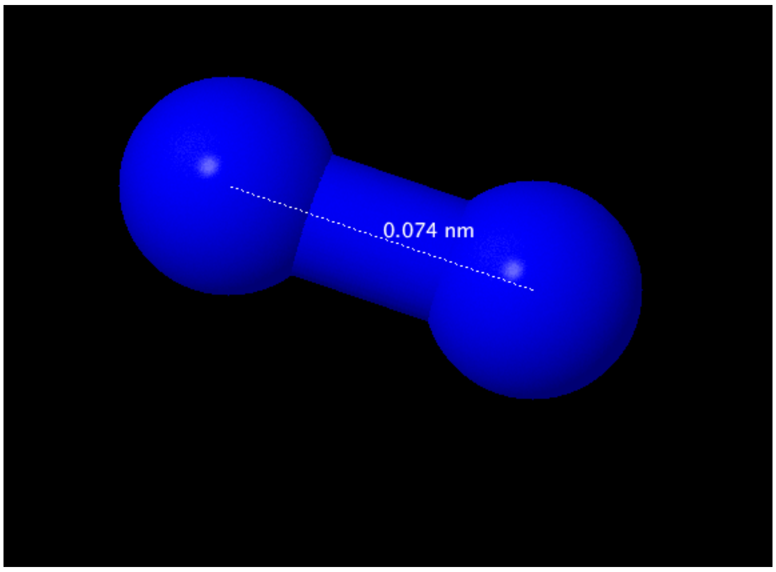


Figure 13: Bond length in H_2 molecule calculated with pbe0.

Appendix

Appendix I: Sample python/bash scripts

Sample script for Exercise 2

Below, you find a sample script in python and bash to calculate a molecule with pre-defined bond distances.

```
#########################################################################
# This is a simple example how to run a series of distances             #
# Note: To run this script, you must be in a directoy which contains    #
#  /  geometry.template file with HF, where the distance between H and F#
#     is determined by a variable named DIST                            #
#                                                                       #
#  /  a control.template file which contains                            #
#     the correct method and basis set specifications.                  #
# Example input files are provided in the same directory as this script #
#########################################################################

import os, shutil
#Variable defintion:
#This variable should point to your local FHI-aims executable
#Variable defintion:
AIMSBIN='aims.x␣'


f=open('geometry.in.template','r')     #read geometry template
template=f.read()
f.close

#Loop over the distances
for Distance in [0.7, 0.8, 0.85, 0.87, 0.89, 0.9,1 0.93, 0.95,\
                 1.0, 1.1, 1.2, 1.3]:
   print Distance
   # Create directory with the distance as name
   if not os.path.exists(str(Distance)): os.mkdir(str(Distance))
   # copy the control file into the new directoy
   shutil.copy('control.in.template', str(Distance)+'/control.in')
   # the the geometry template  and
   # replace the term DIST by the current distance and
   # copy the new file into the directory
   out=open(str(Distance)+'/geometry.in','w')
   template_out=template.replace('Dist',str(Distance))
   out.write(template_out)
   out.close()
   # Change directory
   os.chdir(str(Distance));
   # Run aims and pipe the output into a file named "output"
   os.system(AIMSBIN+'␣|␣tee␣output')
   # Go back to the original directory
   os.chdir('..')
```

Figure 14: This is a sample python script for calculating a molecule with different pre-defined bond distances.

```bash
##########################################################################
# This is a simple example how to run a series of distances             #
# Note: To run this script, you must be in a directoy which contains    #
#  /  geometry.template file with HF, where the distance between H and F#
#     is determined by a variable named DIST                             #
#                                                                        #
#  /  a control.template file which contains                             #
#     the correct method and basis set specifications.                   #
# Example input files are provided in the same directory as this script #
##########################################################################

#!/bin/bash -l
ulimit -s unlimited

#Variable defintion:
#This variable should point to your local FHI-aims executable
AIMSBIN=aims.x


#Loop over the distances
for Distance in 0.7 0.8 0.85 0.87 0.89 0.91 0.93 0.95 1.0 1.1 1.2 1.3 ;
 do
   echo $Distance
   # Create directory with the distance as name
   mkdir $Distance;
   # copy the control file into the new directoy
   cp control.in.template $Distance/control.in
   # the the geometry template  and
   # replace the term DIST by the current distance and
   # copy the new file into the directory
   sed "s/Dist/$Distance/g" geometry.in.template > $Distance/geometry.in
   # Change directory
   cd $Distance;
   # Run aims and pipe the output into a file named "output"
   $AIMSBIN | tee output;
   # Go back to the original directory
   cd ..
done;
```

Figure 15: This is a sample bash script for calculating a molecule with different pre-defined bond distances.

Sample script for Exercise 10

```
#########################################################################
# This is a simple example how to run a series of distances             #
# Note: To run this script, you must be in a directoy which contains    #
#  /  geometry.template file with HF, where the distance between H and F#
#     is determined by a variable named DIST                            #
#                                                                       #
#  /  a control.template file which contains                            #
#     the correct method and basis set specifications.                  #
# Example input files are provided in the same directory as this script #
#########################################################################

import os, shutil
#Variable defintion:
#This variable should point to your local FHI-aims executable
#Variable defintion:
AIMSBIN='aims.x␣'


f=open('geometry.in.template','r')     #read geometry template
template=f.read()
f.close

#Loop over the distances
for Distance in [0.7, 0.8, 0.85, 0.87, 0.89, 0.9,1 0.93, 0.95,\
                 1.0, 1.1, 1.2, 1.3]:
   print Distance
   # Create directory with the distance as name
   if not os.path.exists(str(Distance)): os.mkdir(str(Distance))
   # copy the control file into the new directoy
   shutil.copy('control.in.template', str(Distance)+'/control.in')
   # the the geometry template  and
   # replace the term DIST by the current distance and
   # copy the new file into the directory
   out=open(str(Distance)+'/geometry.in','w')
   template_out=template.replace('Dist',str(Distance))
   out.write(template_out)
   out.close()
   # Change directory
   os.chdir(str(Distance));
   # Run aims and pipe the output into a file named "output"
   os.system(AIMSBIN+'␣|␣tee␣output')
   # Go back to the original directory
   os.chdir('..')
```

Figure 16: This is a sample python script that tests different settings for the density mixing schemes.

Appendix II: Bash

Bash is a Unix shell and command language for the GNU Project and the default shell on Linux and OS X systems. We will use it to execute most programs and exercises. Below you find a list of the most import commands. It furthermore offers a full programming language (shell script) to automatize tasks e.g. via loops.

Basic control:

```
TAB - auto completion of file or command
Up/Down - See previous commands
CTRL R - reverse search history
Middle Mouse Button - Paste at prompt position
CTRL L - Clear the terminal
!! - repeat last command
```

Basic navigation:

```
ls -a - list all files and folders
ls <folderName> - list files in folder
ls -lh - Detailed list, Human readable
ls -l *.jpg - list jpeg files only
ls -lh <fileName> - Result for file only

cd <folderName> - change directory
cd / - go to root
cd ..- go up one folder, tip: ../../../
pwd - print working directory
```

Basic file operations:

```
cat <fileName> - show content of file
head - from the top
   -n <\#oflines> <fileName>
tail - from the bottom
   -n <\#oflines> <fileName>
mkdir - create new folder
mkdir myStuff ..
mkdir myStuff/pictures/ ..

touch <fileName> - create a file

cp image.jpg newimage.jpg - copy and rename a file
cp image.jpg <folderName>/ - copy to folder
cp image.jpg folder/sameImageNewName.jpg
cp -R stuff otherStuff - copy and rename a folder
cp *.txt stuff/ - copy all of *<file type> to folder

mv file.txt Documents/ - move file to a folder
mv <folderName> <folderName2> - move folder in folder
mv filename.txt filename2.txt - rename file
mv <fileName> stuff/newfileName
  if folder name has spaces use " "
mv <folderName>/ .. - move folder up in hierarchy

rm <fileName> .. - delete file (s)
rm -i <fileName> .. - ask for confirmation
rm -f <fileName> - force deletion of a file
rm -r <foldername>/ - delete folder
```

Extract, sort and filter data:

```
grep <someText> <fileName> - search for text in file
   -i - Doesn't consider uppercase words
   -I - exclude binary files
grep -r <text> <folderName>/ - search for file names
                               with occurrence of the text
```

Flow redirection -redirecting results of commands:

```
'>' at the end of a command to redirect the result to a file
ex --> ps -ejH > process.txt
'>>' to redirect the result to the end of a file
```

Chain commands

```
'|' at the end of a command to enter another one
    ex --> du | sort -nr | less
```

References

[1] B. G. Johnson, P. M. W. Gill and J. A. Pople, *The performance of a family of density functional methods*, The Journal of Chemical Physics **98**, 5612 (1993).