Exercise 4: Calculating Absorption Spectra using QM/MM Methods

Gustavo Cardenas
Online Course on Simulation of Biological Systems

February 24, 2022

1 Introduction

In this last hands-on session of the course we will compute the absorption spectrum of the azobenzene molecule (that's right, the same molecule whose isomerization you studied on exercise 2). This perhaps will be a shorter hands-on session, as we will not run brand new molecular dynamics simulations (which, at this point, you may have gotten acquainted with), but instead we will use the simulations we have performed on exercise 2, and run single point QM/MM calculations on an ensemble of geometries drawn from those previous simulations. Now, what *is* the absorption spectrum of a molecule? In short terms, it consists of the electronic transitions from the Ground State of a molecule towards different singlet Excited States (provide the Ground State itself is a singlet). It is represented as the intensity of absorption as a function of the energy involved in each electronic transition. Now, the first approach to simulating an absorption spectrum is by taking the equilibrium geometry of the molecule and computing its *vertical* electronic excitations towards the excited states (see Figure 1 a). This generates a "bar" spectrum (Figure 1 c); however, in real life what we obtain from a measurement is not a bar for each transition, but instead a *band*. There are several reasons for the broadening of an electronic transition, but perhaps the most important one is that in reality molecules move, and each possible configuration can generate the same electronic transitions as those of the equilibrium geometry, but not having the exact same energies, hence the broadening (see Figure 1 b,d). What we are doing today is precisely to compute the spectrum on different geometries, in order to account for their contributions to the overall absorptions spectrum.

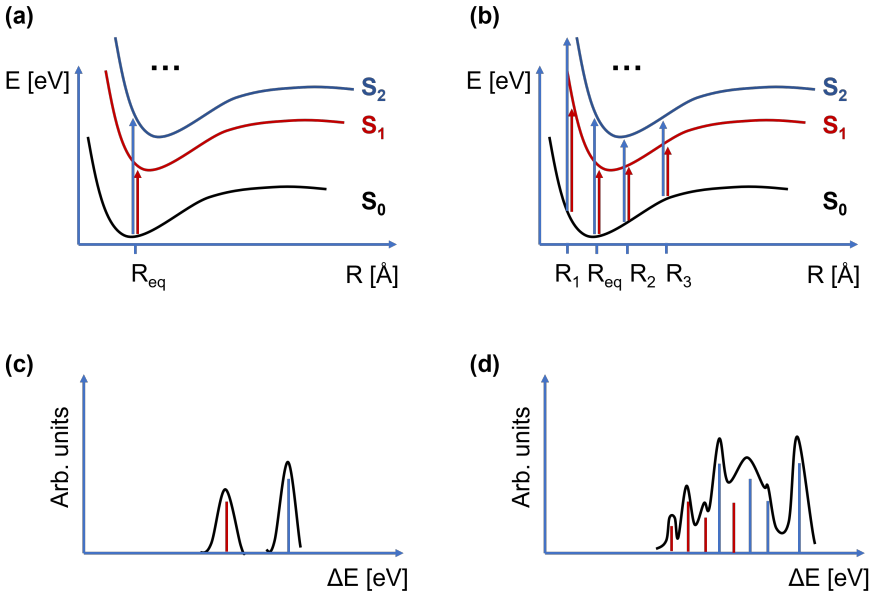

Figure 1: (a) Vertical electronic transitions to the S1 and S2 excited states at the equilibrium geometry. (b) Vertical electronic transitions to the S1 and S2 excited states for an ensemble of geometries. (c) Resulting absorption spectrum from the excitations computed at the equilibrium geometry. (d) Absorption spectrum computed considering an ensemble of geometries.

2 Setup and Tasks

2.1 Architecture of the Folders

Figure 2 contains the architecture of today's tutorial. It will be subdivided in two parts:

- Part 1 (1_ensemble): Compute the absorption spectrum on different geometries to obtain the overall spectral lineshape
- Part 2 (2_qm_size): We will assess the effect of adding different water molecules to the calculation of the spectrum. For consistency (and simplicity) we will consider only one geometry.

In this exercise we will use the AmberTools19 package, the Orca quantum chemistry program, and MoBioTools. Instead of xtb, we will use another package developed by the group of Stefan Grimme, sTDA, which is a semiempirical approximation to Time Dependent DFT. Of course we are using it since it is computationally cheap, so we will have our results in a reasonable time. **If you already have orca, you don't need to install sTDA!**. To execute the scripts to visualize the spectra, we will also need the matplotlib and the tkinter python packags. If you downloaded Anaconda, you should already have them.

At this point, please copy the 4_SPECTRA folder on your working directory (or otherwise follow the folder architecture in 2), then go to your version of the 4_SPECTRA folder.

```
cd 4_SPECTRA
```

This is where the tutorial begins!

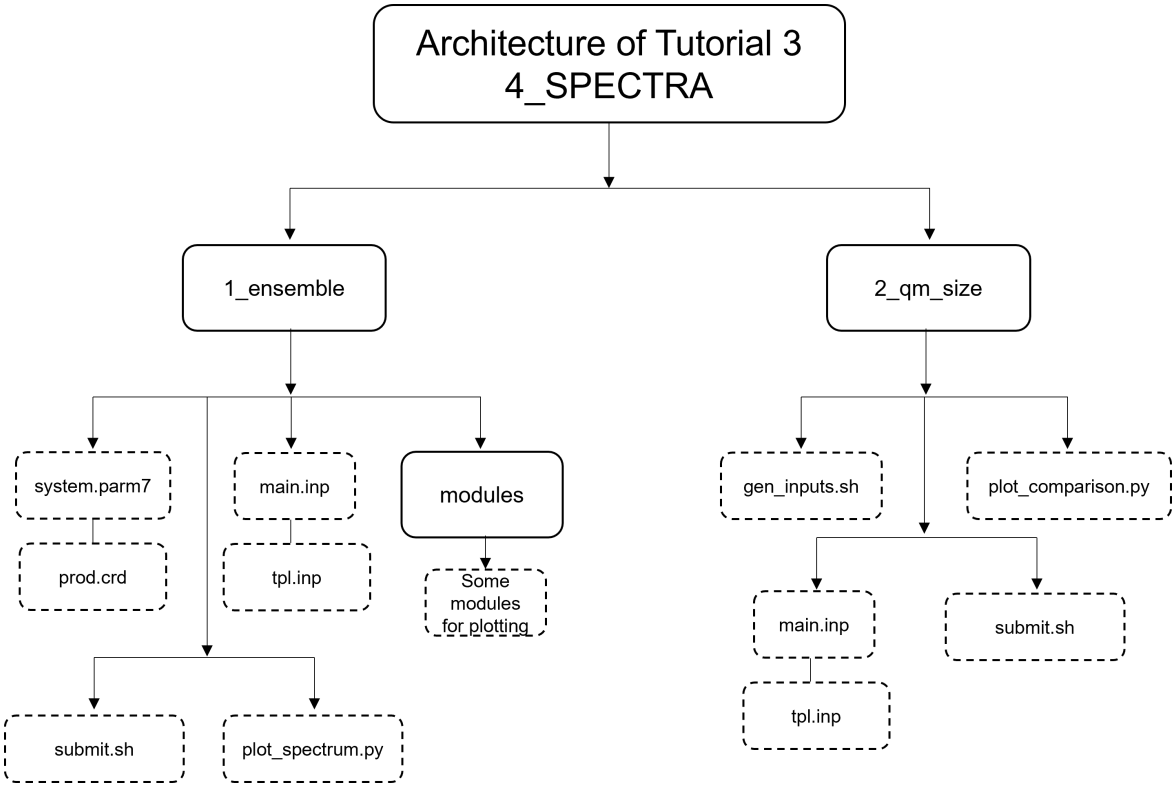

Figure 2: Architecture of exercise 4

2.2 Part 1: Absorption Spectrum from an Ensemble of Geometries

We will generate an input file for each of the geometries of interest using the `MoBioTools` package. In this exercise, we will use the production trajectory of azobenzene from exercise 2, which consists of 200 geometries/snapshots. However, due to time constraints, we will only consider 10 geometries, which will nonetheless provide us with a good idea of the importance of sampling for simulating absorption spectra.

Please go to the 1_ensemble folder:

```
cd 1_ensemble
```

there you will find the files associated with the MD trajectory of interest, `system.parm7` and `prod.crd`, and the two necessary inputs for MoBioTools, `main.inp` and `tpl.inp`. Remember that in order to perform Single Point calculations on top of snapshots of a given MD trajectory, **we will first need to image them to the primitive cell**, as we no longer have Periodic Boundary Conditions. We will use cpptraj, as usual:

```
parm system.parm7
trajin prod.crd
autoimage mask :1
trajout image.crd
go
quit
```

Once we have our imaged trajectory, we will fetch 10 geometries out of it and generate an input for each one of them. Let us take a look at the two necessary files for MoBioTools:

```
main.inp

&main
tpl   = orca
top   = system.parm7
traj  = imaged.crd
qmmask = :1
geoms = 0 9
&end
```

Of course, you can choose whatever range of geometries you prefer. The template file to generate the `orca` input files is the following:

```
tpl.inp

&header
 !pbe def2-svp
&end

&externchg
&end
```

```
&chgspin
 0,1
&end

&tddft
 mode sTDA
 ethresh 5.0
&end
```

Now this is where things start getting interesting. First of all, notice that we are no longer using XTB four calculation; instead, we will use Density Functional Theory (DFT) for which we need to define an exchange-correlation functional (`pbe` in our case), and we will also specify the basis set of functions we want to use, in our case `def2-svp`. These settings will determine how we will obtain the Ground State of our system. We already know what the `&externchg` and `&shgspin` stand for. The `&tddft` section will enable the TD-DFT calculation in orca. The `mode sTDA` keyword states that we want to use the sTDA approximation, and the `ethresh 5.0` indicates a maximum energy threshold of 5 eV, so that no electronic transitions above it be computed. Now, let us generate the desired input files by calling the `main_qminputs.py` script:

```
main_qminputs.py -i main.inp -t tpl.inp
```

This will generate one folder for each of the requested geometries. At this point, we can go, for example, to geom6

```
cd geom6
```

and visualize the input file it has generated:

```
!pbe def2-svp

* xyz 0 1
C       17.426000       16.956000       12.205000
C       16.690000       15.927000       12.677000
C       16.318000       15.982000       14.107000
C       16.723000       17.065000       14.898000
N       16.339000       17.040000       16.261000
N       17.219000       17.007000       17.149000
C       16.808000       16.672000       18.496000
C       16.095000       17.561000       19.410000
C       15.668000       17.143000       20.661000
C       15.641000       15.815000       20.912000
C       16.352000       14.868000       20.103000
C       16.926000       15.326000       18.908000
C       17.522000       18.098000       14.331000
C       17.753000       18.036000       12.947000
H       17.558000       17.005000       11.128000
```

```
H       16.347000       15.105000       12.055000
H       15.627000       15.285000       14.573000
H       15.921000       18.566000       19.036000
H       15.083000       17.801000       21.297000
H       15.254000       15.491000       21.874000
H       16.294000       13.841000       20.451000
H       17.443000       14.630000       18.255000
H       17.873000       18.960000       14.891000
H       18.201000       18.894000       12.455000
*

%pointcharges "charges_geom6.xyz"

%tddft
mode sTDA
ethresh 5.0
end
```

It basically reports the settings we requested in the tpl.inp file, with the addition of the xyz coordinates of geometry 6 and the corresponding external point charges (present in the `charges_geom6.xyz`), that do not belong to the QM region.

At this point we can proceed with the actual calculations. Please go back to the 1_ensemble folder

```
cd ..
```

where you will find the `submit.sh` submission script:

```
#!/bin/bash

for i in {0..9}
do
    cd geom"$i"
    orca geom"$i".inp > geom"$i".inp.out
    cd ..
done
```

Execute it as

```
nohup ./submit.sh &
```

This will submit the `orca` calculations sequentially. For 10 geometries it will take overall 3 to 4 minutes. Once the calculations are done, let us visualize any of the output files to look for the information we are interested in. For example, we can open the output of geom0 with vi

```
vi geom0/geom0.inp.out
```

The string we are interested in reads `"ABSORPTION SPECTRUM VIA TRANSITION ELECTRIC DIPOLE MOMENTS"`. We can search it with slash. It will report a table with the index of the

state (indexing starts with 0), the transition energies (in cm-1 and in nm) and the corresponding oscillator strengths (intensities). Below is a section of the table, containing the relevant informations for us.

ABSORPTION SPECTRUM VIA ...

State	Energy (cm−1)	Wavelength (nm)	fosc	...
0	16114.2	620.6	0.035645391	...
1	23107.8	432.8	0.003628343	...
2	24404.2	409.8	0.028877629	...
3	25025.0	399.6	0.050092679	...
4	30576.1	327.1	0.009894089	...
5	31204.3	320.5	0.000640855	...
6	31508.8	317.4	0.000565140	...
7	31998.8	312.5	0.017636388	...
8	34543.5	289.5	0.240851756	...
9	34833.6	287.1	0.073754131	...
10	36683.9	272.6	0.000904304	...
11	36959.0	270.6	0.006109217	...
12	37132.6	269.3	0.041270629	...
13	38751.9	258.1	0.058697401	...

It has computed overall 13 excited states, but for the sake of visualization, and also consistency with the other geometries (as perhaps for other geometries there were computed either more or less than 13 states, due to the threshold we selected of 5.0 eV), we will consider a smaller number, for example up to 5 excited states. Now let us visualize the spectrum generated by one geometry. We will use the in-house written `plot_spectrum.py` script, which takes at most four arguments; remember, you will need the `matplotlib` and the `tkinter` python modules!. We can visualize them by executing

```
python3 plot_spectrum.py -h
```

which will print out the following:

```
usage: Plot spectrum from an ensemble of Orca calculations.
           Retrieve each output from the output geomN/geomN.inp.out,
           where N is the geometry of interest (A range can also
                 be defined)
       [-h] [-f F [F ...]] [-u UNITS] [-m MAX] [-cb COLOR_BARS]

optional arguments:
  -h, --help          show this help message and exit
  -f F [F ...]        Range of geometries to analyze
  -u UNITS            Units of energy (default = eV)
  -m MAX              maximum state to consider (default = 5)
  -cb COLOR_BARS      Color bars by state (default = False)
```

The first argument (`-f`) requests a range of geometries to analyze, for example, 0 9 stands for geometries 0 up to 9. The `-cb` argument asks for whether to present colored bars, that is, whether to color each bar depending on the transition index it corresponds to (within the script they are indexed as "states", but for different geometries the ordering of the states might switch, so this description is not entierly correct); True uses different colors. The other arguments are self-explanatory. Let us now plot the absorption spectrum of geometry 0 by executing:

```
python3 plot_spectrum.py -f 0 -u eV -m 5 -cb True
```

so that we will plot at most 5 excited states using eV as the energy units. The spectrum will consist of five vertical bars, remember, each bar represents an electronic transition. Now what is usually done in the literature when one geometry (in general the equilibrium geometry) is employed to compute the spectrum is to apply a broadening on top of each vertical bar (for example a broadening with a Gaussian function). This broadening is useful to better compare the calculation with the experimental spectrum but **does NOT have any physical meaning!**. We can apply the broadening in our spectrum by writing a value in eV inside the entry square below the plot, which indicates the so-called Full Width-at-Half Maximum (FWHM), a quantity proportional to the standard deviation if you use Gaussian broadenings. We can select, for example 0.5 eV. So far, so good, but now we want to account for part of that broadening with actual physical information, in out case, by accounting for molecular motion (we wrote *part* of the broadening, as there are other effects that contribute to it, for example Heisenberg's uncertainty principle, but this is discussion for another day). Thus, we will now plot the absorption spectrum for our ensemble of 10 geometries (yes, these are very few geometries, we would need hundreds to actually have "decent" results). Please execute again the script, as follows:

```
python3 plot_spectrum.py -f 0 9 -u eV -m 5 -cb True
```

where we are now considering geometries 0 to 9. We still have a bar spectrum, but now we can see that transitions towards different states "cluster" on different energy intervals, thus better resembling what we call a "spectral band". Of course we can also apply a broadening on top of each bar, but now, again, we see that each band correspond to different *vibronic* transitions. And that's it, we have seen how to compute absorption spectra using QM/MM and ensembles of geometries. Of course, this is far from all that can be done in this context, indeed, one could (and should) analyze the character of each of the electronic excitations, but for the time being we can stop here.

2.3 Part 2: Variation of the Size of the QM region

Up until now, we have only included the solute molecule in the QM region, but it might be of interest for some of you what could happen if we increase the number of molecules in that region. This is actually a very important question, as the absolute values will change when we change the QM region of the system. Of course we are mainly interested in Energy differences, and these should in principle vary little, for otherwise vaguely said, our microscopic approaches should not reproduce what we see in the laboratory. Of course in practice these differences often vary as we increase the size of the QM region (which is the same as saying,

we increase the accuracy of our method), but what we should expect is that they converge at some point. Here we will see what happens to the excitation energies of the first few excited states, by increasing the QM region from 1 water molecule to 5, but by no means should we expect convergence. Also, bear in mind that this analysis for excited states is not without flaws, as the character of some excited states might vary as we increase the number of water molecules.

Let us go now to the folder 2_qm_size. Please copy there the trajectory and the topology files of the previous section. There, you will find again the input files for the MoBioTools package. They will be exactly the same as before, with the exception that the `main.inp` file bears two extra sections for selecting molecules (residues) that are to be considered as solvent molecules, and the number of closest molecules (to the mask in qmmask) to be considered as QM molecules:

```
&main
  tpl = orca
  top = system.parm7
  traj = imaged.crd
  qmmask = :1
  geoms = 6
  solvmask = :WAT
  closest = 1
&end
```

We will consider a single geometry (to make the scaling consistent), for example geometry 6. We will have to generate one `orca` input file for each calculation, each with a different number of water molecules. To do so, we will use the `gen_inputs.sh` script, which will generate the folders `Nres-geom6`, where N goes from 1 to 5, and each folder contains the orca input with the desired number of water molecules. The script will also generate the `xyz_res` folder, which will contain the `xyz` coordinate files of all the inputs under consideration, for the sake of visualization. Please generate the input files as follows

```
./gen_inputs.sh
```

and then submit the QM/MM calculations using the `submit.sh` script:

```
nohup ./submit.sh &
```

In the meantime, we can visualize the snapshot we are considering (geometry 6), and how the QM region gradually increases. First, open vmd and load the initial trajectory (with the files `system.parm7` and the *imaged* coordinates on `imaged.crd`). Then from within `vmd`, open each of the xyz files contained in the `xyz_res` folder. If you change the representation of each loaded "molecule" (in the vmd jargon) one at a time, you will see which water molecules are being included in the QM region for each loaded xyz file.

Once the calculations are done, we will see how the energies of the first five excited sates have changed as we increased the size of the QM region. Please, execute the script `plot_comparison.py` as follows:

```
python3 plot_comparison.py
```

This will generate an image showing a scatter plot of the energies of the excitations under study as the number of water molecules goes from 1 to 5.

The goal of this exercise was of course not to produce accurate absorption spectra, but rather to see ow to tackle the problem of simulating these spectra on systems in complex environments. With this, the tutorial comes to an end, but of course if you have any questions, comments or curiosities, we will be happy to come back to you. Thank you for following us up to this point!