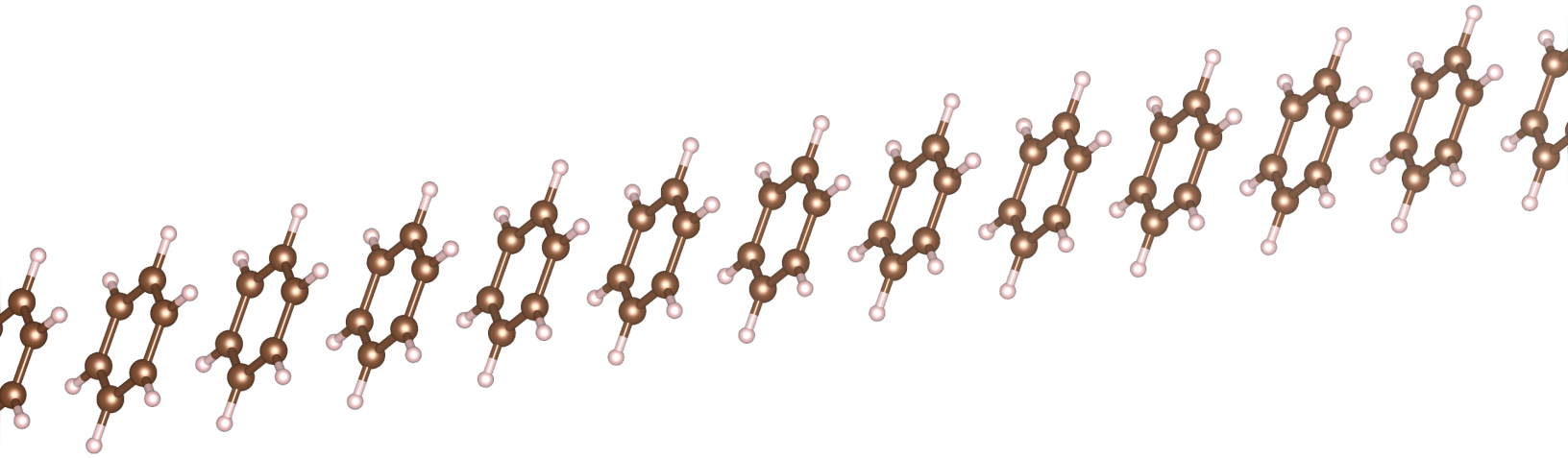
Density functional theory and beyond: Computational materials science for real materials

Los Angeles, July 21–August 1, 2014



Tutorial 3: van der Waals Interactions
Manuscript for Exercise Problems

Prepared by Jan Hermann and Alexandre Tkatchenko

Fritz-Haber-Institut der Max-Planck-Gesellschaft

Los Angeles, July 21, 2014

Contents

A short introduction

Now that you have learned how to run basic DFT calculations for molecular and periodic systems, we will now show you how you can calculate intermolecular interactions with FHI-aims and what different options you have.

Both previous tutorials employed gradient-corrected or hybrid exchange–correlation functionals. These might work very well for covalent, ionic and metallic bonds and to some degree even for intermolecular interactions, but they certainly miss van der Waals interactions. These are often relatively weak but always attractive, purely quantum forces which exist between every two (or more) fragments of matter and thus become important for large and realistic systems.

There are many methods which treat van der Waals interactions within DFT. The simplest approach is to divide the molecular system into atomic fragments and employ a pairwise approximation. This leads to expressions of the form

$$\sum_{i<j} f(R_{ij}) \frac{C_{6,ij}}{R_{ij}^6} \tag{1}$$

where the sum runs over pairs of atoms i, j, R is the distance between said atoms and C_6 is a coefficient describing the strength of the interaction. f is a so-called damping function which is an empirical entity preventing R^{-6} from diverging and ensuring a smooth coupling of the van der Waals energy to the DFT energy.

In the simplest variant, tabulated empirical C_6 coefficients for individual atoms are used and f has a simple analytical form (DFT-Dn series [1, 2]). In a more elaborated scheme, the $C_6[n]$ coefficients are functionals of the electron density n and thus take into account the local chemical environment (Tkatchenko–Scheffler [3]). In a yet more complex scheme, the C_6 coefficients may be derived from the Kohn–Sham orbitals (Becke–Johnson [4]).

A related, but alternative approach to van der Waals interactions is the use of non-local density functionals of the form

$$\iint n(\mathbf{r}_1)\Phi[n](\mathbf{r}_1,\mathbf{r}_2)n(\mathbf{r}_2)\mathrm{d}\mathbf{r}_1\mathrm{d}\mathbf{r}_2 \tag{2}$$

where Φ is the so-called non-local kernel. That is also the quantity which distinguishes different non-local functionals. The two most widely adopted families are vdW-DF [5] and Vydrov–van Voorhis [6].

The non-local functionals share with the pairwise method that they are essentially two-body, or additive for long distances. But this is known to be not valid for van der Waals interactions. There are again several ways how to go beyond pairwise approximations. The simplest approach is to extend the pairwise methods by adding three-body and higher-body terms. This goes in the spirit of adding higher-order terms in perturbation theory. A more satisfactory approach which had been gaining popularity recently is the random-phase approximation (RPA) used on top of a DFT calculation. It is general, all-purpose but it's major disadvantage is a higher computational cost. A way around this is to use RPA with approximate response functions, which is a strategy of the Many-Body Dispersion method. ("dispersion" is an alternative name for van der Waals interactions more common in the chemical community.)

None of the methods mentioned above are exact and in this tutorial, we will limit ourselves to the Tkatchenko–Scheffler (TS) method [3] and the Many-Body Dispersion (MBD) [7, 8] method as representatives of the pairwise and many-body classes. They are both implemented in FHI-aims and well established. The TS method was already briefly described. The idea of MBD is to approximate response of atoms to electromagnetic field (input for RPA) by harmonic dipole oscillators which can then be solved exactly. The integral quantity in this model is the dynamic polarizability $\alpha(\omega)$ which is related to C_6 of the TS method by the formula,

$$C_{6,ij} = \frac{3}{\pi}\int_0^\infty \alpha_i(\mathrm{i}\omega)\alpha_j(\mathrm{i}\omega)\mathrm{d}\omega \tag{3}$$

If you are interested, you can find more details in Appendix A.

Finally, mind that both TS and MBD methods have some shortcomings. Apart from the self-evident absence of many-body effects in the pairwise TS method, both methods can suffer from the fact that they are formulated in terms of atoms. Hence in systems where the partitioning into atoms is problematic, such as in metals (delocalized electrons) or ionic systems (charge transfer), these methods might give inaccurate or even qualitatively wrong results. The obvious remedy might be to turn at least partly from atoms to electron density (cf. non-local functionals), and this branch of unification is under development.

Exercises

We have prepared nine small problems divided in three parts which we hope will illustrate various aspects of van der Waals interactions. The first part deals with a benzene dimer which has been one of the widely studied van der Waals systems. The relevant FHI-aims keywords and output are introduced and we will dig a bit more into the inner workings of the methods. The second part introduces van der Waals interactions in periodic systems, still taking advantage of the benzene molecule. The third part dabs lightly into a more recent topic, showcasing some calculations with graphene.

All the files related to this tutorial can be found in

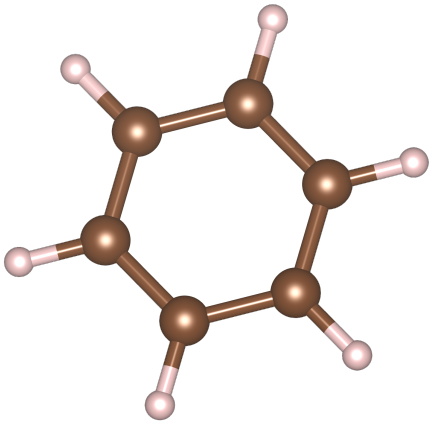
```
$HANDSON/hands-on-2014-tutorials/tutorial_3
```

It is organized as follows. In `doc`, you can find this text (`tutorial3.pdf`) and also the part of FHI-aims documentation related to MBD (`doc-aims-mbd.pdf`). `problems` contains individual directory for each problem. In each of them except for the first one, you can find Python script `run.py` to run the relevant calculations and `extract.py` to gather and analyze results. These scripts were prepared beforehand as most of our problems comprise a series of individual FHI-aims calculations. During the tutorial, we will modify some of these scripts and other files. Finally, `utilities` contains scripts used by `run.py` and `extract.py`. Namely, `run_aims.sh` automatically creates `RUN.[n]` directory, copies input files in it, adds basis set information (`add_species.py`) and runs FHI-aims. `geom.py` defines classes `Atom`, `Molecule` and `Crystal` which will enables us easy creation and manipulation of. . . atoms, molecules and crystals. `parse_aims.py` can be used to parse FHI-aims output and load it into Python and finally `pretty_table.py` serves for printing pretty tables.

1 A study of benzene dimer

The benzene dimer has served as a test case for van der Waals methods since the 1980s and it took over two decades to finally resolve its potential energy surface (PES). Particularly challenging is the description of almost degenerate T-shaped (T) and parallel-displaced (PD) structures.

Problem 1: Single benzene molecule



Task Find Hirshfeld volume and charge of carbon and hydrogen in a benzene molecule. Compare C_6 coefficients calculated from fully screened and short-range screened polarizability. Plot dynamic polarizabilities along the principal axes of the polarizability tensor. Check that the molecular C_6 coefficient in FHI-aims is correct.

The core `control.in` file which we will use looks like

```
xc                 pbe
sc_accuracy_rho    1e-5
sc_accuracy_etot   1e-6
```

That is we use a standard PBE functional and reasonable SCF convergence criteria. So go ahead, and create this file. To start with van der Waals, add a line consisting of only one keyword

```
vdw_correction_hirshfeld
```

This activates the evaluation of the TS method.

Next, we create the geometry file for a benzene molecule. We will do it by first creating a −CH element and then rotating it five times by 60°. We can do that easily in Octave, an open-source clone of MATLAB.[1] So start it with

```
$ octave
```

and enter the following commands

```
> rCC = 1.397
> rCH = 1.084
> CH = [rCC 0 0; rCC+rCH 0 0] # sets a 2-by-3 matrix
> rot = [cosd(60) sind(60) 0; -sind(60) cosd(60) 0; 0 0 1]
```

[1] With a little bit of careful programming, one can write programs that run both in Octave and MATLAB.

```
> benzene = CH;
> for i = 1:5
>     CH = CH*rot; # matrix multiplication of CH and rot
>     benzene = [benzene; CH]; # appends rows of CH to rows of benzene
> end
> disp(benzene) # short for display
```

This will display the xyz coordinates for a benzene molecule where the C and H atoms are alternating. Exit from Octave (`Ctrl+D`) and save it to some file. We have to convert it into the FHI-aims's `geometry.in` which has the format of

```
atom [x] [y] [z] [element]
```

Feel free to do it as you wish or use the following `awk` script,

```
$ awk '{print "atom",$0,(i++%2==0)?"C":"H"}' [file]
```

You can check the validity of the generated file by viewing it in `jmol`.

Finally, append the basis set information to the control file, for example with

```
$ cat $HANDSON/aimsfiles/species_defaults/light/*_{C,H}_* >> control.in
```

You can run FHI-aims now with

```
$ mpirun -n 4 [aims executable] > aims.out
```

After the calculation is finished (a couple of seconds), you can find the output of FHI-aims in `aims.out`. The TS method is run after the SCF cycle is converged as is typical for van der Waals methods. So either search for "Hirshfeld analysis" or skip close the end of the file.

```
Evaluating non-empirical van der Waals correction (Tkatchenko/Scheffler 2009).
Performing Hirshfeld analysis of fragment charges and moments.
------------------------------------------------------------------------
| Atom    1: C
```

Let us pause here for a moment and say a tiny bit more about the TS method. We already mentioned that the C_6 are calculated from the electron density and for it to be possible, the density has to be first divided between individual atoms. This can be done in many ways and what we use is the so-called Hirshfeld partitioning (details in Appendix A). Once having "atomic densities", all sorts of quantities can be calculated. Note the Hirshfeld charge (partial charge) showing the flow of electrons from hydrogen to carbon. A more important quantity for us is the Hirshfeld volume, because the core idea of the TS method is to obtain C_6 in Eq. (1) as

$$C_6^{\text{eff}} = C_6^{\text{free}} \left(\frac{V_{\text{Hirshfeld}}}{V_{\text{free}}} \right)^2 \tag{4}$$

where C_6^{eff} is the effective atomic C_6 in a molecule and C_6^{free} is C_6 of a free atom. You can see that in the case of a benzene molecule, atomic C_6 will be smaller than their free-atom counterparts. Lower in the output, you can find total energy and its various components. Having a single molecule though, we are not really interested in the van der Waals energy yet.

In the next step, replace `vdw_correction_hirshfeld` in the control file with

```
many_body_dispersion
```

This activates the MBD method. Note that only one of these keywords should be used at the same time. Run FHI-aims again and check the output.

By Eq. 3, we need to know the dynamic polarizability to calculate C_6. This happens automatically in FHI-aims, but being a key quantity in the van der Waals theory, you can find the dynamic polarizability listed in the output,

```
| Many-Body Dispersion (MBD@rsSCS) energy
| Dynamic molecular polarizability alpha(iw) (bohr^3)
| ------------------------------------------------------------------------------
|  omega(Ha)  alpha_iso     alpha_xx      alpha_yy      alpha_zz
|   0.000000  0.674230E+02  0.363629E+02  0.829508E+02  0.829554E+02
|   0.039290  0.670600E+02  0.361629E+02  0.825062E+02  0.825108E+02
```

Dynamic polarizability describes the response of a molecule in a similar same way as static polarizability does, but for an oscillating electric field,

$$\mathbf{p}(\omega) = \boldsymbol{\alpha}(\omega)\mathbf{E}(\omega)$$

where ω is the frequency, E is the generating field, p is the induced dipole moment and α is the dynamic polarizability which is in general a tensor. As light is just an oscillating electric field from the point of view of many processes, the knowledge of dynamic polarizability is enough to calculate many interesting matter–light interaction properties. On the other hand, van der Waals energy can be viewed as coming from fluctuations of these dipoles, which can be described by the same molecular polarizability. These two phenomena, inducing and fluctuating of the electric dipole in matter can be nicely connected by the so-called fluctuation-dissipation theorem.

In the output, the isotropic and principal components of $\boldsymbol{\alpha}(\omega)$ are listed. In case of the benzene molecule oriented as we have it, the principal axes coincide with the Cartesian axes. What does "rsSCS" in MBD@rsSCS mean? After constructing the model atomic dipole oscillators and before solving the Hamiltonian for them, there is one additional step, called screening. This adjusts the atomic polarizabilities such that the influence of neighboring atoms is taken into account. Furthermore, the polarizabilities need to be screened only for the short-range part of the Coulomb potential (see Appendix A for details). "rsSCS" thus stands for "range-separated self-consistent screening". The

reason for this labeling is that there are also other flavors of MBD, briefly mentioned below.

Right under that, there is

```
  | Molecular C6 coefficient        :        1963.12847354    hartree bohr^6
  | ---------------------------------------------------------------------
  | Partitioned C6 (hartree bohr^6) | alpha (bohr^3) of atom in molecule
  | ---------------------------------------------------------------------
  | ATOM   1  C   35.452249        9.852572
  | ATOM   2  H    2.047265        2.345397
```

which states the total molecular C_6 coefficients and partitioned C_6 and polarizabilities. "Paritioned" here does not mean actual partitioning of the molecular C_6 coefficient, but rather rescaling of free atom values using information from the Hirshfeld partitioning (see Appendix A for details).

Underneath, you can find the total and van der Waals energy,

```
  Total energy components:
[...]
  | MBD@rsSCS energy           :         -0.00789257 Ha        -0.21476777 eV
[...]
  | Total energy               :       -232.03427706 Ha     -6313.97393032 eV
```

There are two similar sections above for MBD@SCS and MBD@TS. These are older versions which have problems in some situations for some materials, whereas MBD@rsSCS is currently the most up-to-date version and we will be using just that in what follows.[2] The info about energies for the two older MBD versions deserves a couple of comments,

```
  | Pairwise TS energy               :       -0.002520321 Ha          -0.068581415 eV
  | Pairwise TS+SCS energy           :       -0.002755232 Ha          -0.074973687 eV
  | MBD@SCS energy                   :       -0.011341600 Ha          -0.308620633 eV
  | Total energy + TS energy         :     -232.028904843 Ha       -6313.827744810 eV
  | Total energy + TS+SCS energy     :     -232.029139755 Ha       -6313.834137082 eV
  | Total energy + MBD@SCS energy    :     -232.037726123 Ha       -6314.067784027 eV
```

The TS here is exactly the same as we would got with `vdw_correction_hirshfeld` so from now, we will use just the `many_body_dispersion` keyword. TS+SCS is the TS method evaluated with the C_6 coefficients obtained from the (fully) screened polarizabilities. Finally, what is meant by total energy *here* is the bare DFT energy (PBE in our case) *before* the final MBD@rsSCS correction.

[2] MBD@SCS is done with polarizabilities screened with the full Coulomb potential (no range-separation) and MBD@TE with polarizabilities without any screening. In MBD@SCS, the long-range screening is effectively double-counted, first in the actual screening, second in solving the coupled Hamiltonian. This is taken care of in MBD@rsSCS

Let us turn back to the dynamic polarizability. First isolate the listing of the polarizability (under `omega(Ha)`) and save it in `omega.txt`. To get rid of the leading pipe symbols in the polarizability output, you can use

```
$ sed -i 's/|//g' omega.txt
```

Then start Octave again and use the following commands to display the graph of dynamic polarizabilities.

```
> load omega.txt
> alpha = omega(:, 2:5);
> omega = omega(:, 1);
> plot(omega, alpha(:, 2:3));
> title("dynamic polarizablity")
> xlabel("frequency (a.u.)")
> ylabel("polarizability (a.u.)")
> legend("alpha_{xx}", "alpha_{yy}")
```

Notice three things. First, the polarizability decreases with frequency as the electrons are not able to respond fast enough to rapid electromagnetic oscillations. Second, there is a large difference between the out-of-plane (`alpha_xx`) and in-plane (`alpha_yy`) polarization.[3] This would not be the case without the polarizability screening. Third, at higher frequencies, the anisotropy disappears. This is because the screening ceases to be effective.

The last thing to do is to calculate the molecular C_6 coefficient from the dynamic polarizability using Eq. (3), where $\int \mathrm{d}\omega \to \sum_i w_i$. The integration weights are not listed in the output, but they can be found in the source code of FHI-aims. You can can the molecular C6 coefficient in Octave with (the same session as for the plotting)

```
> load int_weight.txt # load integration weight
> C6 = 3/pi*sum(alpha(:, 1).^2.*int_weight)
> # .^ and .* stand for elementwise operations
```

Compare the result with the output from FHI-aims.

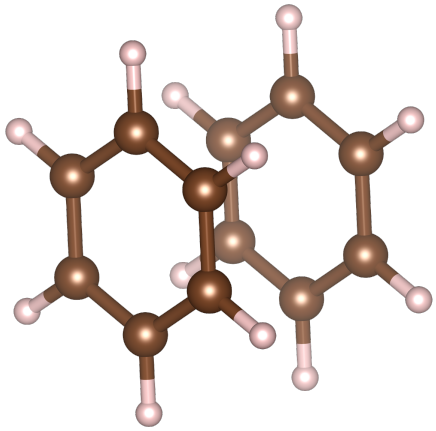
Problem 2: Stacked interaction

Task For the stacked configuration of benzene dimer, find the total energy minimum along the intermolecular distance. Compare the predictions of PBE, PBE+TS and PBE+MBD with a reference method.

[3] The axis labels do not correspond here to the Cartesian axes, because they are derived as the principal axes of the polarizability tensor, independently of the actual geometry.

The interaction curve of two molecules A and B is

$$E_{\mathrm{int}}(r) = E(AB, |AB| = r) - E(A) - E(B)$$



where $E_{\mathrm{int}}(r)$ is the interaction energy of molecules at a distance r and E are total energies of the specified systems. In the case of a benzene dimer, $A = B$, so we will need to run FHI-aims for a single benzene molecule and then for a series of benzene dimers at different separations. To save time and not to bore you, we will provide Python scripts for running and analysis beginning with this exercise. But before we use the scripts, let us briefly walk you through them.

The control file in this exercise is identical to the one we ended up with in the last exercise, but without the basis set information. This will be added by the script `run.py`, so take a look at it. The script first loads some modules and parses command-line arguments. In the next step, it builds a benzene molecule in the same manner as you did in the previous exercise, just using Python and the `geom` module.

```
rCC, rCH = 1.397, 1.084 # set C-C and C-H distances
CH = geom.Molecule([geom.Atom("C", (rCC, 0, 0)),
                    geom.Atom("H", (rCC+rCH, 0, 0))]) # create a -CH fragment
benzene = geom.Molecule() # initiate an empty molecule
for angle in np.arange(0, 360, 60): # for each angle in {0, 60,.. 300}
    benzene.atoms.extend(CH.rotate("z", angle, center=(0, 0, 0)).atoms)
    # rotate the -CH fragment and add it to the benzene molecule
```

The it loops over different benzene–benzene distances (0 for a single benzene molecule),

```
for dist in [0, 3.2, 3.4, 3.6, 3.8, 4, 4.4, 4.8, 5.6]:
    if dist == 0:
        m = benzene
    else:
        m = benzene.join(benzene.shift((0, 0, dist)))
    # run AIMS for a geometry specified by m...
```

A directory is created for each of them, the geometry is printed into a file,

```
with open("geometry.in", "w") as f:
    print >>f, benzene.to_str("aims") # convert to AIMS format
```

and finally if the `--dry` option was not given, FHI-aims is run using a `light` basis set on a number of processors given as an argument to `run.py`. The `run_aims.sh` script takes care of the rest. So do it!

```
$ ./run.py 4
```

After a short time, you can start the analysis by checking the generated geometries in `RUN.1/*/RUN.1/geometry.in` with `jmol`. The results can be evaluated with the `extract.py` script. First, it gathers the data,

```
results = []
for rundir in glob.glob("*"): # for all directories
    # get distance from a dir name
    dist, = re.match(r"([\d\.]+)", rundir).groups()
    dist = float(dist) # convert from string to float
    output = os.path.join(rundir, "RUN.1", "aims.out") # AIMS output file
    with open(output) as f: # read and parse the output
        results.append((dist, parse_aims.AIMS_output(f).results))
results = sorted(results) # sort by distance
```

Next, it calculates the interaction energies for all methods we want,

```
methods = ["DFT energy", "Total energy + TS energy", "Total energy + MBD@rsSCS energy"]
energies = {method: [] for method in methods}
for dist, data in results:
    if dist == 0: # if monomer, save the data in a special variable
        momoner = {method: data["Energies"][method] for method in methods}
    else:
        for method in methods: # calculate interaction energy for each method
            energies[method].append(data["Energies"][method]-2*momoner[method])
```

And finally it plots the results, together with a reference coupled cluster (CCSD(T)) value,

```
for method in methods:
    plt.plot(dists, energies[method]) # plot each method
plt.plot([3.87], [-1.71], "o") # add a point for CCSD(T) value
plt.legend(methods + ["CCSD(T)"], numpoints=1) # make legend
```

You can find the generated figure in `plot.pdf` You can see that while PBE gives no binding at all, both van der Waals methods predict interaction. And while TS overbinds by roughly by 1 kcal/mol, MBD gives very accurate energy. This should not be surprising, given how big the many-body effects are regarding the polarizability of a benzene molecule (Problem 1). On the other hand, we used only a light basis set, so you should check the converged PBE+MBD energy using `tight` setting.

Problem 3: Investigation of the benzene dimer PES

Task Calculate energies of 10 given stationary points on the benzene dimer potential energy surface (PES). Compare different methods.

The stationary point geometries are given in `bz-dimer.geoms.txt` as labeled xyz files

```
M1
24

C      1.205074   -0.695750    0.000000
# ...
H     -2.140382   -0.474604    3.488382
------------------------------------------------------------
M2
24
# ...
```

and we can extract them easily with

```
geoms = {}
with open("bz-dimer.geoms.txt") as f:
    geom_strings = re.split(r"\n-+\n", f.read()) # split on --- lines
    for s in geom_strings:
        label, xyz = s.split("\n", 1) # split first line with a label
        # parse xyz geometry
        geoms[label.strip()] = geom.Molecule().from_str(xyz, fmt="xyz")
```

After running FHI-aims for the geometries with `run.py`, we can read the benchmark energies in `bz-dimer.energies.csv` with

```
benchmark = {}
with open("bz-dimer.energies.csv") as f:
    csvreader = csv.DictReader(f)
    for row in csvreader:
        benchmark[row["label"]] = float(row["energy"])
```

The rest of `extract.py` is then simply calculating the interaction energies as in Problem 2 and constructing a `prettytable` table and printing it. You can see which geometries correspond to which labels by using `jmol`.

You can again see that PBE severely underbinds, while PBE+TS overbinds. But rather than this general trend, a more stringent test is available here. The hardest problem in describing the benzene dimer PES is the comparison of T-shaped (T)

and parallel-displaced (PD) structures (see [9] for pictures). They should be almost degenerate (< 0.1 kcal/mol), but you can see that PBE+MBD gets this correctly.

2 An investigation of periodic benzene systems

In this part, we will be still busy with the benzene molecule, but this time in infinitely extended arrangements, namely a benzene crystal and various benzene chains.

Problem 4: Benzene crystal

Task Calculate lattice energy of the benzene crystal using different van der Waals methods and compare with the experimental value [10]. Test convergence of the MBD energy with the MBD supercell size.

Periodic calculations of van der Waals energy do not differ too much from non-periodic in FHI-aims. As with ordinary DFT calculations, we have to add the information about k-point sampling to `control.in`. We use a template for the control file, with a line

```
{k_grid:}
```

to be replaced from within `run.py`. A second thing is the specification of the so called MBD supercell. Currently, the MBD energy in a periodic cell is calculated by replicating the unit cell, evaluating MBD for this supercell and averaging over all the replicas (details in Appendix A). Naturally, the energy should be converged with the supercell size. We can modify its size by specifying

```
mbd_supercell_cutoff  15.
```

which means that only unit cell replica within 15 Å from the origin are used. The default is 25 Å and we are using a lower value to shorten the running time.

The structures of a benzene crystal and molecule are in `benzene-crystal.aims` and `benzene.xyz` (you can check also crystal structures with `jmol`). The unit cell contains 4 benzene molecules. We can read them both with

```
with open("benzene.xyz") as f:
    molecule = geom.Molecule.from_str(f.read(), fmt="xyz")
with open("benzene-crystal.aims") as f:
    crystal = geom.Crystal.from_str(f.read(), fmt="aims")
```

The actual `control.in` can be created from the template by

```
with open("control.in", "w") as f:
    print >>f, control.format(k_grid="k_grid 2 2 2"
                                  if label == "crystal"
                                  else "")
```

Run the prepared `run.py`.

After FHI-aims is finished, get the results with `extract.py`. You can once again see that PBE gives you almost no binding, PBE+TS overbinds and PBE+MBD gives a cohesive energy of benzene crystal in very good agreement with experiment.

Try increase the value of `mbd_supercell_cutoff` and run again to see how well is the energy converged. Do not forget to analyze the new results in `RUN.2` with

```
./extract.py --dir RUN.2
```

Problem 5: Benzene chain

Task Calculate van der Waals energy per molecule of an infinite chain of stacked benzene molecules depending on the benzene–benzene separation. Compare with the binding of a benzene dimer.

A benzene chain is a 1-dimensional periodic system, but FHI-aims is capable of either non-periodic or 3-dimensional periodic calculations. To remedy this issue, we add a repetition in the other two dimensions too, but make the separation so large that there will be practically no interaction between the parallel chains. The MBD routine has to know about this by specifying

```
mbd_scs_vacuum_axis   .true. .true. .false.
```

in the control file. This has the meaning that the system is actually not periodic along the x and y axes. (The chain will go along the z axis.)

Similarly to the case of the benzene dimer, we can generate all necessary calculations with

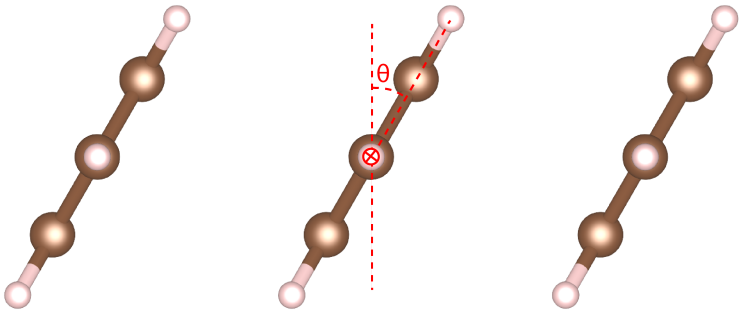
```
for dist in [0, 3.2, 3.4, 3.6, 3.8, 4., 4.2, 4.6, 5, 5.4]:
    if dist == 0:
        m = benzene
    else:
        m = geom.Crystal.from_molecule(benzene, [100, 100, dist])
```

where the parallel chains will be separated by 100 Å.

After `run.py` and `extract.py`, you can look at the generated `plot.pdf`. Surprisingly, the binding is almost the same as with the benzene dimer. This is caused by the fact that in a 1D system, the amount of matter at a distance r is independent of r (as opposed to $\sim r^2$ in a 3D crystal), while the interaction falls off quickly as r^{-6}. The two neighboring molecules thus constitute a vast majority of the interaction.

Problem 6: Benzene chain reloaded *

This problem has a relatively long running time. You might consider skipping it for now and returning to it later if you have some time left.



Task Consider the problem of tilting the molecules in a benzene chain. Find out the dependence of the binding on the tilt angle θ.

We can extend `run.py` from the previous problem and add another loop over the tilt angles. We also want to ensure that the atoms in two neighboring benzene molecules will not get too close in the process of tilting, so we want to add a constant offset to the separation distances for each angle.

```
offset = {0: 0, 30: 0.3, 60: 1.6, 90: 3}
for angle in [0, 30, 60, 90]:
    for dist in [0, 3.4, 3.8, 4.4]:
        if dist == 0:
            if angle > 0: # run only for dist == 0 and angle == 0
                continue
            m = benzene
        else:
            dist += offset[angle]
            m = geom.Crystal.from_molecule(benzene.rotate("x", angle),
                                           [100, 100, dist])
```

`run.py` and `extract.py` later, you can see the curves in `plot.pdf`. Notice several things. First, binding is strongest for 30°, in agreement with the fact that the parallel-displaced configuration of a benzene dimer binds stronger than the stacked configuration. Second, the electrostatic component of the binding energy (PBE) is strongest for 60°. Third, the difference between TS and MBD is quite big for 0° and 30°, but almost vanishes for larger tilts. This illustrate the dependence of the many-body effects on the exact geometry of the studied system.

3 An inquiry into graphene interactions

The recent graphene boom took place after the experiments of Geim and Novoselov in 2004. In 2010, they were awarded the Noble price for their work. Graphene is being studied from all possible angles and we will deal here with interactions of graphene with other entities.

Problem 7: Graphene and benzene

Task Calculate the interaction curve of a single benzene molecule on a graphene sheet using PBE, PBE+TS and PBE+MBE methods. Compare with experimental value.

Graphene is a 2D-periodic material, but graphene with a single adsorbed benzene molecule is a non-periodic infinite system. To model it within a 3D-periodic code, we again introduce artificial periodicity. We replicate the graphene sheet in perpendicular direction, with enough vacuum in between the parallel sheets. And instead of a single benzene molecule, we will have its 2D-periodic lattice, with the individual molecule far enough from each other so there is only a negligible interaction between them.

We can generate the graphene surface quite simply, by first creating the graphene unit cell and then replicating it in the in-plane directions to create supercells to host the individual benzene molecules.

```
rCC = 1.420 # C-C distance
a_uc = rCC*np.sqrt(3) # lattice constant a of the hexagonal lattice
sc_size = 5 # number of unit cell replicas in each direction
a_sc = sc_size*a_uc # lattice constant of the supercell
interlayer = 100. # vacuum thickness in between parallel graphene sheets
graphene = geom.Crystal([geom.Atom("C", (0, 0, 0)),
                         geom.Atom("C", (1./3, 1./3, 0))],
                        hexa=(a_uc, a_uc, interlayer),
                        coord_type="d") # fractional (direct) coordinates
graphene.set_coord_type("c") # convert to cartesian coordinates
graphene = graphene.copy_cell((sc_size, sc_size, 1)) # replicate unit cell
```

First definitely check the generated geometry, so that you are sure what is going on. Then, the `plot.pdf` file you will get from `plot.pdf` shows that the PBE+TS method compared with the experimental value overestimates the binding energy by $\sim 5\,\mathrm{kcal}$ whereas PBE+MBD is in a very good agreement.

Problem 8: Graphene bilayer

Task Calculate the dependence of the binding energy of a graphene bilayer using PBE, PBE+TS and PBE+MBD methods. Test the convergence of the MBD energy

with the site of the MBD supercell.

With a graphene bilayer without any adsorbed molecule, we can restrict the calculation back to the bare graphene unit cell, using proper k-point sampling. The geometries can be generated by

```
for dist in [0, 2.8, 3.4, 3.7, 4, 5, 7]:
    if dist == 0: # graphene sheet monomer
        m = graphene
    else:
        second_layer = graphene.copy()
        for atom in second_layer.atoms:
            atom.R[2] += dist # shift the second graphene sheet
        m = graphene.copy()
        m.atoms.extend(second_layer.atoms) # add the two sheets together
```

Obtain the `plot.pdf` figure. Here you can see a significantly different binding curve compared to the benzene dimer, with much more slowly decaying tail. This is owing to the dimensionality of the materials and it can be shown that in case of 2D–2D interaction, the van der Waals interaction decays as r^{-4}. In this case, PBE+MBD slightly underbinds compared to the experimental value. But this might be due to too small MBD supercell cutoff. Try running the same calculation for a series of `mbd_supercell_cutoff` $\in \{10, 15, 25, 35\}$ and see how the MBD energy converges.

Problem 9: Graphene multilayer

Task Calculate the dependence of the binding energy per graphene sheet of a slab of multiple graphene sheets at a constant separation distance. Use PBE, PBE+TS and PBE+MBD methods.

Building upon the previous exercise scripts, we can build the graphene multilayer simply with

```
layer_sep = 3.5 # separation of graphene sheets
for n_layer in range(1, 9):
    if n_layer == 1: # single layer
        m = graphene.copy()
        add_layer = graphene.copy() # initialize the add layer
    else:
        for atom in add_layer.atoms:
            atom.R[2] += layer_sep
        m.atoms.extend(add_layer.copy().atoms)
```

Run the exercise and open the `plot.pdf` file. The binding energies are calculated with

```
kcal*(data["Energies"][method]-n*monomer[method])/n
```

where `n` is the number of the sheets.

There are several things to notice. First, PBE gives no binding at all. This serves as an example that common GGA functionals can perform poorly even for bulk matter. In the limit of infinite number of graphene layers, we get graphite and PBE would predict that graphite is not a stable material. Second, the interaction converges quite slowly, which is to be compared with the infinite benzene chain, where we argued that only the nearest neighbors are significantly contributing. Third, mind the almost two-fold overestimation of the binding energy by TS for the eight layers. Finally, notice when looking at the tails, that TS converges noticeably slower than MBD, owning to the compensation by depolarization in case of MBD.

References

[1] S. Grimme, *Accurate description of van der Waals complexes by density functional theory including empirical corrections*, Journal of computational chemistry **25**, 1463 (2004).

[2] S. Grimme, J. Antony, S. Ehrlich and H. Krieg, *A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu*, The Journal of Chemical Physics **132**, 154104 (2010).

[3] A. Tkatchenko and M. Scheffler, *Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data*, Physical Review Letters **102**, 073005 (2009).

[4] A. D. Becke and E. R. Johnson, *A density-functional model of the dispersion interaction*, The Journal of Chemical Physics **123**, 154101 (2005).

[5] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth and B. I. Lundqvist, *Van der Waals Density Functional for General Geometries*, Physical Review Letters **92**, 246401 (2004).

[6] O. A. Vydrov and T. Van Voorhis, *Nonlocal van der Waals density functional: The simpler the better*, The Journal of chemical physics **133**, 244103 (2010).

[7] A. Ambrosetti, A. M. Reilly, R. A. D. Jr and A. Tkatchenko, *Long-range correlation energy calculated from coupled atomic response functions*, The Journal of Chemical Physics **140**, 18A508 (2014).

[8] R. A. DiStasio, Jr., V. V. Gobre and A. Tkatchenko, *Many-body van der Waals interactions in molecules and condensed matter*, Journal of Physics: Condensed Matter **26**, 213202 (2014).

[9] O. Bludský, M. Rubeš, P. Soldán, P. Nachtigall and M. Rubes, *Investigation of the benzene-dimer potential energy surface: DFT/CCSD(T) correction scheme*, The Journal of chemical physics **128**, 114102 (2008).

[10] A. M. Reilly and A. Tkatchenko, *Understanding the role of vibrations, exact exchange, and many-body van der Waals interactions in the cohesive properties of molecular crystals*, The Journal of Chemical Physics **139**, 024705 (2013).

A Theoretical background

Here, we would like to present a brief description of the Many-Body Dispersion (MBD) method. The procedure starts with projecting the response of valence electrons onto a set of quantum harmonic oscillators by calculating the "volumes" for free atoms and atoms in a molecule,

$$V_i = \int n_i(\mathbf{r})|\mathbf{r} - \mathbf{R}_i|^3 \mathrm{d}\mathbf{r}$$

where $\mathbf{R}_i$ is the position of atom i and n_i is either its free-atom electron density n_i^{free} or Hirshfeld density n_i^{H},

$$n_i^{\mathrm{H}}(\mathbf{r}) = n(\mathbf{r}) \frac{n_i^{\mathrm{free}}(\mathbf{r})}{\sum_j n_j^{\mathrm{free}}(\mathbf{r})}$$

The ratio of the free and Hirshfeld volumes captures the influence of chemical environment on the individual atoms. To incorporate this effect in the van der Waals interaction, the free atom reference static polarizabilities α_0, C_6 coefficients and van der Waals radii R^{vdW} are renormalized using the volume ratios,

$$\begin{aligned}
\alpha_{0,i} &= \alpha_{0,i}^{\mathrm{free}} \times V_i^{\mathrm{H}}/V_i^{\mathrm{free}} \\
C_{6,i} &= C_{6,i}^{\mathrm{free}} \times (V_i^{\mathrm{H}}/V_i^{\mathrm{free}})^2 \\
R_i^{\mathrm{vdW}} &= R_i^{\mathrm{vdW,free}} \times (V_i^{\mathrm{H}}/V_i^{\mathrm{free}})^{1/3}
\end{aligned}$$

Then, the dynamic polarizabilities are calculated as

$$\alpha_i(\omega) = \alpha_{0,i} \left(1 - \frac{\omega^2}{(\omega_i^{\mathrm{eff}})^2}\right)^{-1}$$

where the effective resonance frequency ω_i^{eff} is defined as

$$\omega_i^{\mathrm{eff}} = \frac{4C_{6,i}}{3\alpha_{0,i}^2}$$

This corresponds to approximating the response of an atom with a single harmonic oscillator.

Screening In the next step, we evaluate the screening of polarizabilities, i. e. how response properties of the atoms are influenced by other atoms. The scalar polarizabilities are first recast as tensors, $\boldsymbol{\alpha}_i = \boldsymbol{\delta}\alpha_i$ ($\boldsymbol{\delta}$ for Kronecker delta) and gathered into one $3N$-by-$3N$ matrix, where N is the number of atoms, $\boldsymbol{\alpha} = \bigoplus_i \boldsymbol{\alpha}_i$. This is the bare polarizability tensor $\boldsymbol{\alpha}$. Next, we want to let the atoms interact through dipole interaction,

$$\mathbf{T}_{\text{dipole}} = \boldsymbol{\nabla} \otimes \boldsymbol{\nabla} \frac{1}{r}$$

To explicitly account for the fact that we do not have point dipoles, but rather atoms where the dipole density is spread over a finite volume, we consider the Coulomb interaction between two Gaussian charge densities, $v_{GG}(r) = \operatorname{erf}(r/\sigma)/r$. Furthermore, we want to perform the screening only with the short-range part of the Coulomb potential. This leads to the final interaction tensor between atoms i and j of the form

$$\mathbf{T}_{ij}^{GG,\text{sr}} = (1 - f(R_{ij}))\, \boldsymbol{\nabla} \otimes \boldsymbol{\nabla} \frac{\operatorname{erf}(r/\sigma_{ij})}{r}$$

where $\sigma_{ij} = \sqrt{\sigma_i^2 + \sigma_j^2}$, $\sigma_i = \left(\sqrt{2/\pi} \times \alpha_i/3\right)^{1/3}$ defines the natural width of the interacting charge densities, f is a damping function and R_{ij} is the distance between the atoms. The short-range screened polarizability tensor $\boldsymbol{\alpha}^{\text{sr}}$ is then obtained using the self-consistent Dyson-like equation,

$$\boldsymbol{\alpha}^{\text{sr}} = \boldsymbol{\alpha} + \boldsymbol{\alpha}\mathbf{T}^{GG,\text{sr}}\boldsymbol{\alpha}^{\text{sr}}$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{11} & \dots & \mathbf{T}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{T}_{N1} & \dots & \mathbf{T}_{NN} \end{bmatrix}$$

The isotropic atomic screened polarizabilities are recovered as

$$\alpha_i^{\text{sr}}(\omega) = \frac{1}{3} \operatorname{Tr}\left[\sum_j \boldsymbol{\alpha}_{ij}^{\text{sr}}(\omega)\right]$$

RPA Finally in the last step, the MBD correlation energy is calculated from the short-range screened polarizabilities (response functions) using the random-phase approximation (RPA) formula,

$$E_{\text{c}} = \frac{1}{2\pi} \int_0^\infty \operatorname{Tr}[\ln(1 - \chi_0 v) + \chi_0 v] \mathrm{d}\omega$$

where in case of MBD, the bare response function χ_0 is $\bigoplus_i \boldsymbol{\delta}\alpha_i^{\mathrm{sr}}$ and the interaction potential v is ("lr" for long-range)

$$\mathbf{T}^{\mathrm{lr}} \sim \mathbf{T}_{ij}^{\mathrm{lr}} = f(R_{ij})\boldsymbol{\nabla} \otimes \boldsymbol{\nabla}\frac{1}{r}$$

A significant convenience of MBD is that because of the use of dipole approximation and other considerations, the RPA integral can be evaluated analytically by recasting the problem as matrix diagonalization.

Periodic MBD

The MBD method could be of course made periodic using Bloch functions and k-point sampling, but we employ a less technically demanding solution.

In the screening part, the effect of atoms from other unit cells is incorporated in the interaction tensor $\mathbf{T}$ by

$$\mathbf{T}_{ij} = \sum_k \mathbf{T}_{ik}, \qquad k \in \{\text{replicas of } j, |jk| < R_{\mathrm{cutoff}}\}$$

In FHI-aims, R_{cutoff} can be controlled with the keyword `mbd_scs_dip_cutoff`.

In the RPA part, the modification is two-fold. First, the interaction tensor is modified in the same way as for screening, just using a different keyword `mbd_cfdm_dip_cutoff`. Second, the original unit cell is replicated into a supercell, the RPA integral is evaluated for the whole supercell as if it was a molecular cluster, and the MBD energy is averaged over all unit cells that constitute the supercell. The size of the supercell is controlled with `mbd_supercell_cutoff`. Mind that you should always check the convergence of MBD energy with all these three parameters in production periodic calculations.