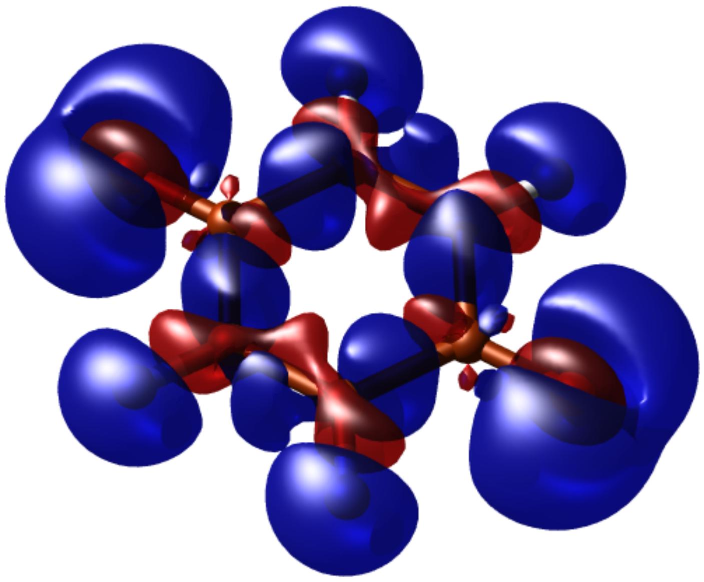
Hands-on workshop:

Density-functional theory and beyond - accuracy, efficiency and reproducibility in computational materials science

Berlin, July 31 - August 11, 2017



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

Prepared by Igor Y. Zhang, Hagen-Henrik Kowalski and Tonghao Shen
Fritz-Haber-Institut der Max-Planck-Gesellschaft
Berlin, July 31, 2017

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. The third part is dedicated to efficiently obtaining and visualizing electronic structure derived data. 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:

As first step please copy the folder `tutorial_1` from `$HandsOn` to your working directory.

For every exercise, we also provide solutions and sample input files. They can be found in $HandsOn/tutorial_1/solutions and $HandsOn/tutorial_1/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.

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 revoke 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`
- **Start calculation**
  `aims.x | tee output`

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.

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. 1 shows an example `geometry.in` file for a hydrogen atom.

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

Figure 1: 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. 2 shows a minimal example of a `control.in` file, which can be used as a template during the tutorial. These basic settings should be used as default for part I of this tutorial, unless specified otherwise.

```
#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 100

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

Figure 2: Default physical and computational settings for `control.in`.

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.
- `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.
- `default_initial_moment 1`
  Sets the initial spin of the atoms. The value 1 requests an initial spin moment of 1. Only necessary for `spin collinear` calculations.
- `sc_iter_limit`
  Determines the limit of self-concistency steps allowed in the calculation.

Additional remark:
Please note that you also can change the convergence criteria of the self-consistent cycle using the keywords `sc_accuracy_eev` for the sum of the Kohn-Sham eigenvalues, `sc_accuracy_etot` for the total energy, and `sc_accuracy_rho` for the electron density. However as the defaults implemented in FHI-aims produce reliable results we shall ignore them for the time being.

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*. They can be found in the directory

```
    $SPECIES_DEFAULTS
```

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

```
 cat $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. 3

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 functions. 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”. 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.

```
########################################################################
#
#  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 3: 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.

Additional tools and programs

Bash shell:
A short list of the basic bash (command line) commands is given in Appendix II.
Scripts:
For some exercises, scripts are required for dedicated tasks, such as the manipulation of cube files. All scripts you will need for this tutorial can be found in
```
  $HandsOn/tutorial_1/utilities
```
Visualization tools:
To visualize structures, vibrational modes, charge density plots, etc., several programs including molden, avogradro, vmd, jmol, and gdis are installed on your work stations. A short jmol tutorial video can be found in $HandsOn/tutorial_1/jmol_tutorial.ogv. Structure created in jmol can be saved as xyz-file by typing:
```
write "file.xyz" as xyz
```
in the jmol console (see tutorial video).
Format conversion:
To convert one atomic structure file format into another you can use openbabel. To list all availbe formats type:
```
babel -L formats
```
To convert an xyz fromat file into FHI-aims input type:
```
babel -ixyz file.xyz -ofhiaims geometry.in
```
Plotting and Editing:
Gnuplot, qtiplot, matplotlib and xmgrace are installed on your PC to visualise and plot data. Also several editors are available, including gedit, emacs, and vi. The simplest one to handle is probably gedit.
Interactive python shell:
To interactively run python scripts `ipython` is available on the workshop computers.

Part I: Basic electronic structure with FHI-aims

Problem I: 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. 1. 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. 2. 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. Now, run FHI-aims:
   ```
   aims.x | tee output
   ```

   One 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   :  -0.136054753597673E+02  eV
   | Total energy corrected     :  -0.136054753597673E+02  eV
   | Electronic free energy     :  -0.136054753597673E+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
   ```

4. 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 the command
   ```
   xmgrace results.dat
   ```

Optional: 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

Optional part: 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: Hydrofluoric acid (HF): bond length and dipole moment

Hydrofluoric acid (HF)

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, atomization energy (ΔH_{at}), and the dipole moment for hydrogen fluoride (HF) with two methods. From a technical perspective, this exercise teaches how simple shell scripting can be used to make your (computational) life easier.

Eductational Objectives:

- Become acquainted with shell scripting.
- Learn how to compute dipoles and atomic charges.
- Find the equilibrium bond distance of a simple diatomic molecule.
- Assess the reliability of charge partition schemes.

Tasks:

1. The first task of this exercise will be to find the equilibrium bond distance of HF from a series of calculations. Start by creating a `geometry.template` file which contains a H and a F atom, as shown in Fig. 4.

```
#HF at variable bond distances
atom 0.0 0.0 0.0 H
atom 0.0 0.0 Dist F
```

Figure 4: The geometry data for a HF molecule. H is put on the origin and F is located `Dist` Å away from the origin along the z-axis.

   In this example, H is put on the origin and F is located `Dist` Å away from the origin along the z-axis. Hereby, `Dist` is a placeholder which will be replaced by the actual distance later using a script.

2. Create a `control.template` file, and specify a `hf` calculation for a neutral system. Feel free to copy contents from `control.in` file in the previous exercise, but set *spin* to none by commenting the `spin` and `default_initial_moment` lines. Please remove the *really tight* species setting for hydrogen and paste the *tight* setting for H **and** F into the `control.template` file. We also want to compute the electronic dipole moment, which is requested using the keyword `output dipole`. Note that although it doesn't matter where you put the keyword, it is recommended to put all manually set keywords *before* the species tags.

3. Next, create a bash script which runs FHI-aims for a series of bond distances (ideally between 0.7Å and 1.3Å with 0.1Å steps, and a denser step width of 0.02 Å between 0.85 Å and 0.95 Å).

   For each distance, it 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.

   Appendix I holds a short sample python and bash script (fig. 8 and 9) for this task. Alternatively, you may also use the scripts we provide in

   `$HandsOn/tutorial_1/skel/exercise_02_HF/templates/run.py` and
   `$HandsOn/tutorial_1/skel/exercise_02_HF/templates/Run.sh`

   Run the script (`python run.py` or `./Run.sh`) and plot the total energies vs. the bond length. This can be scripted as well. Example scripts are also provided in

   `$HandsOn/tutorial_1/skel/exercise_02_HF/templates/PP_Energies.py` and
   `$HandsOn/tutorial_1/skel/exercise_02_HF/templates/PP_Energies.sh`.

   Which bond length corresponds to the lowest energy? How does the bond length compare to the experimental bond length of 0.917Å?

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 and F atoms. Compute the total energies for the single atoms using the methods `hf`. Check the result carefully - is Hund's rule fulfilled? Does the electronic configuration correspond to what you expect from the aufbau principle?

   Note:
   Atoms are highly symmetric systems, often with multiple degenerate solutions. In the case of fluorine, for example, the unpaired p-electron might sit in the p_x, p_y, or p_z orbital. All three solutions are equivalent. If the calculation is started unbiased, it might converge to a superposition of these three cases, which is a saddle point on the potential energy surface and results in partial electron occupations. Although in DFT non-integer occupations are in principle allowed, one should be very suspicious when obtaining such a solution for non-metallic systems. Typically, solutions exist that are lower in energy. They can be found by breaking the inherent symmetry of the problem, for example by applying a small external field at the beginning of the SCF cycle.

   To break the inherent symmetry of an atom and ensure integer occupation, set the keyword `switch_external_pert 10 safe`. This means that for 10 iterations, a small external field in the z-direction is applied and then switched off. Usually, this is sufficient to perturb the SCF out of the symmetric solution and towards the correct electronic structure. Calculate the atomization energy (ΔH_{at}) of HF by subtracting the free-atom energies from the predicted total energy of HF (i.e. the minimum total energy found when varying bond distances).

   $$\Delta H_{at} = E^{HF}_{tot} - E^{H}_{atom} - E^{F}_{atom} \tag{1}$$

   How does this compare to the experimental value of $\Delta H_{at} = 135.2\text{kcal mol}^{-1}$ (5.86 eV)?

5. Now, let us look at the dipole moment. Search for the corresponding line in the output file. Does the dipole differ much between the two methods used? How does the dipole at the equilibrium distance compare with the experimental value of 1.82 Debye? Plot the dipole moment vs. the bond distance. You will find a (mostly) linear correspondence. Do you expect this trend to continue at large distances? Why or why not?

6. **Optional:** Repeat the bond length determination using `pbe0`. How does the optimal bond length change? How much does the total energy change? Are the dipole moments the same?

Optional: Charge partition schemes.

Chemical reactivity and many physical properties are often explained in terms of atomic charges. However, atomic charges are not physical observables, since no unique operator exists to determine this quantity. They rather depend on the chosen charge partition scheme. The charge partition schemes that is probably most commonly used is Mulliken [2]. In FHI-aims, you can request it by specifying `output mulliken` in `control.in`.

For the equilibrium structure of HF, compute the atomic charges with this scheme using `pbe0`. Use the charges to calculate the dipole moment μ in the point dipole approximation. In this approximation, for a two atom system, the dipole moment is defined as:

$$\mu = q \cdot |\vec{r_H} - \vec{r_F}| \tag{2}$$

where q is the atomic partial charge and $\vec{r_H}$ and $\vec{r_F}$ are the atomic positions of the atoms. The absolute value of the difference $|\vec{r_H} - \vec{r_F}|$ is the distance between hydrogen and fluorine. Compare the dipole moment to the one computed by FHI-aims. How do they compare?

Problem III: Molecular oxygen - a critical look !

An important part of every calculation is to always look critically at the output and ensure that the result is reasonable. For some systems, defaults may not be adequate, or assumptions which commonly work well may prove to be wrong. A prime example is the the treatment of spin in systems with degenerate orbitals, such as O_2.

[**Educational Objectives:**]

- Do not trust default settings blindly
- See the effect of an incorrect spin-treatment

Tasks

1. Set up a calculation for O_2 similar to the previous exercise, but leave out the `spin` keyword in `control.in`. Look at the output file to find out which spin treatment is used by default.
2. Look at the output file and find the occupation numbers of the Kohn-Sham orbitals. Does this make sense here?
3. Using `pbe` and `pbe0`, calculate the binding curve in the interval $[0.8, 1.6]$ Å with a stepwidth of 0.1 Å as before. Also, calculate the atomization energy (ΔH_{at}).
4. Repeat the calculation with `spin collinear` and `default_initial_moment 2.0` - is Hund's rule now fulfilled? Compare the results: Do both spin settings yield the same equilibrium bond length? Calculate the difference in the total energy at the equilibrium bond length. Which one is lower? How does it compare to the experimental value of 1.21 Å? How does the atomization energy (ΔH_{at}) compare to the experimental value of 5.18 eV?

Note: The oxygen atom is a notoriously difficult case to converge. If you have problems with it, try using the linear mixer for the SCF by including the keyword `mixer linear` in `control.in`. The linear mixer is guaranteed to converge, but usually requires much more iterations than the default (Pulay) mixing scheme.

Part II: Local structure optimisation

Problem IV: Planar H_3O^+

This exercise covers how to perform geometry optimizations. Specifically, we will relax the H_3O^+ molecule starting from an initial planar guess for the geometry.

Educational Objectives

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

1. Fig. 5 shows a geometry.in file, which can also be found in `$HandsOn/tutorial_1/skel/exercise_04_H3O+_planar/templates`. Please use this as the starting point for your structure relaxation.

```
# Initial guess for planar H3O+
atom   0.00    0.00    0.00    O
atom   0.92   -0.53    0.00    H
atom  -0.92   -0.53    0.00    H
atom   0.00    1.06    0.00    H
```

Figure 5: This geometry.in file gives a starting point for the planar H_3O^+ cation.

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 and, as we are interested in a cation, charge 1.0. 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. 6.

```
# Sample input file for the relaxation
# of the H3O+ molecule
#######################################
  xc  pw-lda
  charge 1.0
  spin none

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

#######Relaxation######################
  relax_geometry trm 1E-3

#######Species#########################
################################################################################
#
#  FHI-aims code project
```

Figure 6: The control.in file used for the structure relaxation of the cation H_3O^+.

As in the exercises before, the basis set must be included in the control.in file. Use the "*light*" species defaults for O and H atoms.

```
cat $SPECIES_DEFAULTS/light/01_H_default  >> control.in
cat $SPECIES_DEFAULTS/light/08_O_default  >> control.in
```

3. Run FHI-aims.
`aims.x | tee H3O+_planar_relaxation.out`

4. To visualize the results, copy the tool create_relax_movie.py from `$HandsOn/tutorial_1/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 H3O+_planar_relaxation.out > H3O+.xyz`

Open the file `H3O+.xyz` with a visualizer, e.g. jmol.
`jmol H3O+.xyz`
(also see jmol tutorial video $HandsOn/tutorial_1/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.

What does the fully relaxed structure look like? Do you believe that this could be the actual total energy minimum?

Problem V: A vibrational analysis of planar H_3O^+ !

Now we will check whether our obtained geometry is a local minimum or saddle point by performing a vibrational analysis.

Educational Objectives

- Learn to calculate vibrations in FHI-aims
- Recognize transition states

We will use the python script `get_vibrations.py` in a folder that contains a control.in and geometry.in file. The script is located in `$HandsOn/tutorial_1/utilities`.

1. Use the same `control.in` as in Problem IV. Next create `geometry.in` file by copying the optimized geometry from the previous exercise. This should be contained in the file `geometry.in.next_step`.
2. Run the vibrational script.
   ```
   >python get_vibrations.py -r aims.x -x -p -m -w -d 0.002 H3O+_planar 2 >
   H3O+_planar.vib.out
   ```
   This will create folders containing the single calculations and calculate the vibrational frequencies. For information about the python script and its options type:
   ```
   >python get_vibrations.py -h
   ```
3. Inspect the `H3O+_planar.vib.out` file.

The script `get_vibrations.py` produces several output files - most importantly the `H3O+_planar.vib.out` and the `H3O+_planar.xyz` file. The `H3O+_planar.vib.out` file contains basic information regarding vibrational frequencies, zero point energies and infrared (IR) intensities. In addition, the `H3O+_planar.xyz` file contains geometrical information about the vibrational modes.

If the Hessian (the matrix of second derivatives of total energy with respect to atomic positions) is positive definite (i.e., there are only positive eigenvalues), the structure is in a local minimum. If not we, found a saddle point or a maximum. Formally, in this case the Hessian will have imaginary vibrational frequencies. However it is also common to output them as negative numbers. There are 3N-6 non-zero modes (3N-5 for linear molecules), where N is the number of atoms. Due to numerical errors you will typically find the remaining 6 (5) 'zero' frequencies to be in the range of $\sim \pm 15$ cm^{-1}. They may be negative, but as their values are numerical errors and not physical, these zero modes should be ignored when decided if the structure is in a local minimum.

To visualize the eigenmodes of the molecule you can use e. g. jmol: `> jmol H3O+_planar.xyz`

In this example, H_3O^+ has one imaginary frequency. What does the corresponding mode look like (see jmol tutorial video $HandsOn/tutorial_1/jmol_tutorial.ogv)?

Problem VI: Getting H_3O^+ right

An imaginary frequency indicates that the curvature of the potential energy surface (PES) along that mode is negative. In other words, following that mode leads to a structure with lower energy. Therefore, a reasonable guess would be to distort the geometry along this direction and optimize the structure again. This is what we are going to do next!

Educational Objectives:

- Use the vibrational analysis of transistion states to find the nearest energetic minimum structure

1. Take the `H3O+_planar.distorted.vibration_1.geometry.in` file from Problem V. It contains the planar geometry distorted along the imaginary mode.
2. For the distorted geometry, optimize the structure again (as described in Problem IV).
   `> aims.x | tee aims.H3O+_distorted_relaxation.out`

What does the optimized structure look like?
How does the energy compare to the planar optimized structure of Problem IV?

Problem VII: Pyramidal H_3O^+

Check whether the 3D geometry of H_3O^+ is stable by performing a vibrational analysis as described in Problem V.

Tasks:

- Perform a vibrational analysis of the pyramidal H_3O^+ cation.
- Is it stable?

Problem VIII: IR spectrum

Besides information about the structural stability, a vibrational calculation also gives the IR spectrum. In this exercise we will compare the IR spectrum for the planar and pyramidal geometry for different numerical settings and different functionals.

Educational Objective

- Visualize the IR-spectrum of the H_3O^+ cation
- Compare experimental and calculated results.

To get started, we can use the data calculated in Problem V and Problem VII. Compare the vibrational frequencies obtained in both problems.

- The IR-spectrum is avaiable as *pdf* in your calculation folder
- Repeat the same thing for the vibrational analysis of the planar structure.

Note:
As you see, the IR-spectrum is very sensitive to the structure. Comparing a calculated IR-spectrum to an experimental one can shed light on the structure that is predominantly present in experiment.

Optional: Convergence of numerical settings and performance of different methods.

- Perform convergence tests with respect to light, tight and really tight settings for the infrared spectrum and for the 3D geometry (pw-lda functional).

  Important:
  Since a harmonic IR-spectrum only makes sense for an optimized geometry, make sure that you request a geometry optimization in your control.in file, i.e. set `relax_geometry trm 1E-3`.
  If set, the vibrational script optimizes the geometry before doing the actual calculation for the vibrations.

- For light settings, compute the PBE IR spectrum and compare it with the pw-lda spectrum.
- Compare your obtained spectrum with the experimentally obtained vibration frequencies. You can look them up online at the NIST webbook:
  http://webbook.nist.gov/cgi/cbook.cgi?ID=C13968086&Mask=800#Electronic-Spec

Part III: Electron density mixing and visualizing electron densities and eigenstates

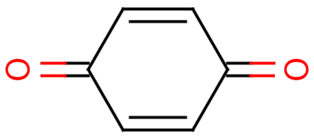
Problem IX: Visualization of Kohn-Sham orbitals and electron densities

Chemical reactions and many important physical effects (such as the formation of interface dipoles) are triggered by electron rearrangement. They are often associated with filling or emptying of molecular frontier orbitals. In this exercise, we will demonstrate how charge distributions are visualised and investigate how well the "actual" filling of a molecular orbital corresponds to the associated eigenstate of the neutral molecule.

Educational Objectives

- Visualization of results
- See how orbitals (wave functions) and charge-density differences compare
- Understand the difference between orbital eigenvalues and ionization energies

1. The molecule for the present example is para-benzoquinone, a readily available small molecule that is a strong electron acceptor. You can either create the geometry input for this molecule yourself (see jmol tutorial video $HandsOn/tutorial_1/jmol_tutorial.ogv) or use the provided input file from the Templates folder. With jmol you can save your structure by typing
`write "benzoquinone.xyz" as xyz`
in the jmol console (see jmol tutorial video $HandsOn/tutorial_1/jmol_tutorial.ogv).



2. Create a control.in input file for a spin-unpolarized, uncharged PBE calculation of the molecule. We want to visualize the highest occupied molecule orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO) in form of a standard cube file. Request the output of the wave functions using the keywords
```
output cube eigenstate homo
cube filename HOMO.cube
output cube eigenstate lumo
cube filename LUMO.cube
```
We also want to look at charge density differences, so we need the total density of the molecule. Add
```
output cube total_density
cube filename total_density_uncharged.cube
```
to the `control.in` file. By default, FHI-aims will figure out the region to plot into the cube files by itself.

However, you are also free to choose the cube region yourself by adding the following lines after the `output cube` keyword:
The line
`cube origin` *x y z*
sets the center of the space to be plotted, with x y z being its cartesian coordinates in Å. The edges of the cube file can be configured using the syntax
`cube edge` *n dx dy dz*,
where n indicates the number of steps of a particular edge (voxel), and dx, dy, dz indicate the length of each individual step in x, y, and z direction, respectively. Note that each cube file, being a 3D object, requires 3 edges to be completely defined. A complete definition of a cube file could look like
```
output cube total_density
```

```
cube filename total_density_uncharged.cube
cube origin 0 0 0
cube edge 100 0.1 0.0 0.0
cube edge 150 0.0 0.1 0.0
cube edge 50 0.0 0.0 0.1
```

3. Run the calculation and look at the output. In the final occupation numbers, search for the energies of the HOMO and the LUMO. How do these energies compare to the experimental values of 10.7 eV for the ionisation potential and 1.9 eV for the electron affinity? Next, use your favourite tool to plot the cube files. (See appendix on how this is done with jmol.) Do HOMO and LUMO both have π character?

4. To see how well the eigenvalues and eigenstates correspond to the physical situation of charging a molecule, prepare and run two more inputs for the molecule with one negative and and one positive charge. Remember that the total number of electrons is odd, and therefore, `spin collinear` and `default_initial_moment 1.0` should be used.

   Obtain the electron addition and removal energies by subtracting the total energy of the charged systems from the total energy of the uncharged system. This is also known as the ΔSCF approach. Compare the values with the HOMO and LUMO eigenvalue of the neutral molecule. How much do they differ?

5. The change in the electron density upon charging (positive and negative) can be obtained by subtracting the total density of the charged system from the total density of the uncharged system. To that end, a python-script called 'subtract_cubes.py' is provided in the utilities folder. Copy this script and the total_density.cube files into a separate folder and execute
   ```
   > python subtract_cubes.py total_density_uncharged.cube
   total_density_charged.cube chargediff.cube
   ```
   to subtract the file given as second argument from the file given as first argument (adjust the file names accordingly). The result will be written to the file provided as final argument. Note that the cube files need to be generated with the same cube file settings in `control.in`. Visualise the density differences. Do you see qualitative differences to the eigenstates? Can you explain (or at least speculate) on the reasons for them?

Problem X: Converging efficiently

Italian chef Antonello Colonna said in an interview with an American newspaper: "Garlic is the king of the kitchen, to eliminate it is like eliminating violins from an orchestra."
The garlic bulb releases a number of low-molecular-weight organosulfur compounds rarely encountered in nature [3]. One of them is the molecule Allicin (see Fig. 7). Allicin is responsible for the characteristic smell of freshly crunched garlic.
In this part of the exercise, we will use this molecule to compare the influence of different electron density mixing settings.

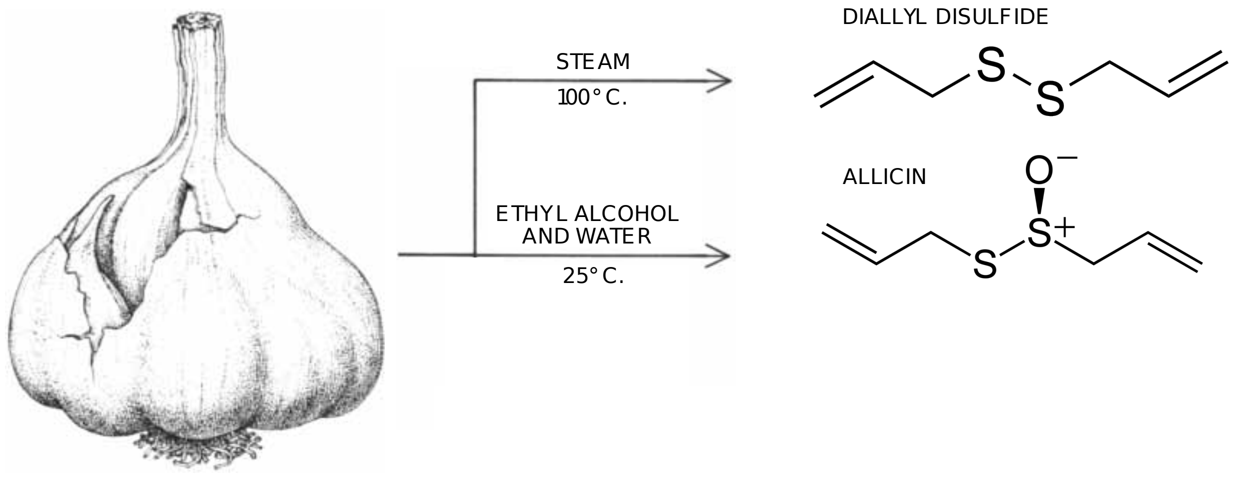


Figure 7: The allicin molecule features a thiosulfinate functional group (R-S(O)-S-R), which gives freshly crunched garlic its unique smell. (Figure adapted from Block [3])

During the self-consistent solution of the calculation, it is beneficial to use a history of densities from previous SCF cycles. FHI-aims provides two mainstream density mixing algorithms, linear and Pulay (also known as Direct Inversion in Iterative Subspace, DIIS).

Educational Objectives

- Obtain a first impression of the performance of different mixing schemes and parameters.

The geometry of the molecule for this exercise, as well as a sample `control.in` file, can be found in the `$HandsOn/tutorial_1/skel/exercise_10_Allicin_mixer/templates` directory.

1. First create a directory for a calculation using the `linear` mixer. In the template `control.in` file you will find three new keywords:

   ```
   mixer TYPE
   charge_mix_param PARA
   n_max_pulay NUM
   ```

   For now ignore the keywords `charge_mix_param` and `n_max_pulay` – add a hash (#) in front of these two lines to deactivate them. The `mixer` keyword specifies the electron density mixing algorithm used. Here we use the *t*ype `linear`, so replace the place holder `TYPE` by `linear`.

2. Run the calculation and open the FHI-aims output file. How many SCF cycles were needed to reach self-consistency? Do you think this many SCF cycles are necessary to converge a rather small molecule?

3. Next we will use a more advanced mixing scheme, the `Pulay` mixer.
   This mixer also takes into account the history of the self-consistent calculation. For this algorithm, `n_max_pulay` *n* and `charge_mix_param` *v* have to be specified, n determines the number of past iterations $\mu - k$ ($k = 1, \cdots, n$) to be mixed with the Kohn-Sham output density of iteration μ. The value v for the keyword `charge_mix_param` determines an additional (system-dependent) linear factor that is multiplied with the output density change of the Pulay mixer.
   We will try three different settings:
   - `mixer pulay`
     `n_max_pulay 3`
     `charge_mix_param 0.1`
   - Use the FHI-aims default settings:
     `mixer pulay`
     `n_max_pulay 8`
     `charge_mix_param 0.2`
   - Finally use the settings specifically optimised for Allicin.
     `mixer pulay`
     `n_max_pulay 5`
     `charge_mix_param 0.3`

4. Compare the number of SCF cycles and computational time with the linear mixer. Did the choice of mixer speed up your calculation? How do the optimized settings compare to the previous calculations?

This exercise can be nicely scripted by automatically replacing the place holder `TYPE, PARA, NUM`. Have a look at the example script in the Appendix I: Sample bash scripts.

We like to thank you for participating in the first tutorial of this workshop. We wish you an interesting and inspiring time here at the 'First-principles simulations of molecules and materials' workshop.

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.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.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 8: 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.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.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 9: This is a sample bash script for calculating a molecule with different pre-defined bond distances.

Sample script for Exercise 10

```python
import os, shutil
#Variable defintion:
AIMSBIN='aims.x'
##############################################################
MIXER=['linear', 'pulay']
N_MAX=[3, 5, 8]
PARAM=[0.1, 0.3, 0.2]

print "****************************************************"
print "EXERCISE␣X:"
print "****************************************************"


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

for typ in MIXER:
  if not os.path.exists(typ): os.mkdir(typ)
  os.chdir(typ)

  if typ != 'linear' :
     for num in N_MAX:
        if num == 3:
           para=PARAM[0]
        elif num == 5 :
           para=PARAM[1]
        elif num == 8 :
           para=PARAM[2]
        if not os.path.exists(str(num-para)): os.mkdir(str(num-para))
        os.chdir(str(num-para))
        shutil.copy('../../geometry_template.in', 'geometry.in')
        out=open('control.in','w')      # replace the term TYPE in control
                                        # template by the current mixer
        template_out=template.replace('TYPE',typ)
        # set paramters
        template_out=template_out.replace('PARA',str(para))
        template_out=template_out.replace('NUM',str(num))
        out.write(template_out)# and copy the new file into the directory
        out.close()
        os.system(AIMSBIN+' | tee output') # Run aims and pipe the output
                                            #into a file named "output"
        os.chdir('..')
  else:
     shutil.copy('../geometry_template.in', 'geometry.in')
     out=open('control.in','w')    # replace the term TYPE in control
                                   # template by the current mixer
     template_out=template.replace('TYPE',typ) # set paramters
     template_out=template_out.replace('charge_mix_param',\
                                  '#charge_mix_param')
     template_out=template_out.replace('n_max_pulay','#n_max_pulay')
     out.write(template_out)# and copy the new file into the directory
     out.close()
     os.system(AIMSBIN+' | tee output') # Run Aims
  os.chdir('..')
```

Figure 10: 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/ ..
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
```

Acknowledgments

We like to thank testers of this tutorials, Sheng Bi, Lixin Lu and many others for their active support and invaluable feedback. Also we would like to thank Carsten Baldauf and the other workshop organisers.

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).

[2] R. S. Mulliken, *Electronic Population Analysis on LCAO-MO Molecular Wave Functions. II. Overlap Populations, Bond Orders and Covalent Bond Energies*, The Journal of Chemical Physics **23**, 1841 (1955).

[3] E. Block, *The Chemistry of Garlic and Onions*, Scientific American **252**, 114 (1985).